

# Bioinformática

con

~ñ

VOL.1

**Editores:**  
**Álvaro Sebastián**  
**Alberto Pascual-García**



# Bioinformática con $\tilde{N}$

Volumen I: Principios de Bioinformática

**Edición y coordinación:**

Álvaro Sebastián y Alberto Pascual-García

**Autores:**

F. Abascal, J. Aguirre, E. Andrés-León, D. Bajic, D. Baú, J. A. Bueren-Calabuig,  
Á. Cortés-Cabrera, I. Dotu, J. M. Fernández, H. G. D. Santos, B. García-Jiménez,  
R. Guantes, I. Irisarri, N. Jiménez-Lozano, J. Klett, R. Méndez, A. Morreale,  
A. Pascual-García, A. Perona, A. Sebastian, M. Stich, S. Tarazona, I. Yruela y R. Zardoya

**Portada:**

Enrique Sahagún (<http://www.scixel.es>)

**Maquetación:**

Álvaro Sebastián

Usando L<sup>A</sup>T<sub>E</sub>X(<http://www.latex-project.org>) y Texmaker  
(<http://www.xmlmath.net/texmaker>)

**Editorial:**

Libro autoeditado e impreso por CreateSpace (<http://www.createspace.com>)

**Depósito legal: SE-NNNNNNN**

**ISBN: NNNNNNNN**

**Copyright de la portada:**

© 2014 Enrique Sahagún

**Copyright de los textos:**

© 2014 Los respectivos autores

**Copyright de las figuras:**

© 2014 Los respectivos autores, si no se indica lo contrario

**Licencia de las figuras:**

La establecida por sus autores en los textos originales

**Licencia de los textos:**

Creative Commons BY-NC-SA 4.0 (<http://creativecommons.org/licenses/by-nc-sa/4.0/>)

La licencia Creative Commons BY-NC-SA 4.0 permite:



- **Compartir:** copiar y redistribuir el material en cualquier medio o formato
- **Adaptar:** mezclar, transformar y crear a partir del material

Bajo los siguientes términos:



**Atribución:** se debe dar el crédito de la obra a los autores originales, proveer un enlace a la licencia e indicar los cambios realizados.



**NoComercial:** no se puede hacer uso del material con fines comerciales.



**CompartirIgual:** Si se mezcla, transforma o crea nuevo material a partir de esta obra, sólo se podrá distribuir utilizando la misma licencia que la obra original.



# **Prólogo Volumen I. Principios de bioinformática.**



## Prólogo

La bioinformática y la biología computacional persiguen ordenar el conocimiento que se deriva del análisis de datos biológicos, fundamentalmente secuencias y genomas, con ayuda de algoritmos y sistemas informáticos. Por su naturaleza transversal han colonizado a otras disciplinas y ya son parte integral de los métodos de trabajo de áreas tan amplias como la biotecnología, la ecología, la biología o la medicina. Proyectos científicos de gran impacto en nuestras vidas, como la secuenciación del genoma humano, la caracterización de las diferentes variedades de cáncer, el diseño de nuevos medicamentos o la selección genómica en el sector agropecuario han sido y seguirán siendo posibles gracias, en gran medida, a los avances de la bioinformática. Todo esto ha ocurrido en los últimos 30 años, y a un ritmo muy difícil de seguir incluso para los especialistas. De hecho, es fácil abrumarse con el volumen de literatura al alcance de la mano, ya sea en canales convencionales como las revistas Bioinformatics o PLoS Computational Biology, editadas por científicos de prestigio, o foros como Biostar o SEQanswers, donde usuarios de todo el mundo comparten protocolos y experiencia al estilo peer to peer.

Como decía, la bioinformática ha evolucionado rápidamente en años recientes y eso se ha traducido en una demanda creciente de profesionales cualificados en el diseño y aplicación de esta clase de herramientas computacionales. Estos especialistas pueden tener currículum vitae muy distintos, habiendo recorrido itinerarios académicos posiblemente diferentes, pero finalmente han de tener una serie de habilidades fundamentales como son conocimientos en genética y biología molecular, dominio de lenguajes de programación y la capacidad de administrar y sacar el máximo rendimiento a los datos y el hardware disponible, ya sea en un laboratorio de investigación o una pequeña empresa biotecnológica. Aunque la bioinformática tiene una dimensión global y se habla en inglés, también es cierto que en muchas universidades a éste y al otro lado del Atlántico la lengua vehicular para la enseñanza es el español. En consecuencia hay un espacio y una demanda real de materiales educativos en español, y muestra de ellos son los cada vez más numerosos blogs y bitácoras que tratan sobre esta materia en nuestra lengua. Este libro es el esfuerzo colectivo de un buen elenco de investigadores expertos, en su mayoría por debajo de los 40 años, por contribuir a llenar ese espacio con contenidos científicos. En efecto, tras el texto pionero publicado por el profesor Lahoz-Beltrá en 2004, este libro es el primero escrito en nuestra lengua que hace un repaso tan amplio y ambicioso del área, y complementa de manera natural los recursos disponibles en la Red.

En esta edición inicial los autores tocan de forma exhaustiva una selección de temas fundamentales de la biología computacional, y por tanto es de esperar que este libro se convierta en un recurso valioso para estudiantes y profesores de las universidades de habla hispana. Para los estudiantes, porque aquí podrán encontrar material para ahondar en su estudio de la bioinformática, incluyendo citas a artículos de la literatura científica, normalmente en inglés; para los profesores universitarios, porque podrán apoyarse en este libro para programar y dotar de contenido a las asignaturas y materias relacionadas con la bioinformática, que todavía en muchas universidades son claramente insuficientes. Finalmente, seguramente este libro contribuya a introducir y traducir al español el caudal constante de términos y conceptos de la bioinformática. El reto que tiene por delante será sin duda el de la renovación permanente para no sucumbir a la evolución del área, de modo que los temas incluidos sean representativos de las áreas vigentes de la disciplina. La filosofía colaborativa de este proyecto, donde tanto el editor como los autores han participado en la selección de materiales y en la toma de decisiones editoriales, parece adecuada para este fin.

Zaragoza, Septiembre de 2014

El Dr. Bruno Contreras es director del Laboratorio de Biología Computacional en la Estación Experimental de Aula Dei (EEAD/CSIC, Zaragoza, España) y profesor invitado de la Universidad Autónoma de México (Cuernavaca, México).

# **Prólogo Volumen II. Biología estructural y de sistemas.**



## Prólogo

Mi primera impresión cuando examiné la recopilación de capítulos que se agrupan en este volumen doble fue la de que, ¡por fin!, alguien había cogido al toro por los cuernos y acometido la ingente tarea de explicar y dar forma a una serie de conceptos y disciplinas de actualidad relacionados con los nuevos enfoques sobre los seres vivos. Cabría pensar que este conocimiento, que dirige una buena parte de los esfuerzos científicos encaminados a comprender la vida y aliviar las enfermedades, se estuviera impartiendo en todos o la mayoría de los grados universitarios actuales en ciencias de la salud. La sobria realidad es que, lamentablemente, sigue siendo mayoritariamente ignorado aunque esta carencia se pueda paliar posteriormente en másteres o cursos de especialización.

La sorpresa que vino a continuación fue que esta valentía correspondía a dos jóvenes investigadores que han conseguido coordinar las aportaciones de otros colegas igual de jóvenes (¡o incluso más!), con flamantes doctorados en materias relacionadas con los temas tratados, y que demuestran haber recogido el testigo recibido de profesores y maestros para facilitar la tarea a cualquier persona hispanohablante interesada en introducirse en el fascinante mundo de la información biológica moderna, su tratamiento y algunas de sus aplicaciones. Entre estas últimas se encuentran metodologías de diseño de fármacos basadas en el conocimiento estructural con detalle atómico de la macromolécula diana, la simulación de los movimientos de estos componentes de la maquinaria que hace posible la vida y el apasionante mundo de los complejos procesos interrelacionados que constituyen la biología de sistemas.

Estos dos volúmenes son como los abrigos crecederos que se compran a los niños para que les duren más de una temporada. Por un parte, cada capítulo recoge una información más que abundante para una introducción a la materia y, por otro lado, hay espacio suficiente entre capítulos para llenar los huecos que una obra de este tipo necesariamente deja sin cubrir, a la espera de ediciones posteriores.

Desde estas breves líneas envío mi felicitación a coordinadores y autores, agradeciéndoles su dedicación y buen hacer, que tendrá como merecida recompensa el reconocimiento de sus lectores, que no dudo serán numerosos a ambos lados del Atlántico.

Federico Gago Badenas

Madrid, Septiembre de 2014

El Dr. Federico Gago es director del grupo de investigación ‘Mecanismo de acción de moléculas con actividad biológica’ en la Universidad de Alcalá (Madrid, España), además de catedrático y profesor en la misma universidad.



# **Autores**



## **Álvaro Sebastián Yagüe (editor)**

Álvaro Sebastián es licenciado en Química y Bioquímica, ambos por la Universidad de Zaragoza (2005). Cursó los estudios de máster en Bioinformática en la Universidad Complutense de Madrid y realizó prácticas en la empresa Crys-tax Pharmaceuticals (Barcelona). Posteriormente volvió al *wet lab* para realizar la tesis doctoral sobre la proteína humana ITIH4 en la Universidad de Zaragoza donde también cursó estudios de doctorado en Ingeniería Biomédica.

Al final del doctorado comenzó una aventura empresarial fundando Idibay Consulting S.L., empresa de diseño web y consultoría informática. En el año 2009 fue contratado como investigador del CSIC en el proyecto europeo STREG para el estudio de la regulación transcripcional en plantas en el laboratorio del Dr. Bruno Contreras en la Estación Experimental de Aula Dei. En 2013 comenzó su periplo internacional realizando una estancia en el laboratorio del Prof. Janusz Bujnicki para el diseño de potenciales de *docking* proteína-proteína y más tarde se incorporó al laboratorio de Biología Evolutiva en la Universidad Adam Mickiewicz (Poznan, Polonia), donde trabaja actualmente con el Prof. Jacek Radwan en el genotipado de la familia de genes del complejo mayor de histocompatibilidad y del receptor de linfocitos T usando tecnologías de secuenciación de nueva generación.

Su corta carrera científica se caracteriza por un vaivén en diferentes áreas y técnicas, desde la síntesis química asimétrica, pasando por el laboratorio bioquímico hasta finalizar en el campo de la bioinformática. La última “aventura” científica, y quizás la más arriesgada y compleja, ha sido coordinar y editar el presente libro.



## **Alberto Pascual García (editor)**

Alberto Pascual García es licenciado en Física por la Universidad Complutense de Madrid y máster en Biofísica por la Universidad Autónoma de Madrid. Ha trabajado en el Laboratorio de Circuitos Neuronales del Hospital Ramón y Cajal, y en la Unidad de Bioinformática del Centro de Biología Molecular Severo Ochoa (CSIC-UAM), donde finaliza su tesis doctoral sobre patrones emergentes en sistemas biológicos complejos, dirigida por el Dr. Ugo Bastolla.

Ha participado en diversos proyectos de investigación en temas tan diversos como evolución de estructura de proteínas, redes de interacción mutualistas o ecología bacteriana, desde una perspectiva multidisciplinar. Ha realizado estancias de investigación en el laboratorio del Prof. Andrés Moya (Centro de Investigación en Salud Pública de Valencia, España) y del Prof. Julián Echave (Universidad Nacional de San Martín, Argentina). En los últimos años ha sido coordinador y profesor de la asignatura de Bioinformática del Máster en Biofísica de la Universidad Autónoma de Madrid.



## Federico Abascal Sebastián de Erice



Federico Abascal se licenció y doctoró en Biología Molecular en la Universidad Autónoma de Madrid. Ha desarrollado su labor de investigación en campos diversos de la Biología, como el análisis de secuencias de proteínas y la genómica, siempre utilizando los ordenadores como herramienta y desde un enfoque evolutivo.

Actualmente trabaja como investigador contratado en el Centro Nacional de Investigaciones Oncológicas, en el grupo del Prof. Alfonso Valencia, con quien hizo la tesis doctoral en el Centro Nacional de Biotecnología. También ha trabajado con los Prof. David Posada (Universidad de Vigo) y Rafael Zardoya (Museo Nacional de Ciencias Naturales), principales culpables de su afición a lo evolutivo.

## Jacobo Aguirre Araujo

Jacobo Aguirre Araujo (Madrid, 1975) se licenció en Física, especialidad Astrofísica, por la Universidad Complutense de Madrid en 1999. Fue profesor de la Universidad Rey Juan Carlos hasta 2006, investigador del Centro de Astrobiología (CSIC-INTA) hasta junio de 2014, y actualmente trabaja en el Centro Nacional de Biotecnología (CSIC).

Su investigación versa sobre el análisis y la modelización de procesos evolutivos en física y biología en el contexto de la dinámica no lineal y las redes complejas. Actualmente, está centrado en el estudio de la evolución de virus y RNA. Ha realizado estancias de investigación en Dinamarca, Alemania y EEUU. En 2003 fundó el Grupo de Astronomía de la URJC y en 2005 recibió el Premio Nacional al Investigador Novel en su modalidad de Física Teórica.



## Eduardo Andrés León



Eduardo Andrés León trabaja en el Instituto de Biomedicina de Sevilla (IBIS) dentro del grupo de Biología Computacional y Bioinformática (CbBio), donde es el responsable técnico de la unidad de cálculo científico de alto rendimiento. Estudió Biología Molecular en la Universidad Autónoma de Madrid y ha trabajado en el Centro Nacional de Biotecnología (CNB-CSIC), Instituto Nacional de Bioinformática y Centro Nacional de Investigaciones Oncológicas (CNIO).

Actualmente colabora con la Fundación Española de Hipercolesterolemia Familiar y con la Escuela Nacional de Sanidad del Instituto de salud Carlos III, donde desarrolla labores docentes. Está interesado en el análisis y estudio integrativo de datos genómicos a gran escala y en la regulación génica basada en elementos no codificantes como los “*small RNAs*”.

## Djordje Bajic

Djordje es licenciado en Bioquímica y máster en Biofísica por la Universidad Autónoma de Madrid. Actualmente es estudiante de doctorado en el laboratorio de Lógica de Sistemas Genómicos en el Centro Nacional de Biotecnología (Madrid).

Su investigación versa en el uso de modelos computacionales y genómica comparativa para comprender la evolución de la expresión génica.



## Davide Baù



Después de haber obtenido un máster en Química en la Universidad de Padua y completar un curso de Bioinformática en la Universidad de Colonia, Davide se trasladó a la Universidad de Dublín (UCD), donde, en 2008, obtuvo su doctorado en la Facultad de Ciencias de la Computación e Informática, bajo la supervisión del Dr. Gianluca Pollastri. Su tesis se centró en el desarrollo de un algoritmo de predicción de estructuras de proteínas bajo la guía de un potencial estadístico basado en técnicas de aprendizaje automático o “*machine learning*”.

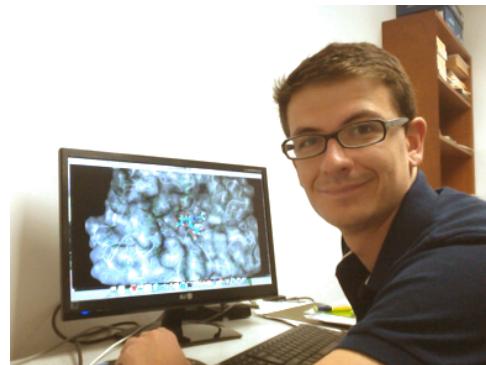
Davide realizó su postdoctorado en el laboratorio del Dr. Marc Martí-Renom, donde comenzó a trabajar en la determinación de estructuras de dominios genómicos y genomas. Durante este tiempo, desarrolló los métodos que llevaron a la determinación del primer modelo de alta resolución de una región genómica humana, el dominio  $\alpha$ -globina y del

primer modelo tridimensional del genoma completo de una bacteria (*Caulobacter crescentus*). Actualmente está involucrado en diferentes colaboraciones que tienen como objetivo determinar la arquitectura del genoma de varios organismos incluyendo humano, levadura, *Mycoplasma pneumoniae*, mosca y ratón.

## Juan Bueren Calabuig

Juan Bueren es licenciado en Farmacia por la Universidad Complutense de Madrid y doctor en Farmacología por la Universidad de Alcalá (Madrid).

Juan ha realizado estudios de mecánica cuántica y dinámica molecular sobre el mecanismo catalítico de diversas enzimas implicadas en la evasión del sistema inmune por el parásito *Trypanosoma cruzi* causante de la enfermedad de Chagas en la Universidad de Florida. También ha trabajado en la Universidad de Edimburgo con el Dr. Julien Michel estudiando el mecanismo de acción de proteínas intrínsecamente desordenadas y a partir de septiembre de 2014 será investigador en la Universidad de Dundee para realizar estudios de proteínas de membrana.



## Ivan Dotu

Ivan Dotu se doctoró en Ingeniería Informática en la Universidad Autónoma de Madrid, especializándose en Programación con Restricciones y Búsqueda Local. En 2007 consiguió una beca de la Fundación Caja Madrid para realizar un postdoctorado en Brown University, donde trabajó en predicción de estructura de proteínas. Tras un año trabajando en la empresa privada, Ivan Dotu volvió a la academia con un puesto de Profesor Investigador en el Departamento de Biología de Boston College, donde trabajó, dentro del laboratorio de Biología Estructural liderado por el Prof. Peter Clote, en temas relacionados con la estructura de RNA y la búsqueda y diseño de RNA funcional.

Ivan Dotu ha publicado decenas de artículos en diversas áreas, ha sido co-PI de un proyecto de NSF y ha co-organizado 2 escuelas de verano en Boston College sobre Métodos Computacionales de RNA. También ha sido o es miembro del comité de programa de conferencias como CP, ECAI y AAAI y revisor de artículos para revistas como BMC Evolutionary Biology, PLOS ONE, PLOS Computational Biology o Nucleic Acids Research. Actualmente, Ivan Dotu es un investigador visitante en la Universidad Politécnica de Cataluña y es colaborador renumerado de un proyecto NIH sobre la interacción entre RNA y proteínas.



## José María Fernández González



José María Fernández es un bioinformático perteneciente al Nodo 2 del Instituto Nacional de Bioinformática, en el Programa de Biología Estructural y Biocomputación del Centro Nacional de Investigaciones Oncológicas. Tiene el título de Ingeniero en Informática por la Universidad de Málaga, y al acabar los estudios se interesó por la bioinformática, entrando en 1999 a formar parte del Protein Design Group en el Centro Nacional de Bioinformática. Allí participó en varios proyectos, entre ellos REGIA (REgulatory Gene Initiative in Arabidopsis), PlaNet Consortium y el proyecto de secuenciación de *Buchnera aphidicola*, creó para el proyecto ORIEL los iHOP web services y colaboró en el germen del actual INB. En 2002 obtuvo el Título de Estudios Avanzados en el departamento de Informática de la Universidad Autónoma de Madrid. En 2006 se trasladó junto con el grupo de Alfonso Valencia al CNIO, ya como bioinformático del INB, participando en proyectos como EMBRACE, ENFIN, ICGC, BLUEPRINT y RD-Connect.

Sus principales líneas de desarrollo a lo largo de estos años han sido el procesamiento, consulta y análisis de secuencias y de grandes volúmenes de información; el diseño y uso de bases de datos de distinto tipo (SQL, XML, NoSQL) con el volumen de datos a nivel bioinformático, y de modelos de datos que se amoldaran a las necesidades de la bioinformática; desarrollo de servicios web en distintos paradigmas (REST, SOAP+WSDL, etc...), así como la colaboración en el diseño e implementación del estándar BioMOBY.

## Álvaro Cortés Cabrera



Álvaro Cortés es licenciado en Farmacia y máster en Biofísica. Actualmente compagina sus estudios de doctorado en la Unidad de Bioinformática del Centro de Biología Molecular Severo Ochoa y su trabajo como profesor en la Universidad de Alcalá.

Su investigación versa en estudios de *docking* proteína-ligando y proteína-proteína.

## Beatriz García Jiménez

Beatriz García Jiménez es investigadora postdoctoral Isaac Peral en el Grupo de Bioinformática del Centro de Biotecnología y Genómica de Plantas UPM-INIA desde Septiembre del 2013.

Obtuvo su tesis doctoral, titulada “Anotación Funcional de Proteínas basada en Representación Relacional en el entorno de la Biología de Sistemas”, por la Universidad Carlos III de Madrid (UC3M) en 2012, dirigida por el Dr. Alfonso Valencia (Bio-CNIO) y la Dra. Araceli Sanchis (Inf-UC3M). Fue profesora ayudante del departamento de Informática de la UC3M desde Marzo del 2006 hasta Agosto de 2013. Realizó una estancia postdoctoral en el departamento de Biología Computacional y Algoritmos aplicados en el Instituto Max-Planck de Informática (MPII) de Febrero a Junio de 2013.

Su tesis ganó el Premio Nacional a la mejor tesis, en el área de ciencias experimentales y tecnológicas, concedido por la Real Academia de Doctores de España. Además, ha recibido otros premios de la UC3M (Premio Extraordinario de Doctorado, 2011-2012), y de la Comunidad de Madrid (Aprovechamiento Académico Excelente, 2004-2005), entre otros.

Está especializada en aprendizaje automático aplicado a la bioinformática, principalmente en predicción de anotación funcional. Sus intereses científicos incluyen representación relacional del conocimiento, minería de datos, anotación funcional, *workflows* y redes de proteínas.

Actualmente estudia cómo extraer información de los *workflows* bioinformáticos, anotándolos semánticamente e identificando fragmentos reusables de conocimiento científico bioinformático, y participa en un proyecto de localización de marcadores de Alzheimer.



## **Helena Gomes Dos Santos**



Helena Gomes Dos Santos se licenció en Biología por la Universidad Autónoma de Madrid, donde además cursó un máster en Biofísica y recientemente se doctoró en Biociencias Moleculares bajo la supervisión del Dr. Ugo Bastolla y el Dr. Antonio Morreale (Centro de Biología Molecular Severo Ochoa, CSIC-UAM, Madrid).

Sus inicios en la bioinformática se remontan al último año de carrera, el laboratorio del Dr. Angelo Messina (Universidad de Catania). Desde muy temprano mostró un especial interés en el mundo computacional, colaborando en el desarrollo e implementación de nuevas herramientas bioinformáticas así como su aplicación en casos reales mediante numerosas colaboraciones con grupos experimentales.

Su labor investigadora se ha centrado principalmente en el estudio de la variabilidad estructural, la dinámica y las posibles interacciones entre proteínas y otras moléculas. Como complemento a dicha formación, en la actualidad se encuentra realizando su investigación postdoctoral en el laboratorio de la Dra. Siltberg-Liberles (Universidad Internacional de Florida), donde evolución y dinámica de proteínas se combinan para ofrecernos una imagen más completa del funcionamiento y posible modulación de las funciones proteicas con una directa aplicación en la mejora del diseño de fármacos asistido por ordenador y la terapia personalizada.

## **Raúl Guantes Navacerrada**

Raúl Guantes es profesor contratado doctor de la Universidad Autónoma de Madrid, donde coordina desde el año 2008 un programa oficial de posgrado en Biofísica y dirige el laboratorio de Biodinámica y Biología Computacional.

Es doctor en Ciencias Químicas por la misma universidad y ha investigado en diversos campos de la Física y la Biología desde una perspectiva teórica: física no lineal y de sistemas complejos, física estadística, física de superficies, neurociencia y biología de sistemas.

Desde el año 2006 centra su investigación en el estudio de las redes biológicas y sus principios de diseño, así como en los orígenes y consecuencias de la variabilidad celular no genética.



## Iker Irisarri Anguita



Iker Irisarri se licenció en Biología por la Universidad de Navarra y obtuvo su doctorado por la Universidad Autónoma de Madrid en 2012, desempeñando su labor investigadora en el Museo Nacional de Ciencias Naturales (CSIC) bajo la dirección del Prof. Rafael Zardoya. Durante este periodo, se especializó en filogenia y evolución molecular de animales, y realizó estancias de investigación en Alemania, Australia y California.

En la actualidad, trabaja como investigador postdoctoral en la Universidad de Constanza (Alemania) en filogenómica y genómica comparada de vertebrados.

## Natalia Jiménez Lozano

Natalia Jiménez es licenciada en Bioquímica por la Universidad de Granada. Realizó su doctorado en el Departamento de Biocomputación del Centro Nacional de Biotecnología.

Cuando terminó el doctorado emprendió el reto de montar desde cero una Unidad de Bioinformática perteneciente al Instituto Nacional de Bioinformática, al que ha pertenecido hasta el año 2012, en el que entró a formar parte de la empresa Bull con la responsabilidad de desarrollar el negocio de la sanidad y ciencias de la vida a nivel internacional.



## Javier Klett Arroyo



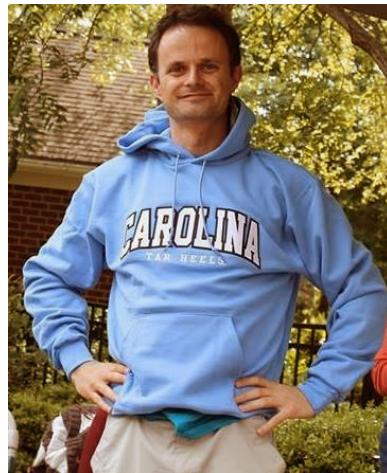
Javier Klett es licenciado en Matemáticas, máster en Biofísica por la Universidad Autónoma de Madrid y realizó su doctorado en la Unidad de Bioinformática del Centro de Biología Molecular Severo Ochoa (Madrid).

Su especialidad son los estudios energéticos y potenciales de unión proteína-proteína. Actualmente compagina la investigación postdoctoral en el grupo de Química y Biología Computacional en el Centro de Investigaciones Biológicas de Madrid y su trabajo como profesor en el Departamento de Bioingeniería e Ingeniería Aeroespacial de la Universidad Carlos III (Madrid).

## Raúl Méndez Giráldez

Raúl Méndez Giráldez, es licenciado en Química por la Universidad de Barcelona y doctor en Ciencias por la Universidad Libre de Bruselas. Su carrera profesional se ha desarrollado en el campo de la bioinformática estructural, inicialmente durante su doctorado evaluando los métodos para predecir la estructura de los complejos de proteína, y más tarde se ha ido centrando en el desarrollo de métodos computacionales de red elástica para estudiar la flexibilidad de las proteínas.

En la actualidad trabaja como investigador postdoctoral en el departamento de Bioquímica y Biofísica de la Universidad de Carolina del Norte (EEUU) utilizando métodos de dinámica molecular y diseño racional de proteínas con aplicaciones terapéuticas.



## Antonio Morreale



Antonio Morreale estudió Ciencias Químicas en la Universidad de Alcalá de Henares, Madrid, donde se doctoró en el año 2001 bajo la supervisión de la Dra. Isabel Iriepa tras una estancia de dieciocho meses en el Chemistry Department de la IUPUI (Indiana, USA) como estudiante de doctorado donde trabajó con los Profs. Kenny B. Lipkowitz (codirector de su tesis) y Donald B. Boyd. Después realizó una estancia postdoctoral en el laboratorio de los Profs. Modesto Orozco y Javier Luque durante casi dos años y medio.

A su vuelta a Madrid (2004) se incorporó a la Unidad de Bioinformática (UB) creada por el Dr. Ángel R. Ortiz en el centro de Biología Molecular Severo Ochoa (CBMSO), primero como postdoctoral asociado (2004-2008) y luego como director de la

línea de diseño de fármacos y codirector del Servicio de Bioinformática del CBMSO. A finales de 2012

se incorpora como tecnólogo senior al área de bioenergía de la compañía Repsol S.A.

Antonio Morreale es autor de unos 60 artículos de investigación en publicaciones especializadas incluyendo revisiones bibliográficas y capítulos de libro. Además, es coinventor de varias patentes, dos de las cuales han sido licenciadas. Ha dirigido cuatro tesis doctorales y ha participado como organizador y profesor en numerosos cursos y másteres.

Su trayectoria científica ha sido una evolución continua desde los principios que rigen las interacciones entre moléculas (tanto cuánticos como clásicos) hasta la aplicación de éstos en el diseño racional de fármacos. En la actualidad trabaja en el modelado y simulación de reacciones catalíticas claves en ciertas rutas metabólicas con la finalidad de obtener nuevos biocombustibles.

## Almudena Perona

Almudena Perona es directora de proyectos en la empresa SmartLigs, compañía biotecnológica dedicada al descubrimiento y desarrollo de nuevos fármacos mediante tecnologías de química computacional. Doctora en Química Orgánica por la Universidad Nacional de Educación a Distancia (UNED), especialista en síntesis orgánica de heterociclos y caracterización de moléculas orgánicas sencillas por NMR.

Como investigadora postdoctoral en la Unidad de Bioinformática del Centro de Biología Molecular Severo Ochoa (CSIC-UAM) ha colaborado en proyectos de diseño de fármacos mediante estudio *in silico* de la interacción entre proteínas y moléculas sencillas, usando herramientas bioinformáticas, principalmente programas de docking proteína-ligando, proteína-proteína y dinámica molecular.



## Michael Stich



Después de estudiar Física en la Freie Universität Berlin y la Universidad de Granada, Michael realizó el doctorado en el Fritz-Haber-Institut de la Max Planck Society en Berlín que terminó en el año 2003.

Obtuvo una beca Marie Curie postdoctoral para estar en la Universidad Complutense de Madrid, y trabajó de postdoc y posteriormente como investigador contratado en el Centro de Astrobiología (CSIC/INTA), a la vez de ser profesor asociado en la Universidad Politécnica de Madrid.

Después de ser investigador asociado en la Harvard University, se incorporó como docente en la Aston University (Birmingham, UK) en 2013. Su trabajo reciente está enfocado en la modelización de sistemas biológicos complejos y de procesos evolutivos, p.ej. RNA.

## Rafael Zardoya San Sebastián

Rafael Zardoya se doctoró en Biología en la Universidad Complutense de Madrid en 1994. Tras una estancia postdoctoral de 2 años en la Universidad del Estado de Nueva York en Stony Brook (EEUU), regresó en 1997 al Museo Nacional de Ciencias Naturales de Madrid donde es profesor de investigación del CSIC desde 2008.

Su línea de investigación se centra en el estudio de los mecanismos evolutivos que generan la diversidad biológica. En particular utiliza secuencias de DNA (genomas mitocondriales, transcriptomas) para la reconstrucción de las relaciones filogenéticas de los grandes grupos de vertebrados y moluscos y realiza análisis genómico/transcriptómicos comparados en un marco evolutivo.



## Sonia Tarazona Campos



Sonia Tarazona se licenció en Matemáticas (1997) y Ciencias y Técnicas Estadísticas (2001) por la Universitat de València. Desde 2005 es profesora asociada en el Departamento de Estadística e Investigación Operativa Aplicadas y Calidad de la Universitat Politècnica de València.

Desde 2008 forma parte del grupo de Genómica de la Expresión Génica liderado por la Dra. Ana Conesa en el Centro de Investigación Príncipe Felipe (Valencia). Durante este tiempo, ha colaborado en diversos proyectos de investigación y actualmente forma parte del proyecto europeo STATegra cuyo objetivo es el desarrollo de métodos estadísticos para la integración de distintos datos “ómicos”.

En paralelo, ha realizado su tesis doctoral que lleva como título “Statistical methods for transcriptomics: from microarrays to RNA-seq” y que será defendida en breve.

## Inmaculada Yruela Guerrero

Inmaculada Yruela es doctora en Química por la Universidad de Sevilla. Después de realizar una estancia postdoctoral en el Max-Planck Institute for Bioinorganic Chemistry en Mülheim and der Ruhr (Alemania) ocupó su actual posición de investigador científico en la Estación Experimental de Aula Dei del Consejo Superior de Investigaciones Científicas (EEAD-CSIC), Zaragoza.

Su investigación se ha centrado en el estudio de la estructura y la función de los fotosistemas y metaloproteínas que intervienen en el transporte electrónico fotosintético, en la homeostasis del Cu en el cloroplasto y en aspectos evolutivos de las enzimas FAD sintetasas cloroplásticas.

Recientemente sus investigaciones se orientan a entender la dinámica evolutiva de las proteínas intrínsecamente desordenadas (IDPs) en plantas y descubrir nuevas redes de regulación transcripcional en estos organismos. Sus actuales proyectos se desarrollan en el Grupo de Biología Computacional y Estructural de la EEAD-CSIC y colabora con el Instituto de Biocomputación y Física de Sistemas Complejos de la Universidad de Zaragoza.



# Índice general

<b>I Fundamentos de bioinformática, estadística y gestión de información científica</b>	<b>1</b>
<b>1. Bases de datos biomédicas, servidores web y otros recursos online</b>	<b>3</b>
1.1. Introducción . . . . .	3
1.1.1. ¿Qué es un dato biológico? . . . . .	3
1.1.2. Dato crudo y dato procesado . . . . .	3
1.1.3. Dato, información y conocimiento . . . . .	4
1.1.4. Características de los datos biológicos . . . . .	4
1.1.5. Tipos de datos . . . . .	5
1.1.6. Ciclo de vida de los datos biológicos . . . . .	6
1.2. Historia de las bases de datos . . . . .	7
1.2.1. ¿Por qué son esenciales las bases de datos? . . . . .	7
1.2.2. Diversidad de las bases de datos . . . . .	8
1.3. Estructura de las bases de datos . . . . .	9
1.4. Interrelación entre bases de datos . . . . .	10
1.5. Calidad y revisión de los datos . . . . .	11
1.5.1. Anotación de los datos . . . . .	11
1.5.2. Vocabularios controlados y ontologías . . . . .	12
1.5.3. Revisores ¿quiénes son y qué hacen? . . . . .	13
1.6. Números de acceso . . . . .	14
1.7. Gestión de las bases de datos . . . . .	14
1.8. Tipos de bases de datos . . . . .	15
1.8.1. Procedencia de los datos . . . . .	15
1.8.2. Revisión de los datos . . . . .	15
1.8.3. Clasificación EBI . . . . .	15
1.8.4. Clasificación NCBI . . . . .	16
1.8.5. Clasificación <i>NAR</i> . . . . .	16
1.8.6. Redundancia . . . . .	16
1.8.7. Acceso . . . . .	17
1.8.8. Búsqueda . . . . .	18
1.9. Bases de datos más relevantes . . . . .	18
1.9.1. UniProt Knowledgebase . . . . .	19
1.9.2. GenBank . . . . .	21
1.10. Obtención de secuencias y formatos . . . . .	22
1.11. Ejercicio . . . . .	23
1.12. Bibliografía . . . . .	27
<b>2. Gestión de bibliografía e información y escritura de artículos científicos</b>	<b>29</b>

2.1.	Introducción . . . . .	29
2.2.	Bases de datos de literatura científica . . . . .	30
2.2.1.	Servidores de artículos por publicar . . . . .	31
2.2.2.	PubMed y PubMed Central . . . . .	31
2.2.3.	Google Scholar . . . . .	34
2.3.	Índices y medidas de impacto científico . . . . .	37
2.4.	Gestores de bibliografía . . . . .	37
2.4.1.	Zotero . . . . .	38
2.4.2.	Mendeley . . . . .	39
2.4.3.	Biblioteca de Google Scholar . . . . .	42
2.5.	Escritura de artículos . . . . .	43
2.5.1.	Formato y plantillas de artículos . . . . .	44
2.5.2.	Proceso de envío y revisión de artículos . . . . .	45
2.6.	Primeros pasos con LaTeX . . . . .	48
2.6.1.	Crear un artículo con LaTeX . . . . .	48
2.6.2.	Compilar código LaTeX . . . . .	51
2.6.3.	Editores LaTeX . . . . .	52
2.6.4.	Instalación de una distribución LaTeX . . . . .	53
2.7.	Bibliografía . . . . .	55
<b>3.</b>	<b>Estadística y R</b> . . . . .	<b>57</b>
3.1.	Introducción . . . . .	57
3.1.1.	Población y muestra . . . . .	57
3.1.2.	Variables aleatorias . . . . .	58
3.1.3.	Introducción a R . . . . .	58
3.2.	Estadística descriptiva . . . . .	60
3.2.1.	Parámetros de posición y dispersión . . . . .	60
3.2.2.	Gráficos en R . . . . .	61
3.2.3.	Ejercicios . . . . .	65
3.3.	Probabilidad . . . . .	66
3.3.1.	Definición y propiedades . . . . .	66
3.3.2.	Distribuciones discretas . . . . .	67
3.3.3.	Distribuciones continuas . . . . .	69
3.3.4.	Ejercicios . . . . .	73
3.4.	Inferencia estadística . . . . .	74
3.4.1.	Estadística paramétrica . . . . .	74
3.4.2.	Estadística no paramétrica . . . . .	78
3.4.3.	Remuestreo . . . . .	78
3.4.4.	Corrección por tests múltiples . . . . .	79
3.4.5.	Enfoque bayesiano . . . . .	80
3.4.6.	Ejercicios . . . . .	81
3.5.	Modelos lineales . . . . .	83
3.5.1.	Modelos ANOVA . . . . .	83
3.5.2.	Regresión lineal . . . . .	86
3.5.3.	Modelos lineales generalizados . . . . .	90
3.5.4.	Ejercicios . . . . .	91
3.6.	Métodos multivariantes . . . . .	92
3.6.1.	Clustering . . . . .	93
3.6.2.	Análisis de Componentes Principales . . . . .	94

3.7. Bibliografía . . . . .	97
<b>4. Fundamentos de programación</b>	<b>99</b>
4.1. Introducción . . . . .	99
4.2. Tipos de lenguajes de programación . . . . .	99
4.2.1. Abstracción . . . . .	99
4.2.2. Ejecución . . . . .	100
4.2.3. Paradigma de programación. . . . .	100
4.2.4. ¿Que lenguaje de programación usar? . . . . .	101
4.3. Consola de Linux, Mac OS X y Windows. Operaciones básicas . . . . .	102
4.3.1. Tipos de Shell . . . . .	103
4.4. Perl y BioPerl . . . . .	104
4.4.1. Introducción a Perl . . . . .	104
4.4.2. Instalación . . . . .	105
4.4.3. Programando en Perl . . . . .	106
4.4.4. BioPerl y la programación orientada a objetos . . . . .	112
4.4.5. Programando con BioPerl . . . . .	113
4.4.6. Ejercicios con BioPerl . . . . .	114
4.4.7. Resumen . . . . .	115
4.5. Bibliografía . . . . .	117
<b>5. Minería de datos</b>	<b>119</b>
5.1. Introducción . . . . .	119
5.1.1. Aplicaciones de la minería de datos . . . . .	120
5.1.2. Conceptos básicos . . . . .	121
5.1.3. Herramienta: Weka . . . . .	122
5.2. Retos de la minería de datos . . . . .	125
5.3. Recopilación y selección de datos . . . . .	126
5.3.1. Repositorios de datos biológicos. . . . .	127
5.3.2. Servicios web . . . . .	129
5.3.3. Librerías de programación . . . . .	135
5.4. Preprocesamiento y transformación de datos . . . . .	136
5.4.1. Balanceo de clases . . . . .	137
5.4.2. Gestión de valores desconocidos . . . . .	138
5.4.3. Normalización . . . . .	139
5.4.4. Discretización . . . . .	139
5.4.5. Selección de atributos . . . . .	140
5.4.6. Extracción de características . . . . .	142
5.4.7. Conjuntos no redundantes . . . . .	142
5.5. Modelado. Aprendizaje automático . . . . .	143
5.6. Evaluación e interpretación de resultados . . . . .	145
5.6.1. Estimación del rendimiento . . . . .	145
5.6.2. Medidas de rendimiento unidimensionales . . . . .	146
5.6.3. Medidas de rendimiento bidimensionales . . . . .	150
5.6.4. Interpretación biológica . . . . .	152
5.7. Bibliografía . . . . .	153
<b>6. Computación paralela y clústeres de cálculo</b>	<b>157</b>
6.1. Computación paralela y ejemplos de programación . . . . .	158

6.1.1. Paralelización en Perl ( <i>threads</i> ) . . . . .	160
6.1.2. Paralelización en Python ( <i>multiprocessing</i> y <i>subprocess</i> ) . . . . .	163
6.2. Clústeres de cálculo . . . . .	167
6.2.1. Nodos, memoria, procesadores, núcleos, procesos e hilos . . . . .	168
6.2.2. Rendimiento . . . . .	169
6.3. Conexión remota . . . . .	171
6.3.1. Protocolo SSH . . . . .	171
6.3.2. Nodo de acceso . . . . .	172
6.4. Transferencia remota de archivos . . . . .	172
6.4.1. Protocolo SCP . . . . .	173
6.5. Sistema gestor de colas . . . . .	173
6.6. <i>Sun Grid Engine</i> . . . . .	175
6.6.1. Ejecución ordinaria de trabajos ( <i>qrsh</i> ) . . . . .	175
6.6.2. Procesamiento por lotes . . . . .	176
6.6.3. Ejecución de procesos por lotes ( <i>qsub</i> ) . . . . .	176
6.6.4. Control del estado de los trabajos ( <i>qstat</i> ) . . . . .	178
6.6.5. Eliminación de procesos ( <i>qdel</i> ), errores y salida estándar . . . . .	180
6.6.6. Requerimientos de memoria . . . . .	180
6.6.7. Uso de múltiples <i>slots</i> . . . . .	182
<b>II Macromoléculas biológicas, alineamiento de secuencias y filogenia</b>	<b>183</b>
<b>7. Macromoléculas biológicas: proteínas, DNA y RNA</b>	<b>185</b>
7.1. Introducción . . . . .	185
7.2. Genes y proteínas . . . . .	186
7.3. Estructura primaria de DNA y RNA . . . . .	188
7.4. El código genético . . . . .	189
7.5. Aminoácidos y enlace peptídico . . . . .	190
7.5.1. Aminoácidos . . . . .	190
7.5.2. Enlace peptídico . . . . .	196
7.5.3. Ángulos de torsión . . . . .	197
7.6. Niveles estructurales en proteínas . . . . .	199
7.6.1. Estructura primaria . . . . .	199
7.6.2. Estructura secundaria . . . . .	200
7.6.3. Estructura terciaria . . . . .	202
7.6.4. Estructura cuaternaria . . . . .	202
7.7. Métodos empíricos . . . . .	203
7.7.1. Cristalización . . . . .	203
7.7.2. Difracción de rayos X . . . . .	204
7.7.3. Resonancia magnética nuclear . . . . .	205
7.8. Herramientas bioinformáticas . . . . .	206
7.8.1. Protein Data Bank . . . . .	206
7.8.2. Visualización de estructuras en 3D . . . . .	206
7.9. Bibliografía . . . . .	209
<b>8. Análisis de secuencias biológicas</b>	<b>211</b>
8.1. Introducción . . . . .	211
8.1.1. Homología de secuencias . . . . .	211

8.1.2. Diferencias alineamiento DNA y proteínas . . . . .	213
8.2. Obtención de secuencias y formatos . . . . .	214
8.2.1. Bases de datos de secuencias de DNA y proteínas . . . . .	214
8.2.2. Formatos de archivos de secuencias . . . . .	214
8.2.3. Ejemplo lectura fichero de secuencia . . . . .	216
8.3. Alineamiento de secuencias . . . . .	216
8.3.1. Definición de similitud e identidad . . . . .	217
8.3.2. Matrices de sustitución . . . . .	217
8.3.3. Significación estadística y <i>E-value</i> . . . . .	221
8.3.4. El <i>twilight</i> u ‘ocaso’ de los alineamientos . . . . .	222
8.3.5. Técnicas y programas de alineamiento . . . . .	223
8.3.6. Alineamiento local de pares de secuencias . . . . .	224
8.3.7. Alineamiento global de pares de secuencias . . . . .	225
8.3.8. Alineamiento múltiple de secuencias . . . . .	226
8.3.9. Edición y visualización de alineamientos . . . . .	227
8.4. Bibliografía . . . . .	229
<b>9. Filogenia y evolución molecular</b>	<b>231</b>
9.1. Introducción . . . . .	231
9.1.1. Teoría de la Evolución . . . . .	232
9.1.2. Evolución molecular . . . . .	232
9.1.3. Interpretación de un árbol . . . . .	233
9.2. Métodos de reconstrucción filogenética . . . . .	236
9.2.1. Alineamiento múltiple . . . . .	237
9.2.2. Modelos de evolución . . . . .	237
9.2.3. Métodos de distancias . . . . .	239
9.2.4. Métodos probabilísticos . . . . .	241
9.3. Contrastos de hipótesis . . . . .	247
9.3.1. Contrastos entre modelos evolutivos . . . . .	247
9.3.2. Robustez y contraste de árboles . . . . .	249
9.4. Reconstrucción de estados ancestrales . . . . .	252
9.5. Guía rápida de reconstrucción filogenética . . . . .	252
9.6. Programas recomendados . . . . .	254
9.7. Bibliografía . . . . .	257
<b>III Biología estructural de proteínas</b>	<b>261</b>
<b>10. Alineamiento de estructura de proteínas</b>	<b>263</b>
10.1. Introducción . . . . .	263
10.2. Descripción general del método . . . . .	265
10.3. Comparaciones locales . . . . .	266
10.4. Construcción del alineamiento . . . . .	268
10.5. Medidas de similitud . . . . .	272
10.5.1. Medidas crudas y normalizaciones . . . . .	272
10.5.2. Una medida con motivación evolutiva . . . . .	274
10.6. Alineamiento múltiple . . . . .	275
10.6.1. Primeros pasos . . . . .	276
10.6.2. Construcción del alineamiento . . . . .	276

10.7. Discusión . . . . .	279
10.8. Bibliografía . . . . .	281
<b>11. Modelos simplificados de plegamiento de proteínas</b>	<b>283</b>
11.1. Introducción . . . . .	283
11.2. Conceptos básicos . . . . .	285
11.2.1. Paradoja de Levinthal y experimento de Anfinsen . . . . .	285
11.2.2. Paisaje energético y principio de mínima frustración . . . . .	286
11.3. Fundamentos teóricos . . . . .	289
11.3.1. paisaje energético del modelo de energía al azar . . . . .	289
11.3.2. Un paisaje energético un poco más realista . . . . .	292
11.3.3. Cinética del plegamiento y <i>gap</i> de estabilidad . . . . .	294
11.3.4. Diagrama de fases . . . . .	296
11.4. Algunos ejemplos computacionales sencillos . . . . .	299
11.4.1. Modelos de grano grueso. . . . .	299
11.4.2. La transición vítreo . . . . .	300
11.4.3. Diseñabilidad . . . . .	303
11.4.4. Plegamiento de una proteína simplificada . . . . .	305
11.5. Discusión . . . . .	308
11.6. Bibliografía . . . . .	311
<b>12. Evolución de estructura de proteínas</b>	<b>313</b>
12.1. Introducción . . . . .	313
12.2. Origen de nuevas proteínas . . . . .	313
12.3. Divergencia estructural gradual . . . . .	316
12.3.1. Clasificación de estructura de proteínas . . . . .	318
12.3.2. Cuantificando la divergencia estructural . . . . .	322
12.4. Evolución estructural mediante ensamblaje de módulos . . . . .	323
12.4.1. Péptidos ancestrales . . . . .	323
12.4.2. Búsqueda por recurrencia . . . . .	324
12.5. Divergencia estructural Vs. ensamblaje de módulos . . . . .	326
12.6. Bibliografía . . . . .	331
<b>13. Proteínas desordenadas</b>	<b>333</b>
13.1. Introducción . . . . .	333
13.2. Desorden en proteínas . . . . .	335
13.3. Predicción de desorden . . . . .	336
13.3.1. Métodos bioinformáticos . . . . .	337
13.4. Composición, distribución y función . . . . .	342
13.5. Enfermedades asociadas a proteínas desordenadas . . . . .	344
13.6. Bibliografía . . . . .	347
<b>IV Biología estructural de ácidos nucleicos</b>	<b>349</b>
<b>14. Estructura, plegamiento y evolución del RNA</b>	<b>351</b>
14.1. Introducción . . . . .	351
14.2. Tipos de RNA . . . . .	351
14.3. Niveles estructurales del RNA . . . . .	353
14.3.1. Composición química y estructura primaria . . . . .	353

14.3.2. Estructura secundaria . . . . .	354
14.3.3. Estructura terciaria . . . . .	356
14.3.4. Ejemplo estructura tRNA . . . . .	358
14.3.5. Arquitectura del RNA . . . . .	358
14.4. Plegamiento de RNA . . . . .	360
14.4.1. Aspectos generales del plegamiento de RNA . . . . .	360
14.4.2. Predicción de estructuras de RNA . . . . .	362
14.4.3. Otros problemas relacionados con el plegamiento . . . . .	368
14.4.4. Biología Sintética de RNA: plegamiento inverso . . . . .	370
14.5. Evolución de RNA . . . . .	371
14.5.1. El RNA como modelo evolutivo . . . . .	371
14.5.2. Redes neutrales de RNA . . . . .	372
14.5.3. Modelización matemática de la evolución . . . . .	374
14.5.4. Limitaciones y líneas futuras . . . . .	377
14.5.5. Evolución dirigida a una estructura objetivo . . . . .	379
14.6. Bibliografía . . . . .	383
<b>15. Estructura y organización del DNA</b>	<b>387</b>
15.0.1. Niveles de organización del DNA en la cromatina . . . . .	387
15.0.2. Determinación de la estructura de dominios genómicos . . . . .	388
15.0.3. Determinación de estructura con IMP . . . . .	390
15.0.4. Ejemplo de determinación de estructura por IMP . . . . .	394
15.0.5. Conclusiones . . . . .	396
15.1. Bibliografía . . . . .	397
<b>V Dinámica estructural y diseño de fármacos</b>	<b>399</b>
<b>16. Diseño de fármacos asistido por ordenador</b>	<b>401</b>
16.1. Introducción . . . . .	401
16.2. Docking proteína-ligando . . . . .	402
16.2.1. Definición del problema del <i>docking</i> . . . . .	402
16.2.2. Componente estructural . . . . .	403
16.2.3. Componente energética . . . . .	403
16.2.4. <i>Docking</i> : consideraciones teóricas . . . . .	406
16.2.5. El proceso de docking . . . . .	409
16.2.6. Evaluación de los resultados . . . . .	410
16.3. Cribado virtual . . . . .	412
16.3.1. Posibles escenarios para el VS . . . . .	413
16.3.2. Estudios de VS retrospectivos y prospectivos . . . . .	414
16.3.3. Herramientas de virtual screening . . . . .	415
16.4. Docking proteína-proteína . . . . .	416
16.4.1. El proceso de docking . . . . .	417
16.4.2. Clasificación y post-procesado de las soluciones . . . . .	419
16.5. Docking proteína-ácido nucleico . . . . .	420
16.6. Conclusiones y perspectivas . . . . .	420
16.7. Bibliografía . . . . .	423
<b>17. Dinámica molecular</b>	<b>425</b>
17.1. Introducción . . . . .	425

17.2. Mecánica molecular . . . . .	426
17.2.1. El campo de fuerzas . . . . .	426
17.2.2. Términos enlazados . . . . .	427
17.2.3. Términos no enlazados . . . . .	428
17.2.4. Parametrización del campo de fuerzas . . . . .	430
17.2.5. Minimización de energía . . . . .	430
17.3. Simulaciones de dinámica molecular . . . . .	431
17.3.1. Cálculo de las fuerzas . . . . .	432
17.3.2. Integración de las ecuaciones de movimiento . . . . .	432
17.3.3. Condiciones de límite periódico . . . . .	433
17.3.4. Cálculo de las interacciones no enlazantes . . . . .	433
17.3.5. Preparación y ejecución de una DM . . . . .	434
17.3.6. Simulaciones de macromoléculas biológicas . . . . .	435
17.4. Métodos híbridos QM/MM . . . . .	438
17.5. Programas y tutoriales . . . . .	439
17.6. Bibliografía . . . . .	441
<b>18. Análisis de modos normales</b>	<b>443</b>
18.1. Introducción al análisis de modos normales . . . . .	443
18.1.1. El oscilador armónico simple . . . . .	443
18.1.2. Modos normales en espacio cartesiano . . . . .	445
18.1.3. Modos normales en espacio diedro . . . . .	448
18.2. Modelos de redes elásticas . . . . .	449
18.2.1. El modelo de red Gaussiana o GNM . . . . .	450
18.2.2. El modelo de red Anisotrópica o ANM . . . . .	451
18.3. La molécula triatómica lineal . . . . .	453
18.4. Ejemplos prácticos . . . . .	456
18.4.1. Introducción a ProDy . . . . .	456
18.4.2. Cálculo de modos normales . . . . .	456
18.4.3. Cálculo de los factores B . . . . .	458
18.4.4. Cálculo de una estructura deformada . . . . .	462
18.5. Bibliografía . . . . .	465
<b>VI Biología de sistemas</b>	<b>467</b>
<b>19. Biología de sistemas</b>	<b>469</b>
19.1. Introducción . . . . .	469
19.2. Redes complejas . . . . .	471
19.2.1. Definición de red compleja y conceptos básicos . . . . .	471
19.2.2. Propiedades de las redes complejas . . . . .	472
19.2.3. Breve descripción de las redes biológicas . . . . .	477
19.3. Redes de regulación . . . . .	478
19.3.1. Modelos lógicos o booleanos . . . . .	478
19.3.2. Modelos cinéticos . . . . .	480
19.3.3. Modelos termodinámicos . . . . .	486
19.3.4. Regulación combinatoria . . . . .	486
19.3.5. Análisis de motivos de red . . . . .	488
19.3.6. Ruido en expresión genética . . . . .	494

19.3.7. Modelos cinéticos estocásticos . . . . .	496
19.3.8. Diferentes fuentes de ruido en expresión genética . . . . .	499
19.4. Redes metabólicas . . . . .	502
19.5. Robustez en los sistemas biológicos . . . . .	506
19.5.1. Redundancia por duplicación génica . . . . .	507
19.5.2. Robustez distribuida en el metabolismo . . . . .	508
19.6. Lecturas adicionales . . . . .	512
19.7. Herramientas computacionales . . . . .	513
19.8. Ejercicios . . . . .	514
19.9. Bibliografía . . . . .	517



## **Parte I**

# **Fundamentos de bioinformática, estadística y gestión de información científica**



# Capítulo 1

## Bases de datos biomédicas, servidores web y otros recursos online

*Natalia Jiménez Lozano*

### 1.1. Introducción

¿Has consultado alguna vez PubMed para descargar un artículo? Las bases de datos se encuentran totalmente integradas en nuestro día a día. La mayoría de los recursos que utilizamos tienen detrás una base de datos. ¡Y lo mejor de todo es que no hace falta ser un experto en bases de datos para consultarlas! Las bases de datos de las que hablaremos en este capítulo disponen de interfaces gráficas que facilitan el proceso de consulta, de manera que cualquiera puede ser capaz de recuperar la información requerida.

¿Has oido hablar de las bases de datos OMIM, ENSEMBL, UniProtKB o GenBank? Las bases de datos bioinformáticas constituyen una herramienta clave para los investigadores e incluso para los clínicos en su trabajo. En este capítulo vamos a aprender algunos conceptos relacionados con las bases de datos en Bioinformática.

#### 1.1.1. ¿Qué es un dato biológico?

Se podría definir *dato* como el resultado de una medición. En este capítulo hablaremos de bases de datos (BD) que albergan información biológica por lo que el *dato biológico* será un tipo particular de dato generado dentro del contexto de una investigación científica. Las mediciones que se realizan en el campo de la investigación no son simples ya que en la mayoría de los casos son el resultado de un complejo flujo de trabajo en el laboratorio donde se utilizan muy diversas técnicas. Podemos poner como ejemplo de dato biológico la secuencia de nucleótidos de un gen determinada mediante técnicas de secuenciación o la banda correspondiente a un fragmento de DNA separado mediante una electroforesis en gel de agarosa.

#### 1.1.2. Diferencia entre dato crudo y dato procesado

En la mayoría de los casos es necesario realizar algún tipo de procesamiento sobre los datos para facilitar la interpretación de los mismos. Un *dato crudo* es un dato obtenido directamente de la técnica

experimental. El procesamiento de los datos normalmente se hace por etapas, de manera que el *dato procesado* resultante de una etapa es el dato crudo de la siguiente.

Ejemplo: *Microarrays* de DNA. Como resultado directo de la técnica experimental, se obtienen unas imágenes de puntos cuyos colores representan las intensidades de las señales (datos crudos). Una vez obtenido los archivos de imágenes, es necesario transformar las intensidades de las señales obtenidas en datos numéricos. De esta manera se obtiene una matriz de números que representa en cada entrada la expresión de cada gen (primer dato procesado). Utilizando esta matriz como dato crudo podemos utilizar cualquier método de agrupamiento para agrupar los genes según la similitud de su perfil de expresión (segundo dato procesado).

### 1.1.3. Dato, información y conocimiento

Existe mucha confusión entre los términos dato, información y conocimiento. De hecho en muchas ocasiones estos términos son considerados erróneamente como sinónimos. La diferencia fundamental radica en que, mientras que a partir de conjuntos de datos se puede derivar la *información* directamente, el *conocimiento* normalmente se deriva de forma indirecta [12, 13]. Utilizaremos un ejemplo para poner de manifiesto la diferencia que existe entre estos tres conceptos.

Ejemplo: Imagina que regentas un herbolario y que tienes una base de datos donde registras todos los datos de tus clientes de manera que conoces sus nombres y los productos que compran en tu establecimiento. Que los clientes Paula y Javier compren leche sin lactosa cada lunes es un dato que tú tienes almacenado en tu base de datos. Cada vez que quieras saber quiénes son los clientes que compran leche sin lactosa o cuántos litros de leche sin lactosa vendes cada día, consultarás a la base de datos y tendrás el resultado. Esto es información. Ahora imagina que hay otros 100 clientes que también compran leche sin lactosa y que todos ellos son alérgicos a la lactosa. Entonces podrás concluir que Paula y Javier deben ser alérgicos a la lactosa también. La alergia de Paula y Javier no se te ha proporcionado como dato y tampoco se puede extraer de la base de datos como información. Sin embargo tú has extrapolado esta información de manera indirecta y a esto es a lo que llamamos conocimiento.

Por lo tanto el dato es objetivo y no abstracto y sin embargo la información y el conocimiento son subjetivos y requieren altos grados de abstracción. La organización de los datos biológicos en bases de datos facilita el descubrimiento de conocimiento ya que permite poner de manifiesto relaciones entre piezas de información que se desconocían en el momento en que la información fue introducida por primera vez (datos crudos o sin procesar). Otro ejemplo de generación de conocimiento sería el derivar los motivos conservados en un conjunto de secuencias proteicas crudas pertenecientes a una base de datos.

### 1.1.4. Características de los datos biológicos

Ahora que sabemos lo que es un dato biológico y lo distinguimos de la información y del conocimiento, veamos cuáles son sus características [4].

- Los datos biológicos son *heterogéneos* porque representan entidades diversas que van desde átomos hasta estudios poblacionales, pasando por secuencias de nucleótidos y proteínas, estructuras proteicas cristalinas, medidas de expresión génica, interacciones proteína-proteína o proteína-DNA, redes e interacción, células, estudios fenotípicos y estudios fisiológicos.

- Los datos biológicos son *complejos*. Para que te hagas una idea de la complejidad de los datos biológicos vamos a compararlos con los datos antropométricos. Imagina la diferencia que hay entre los datos correspondientes al peso de una persona y la determinación una estructura de una proteína sencilla como la insulina. En el primer caso lo único que tendríamos que hacer sería pesar al individuo con una báscula. Sin embargo en el segundo caso tendríamos que purificar la proteína, cristalizarla, obtener el difractograma y a partir de éste determinar las posiciones en el eje X, Y, Z de cada átomo de cada uno de los 110 aminoácidos de la proteína. El dato antropométrico es un número y el dato biológico es un fichero de más de mil líneas, 865 de las cuales corresponden a la posición de uno de los átomos de la estructura.
- Los datos biológicos pueden tener una *naturaleza cuantitativa* (Ej. peso molecular de una proteína) o *cualitativa* (Ej. función de una proteína).
- Los datos biológicos son necesariamente *dinámicos* porque van cambiando según van evolucionando las técnicas que los generan o van surgiendo nuevas técnicas que completan el dato. Ej. la primera secuencia proteica que se determinó, la insulina bovina, depositada en la base de datos UniProtKB en el año 1986 (Identificador P01317; hablaremos de esta base de datos en detalle más adelante) ha sufrido hasta la fecha 126 revisiones. Este dato, como el resto de datos biológicos es incompleto ya que seguirá evolucionando indefinidamente.
- Los datos biológicos pueden proceder de interpretaciones, de análisis computacionales o bien pueden ser datos confirmados experimentalmente. Veamos algunos ejemplos:
  - Datos *procedentes de interpretaciones*: la descripción del nivel de expresión de un gen en un tejido determinado por la técnica de hibridación *in situ*. Mediante esta técnica, el experimentalista obtiene una imagen correspondiente a una sección del organismo que esté estudiando, y tiene que determinar si la expresión en los tejidos de interés es inexistente, débil, media, fuerte o muy fuerte. Por lo tanto en este caso el experimentalista ha de interpretar el dato crudo (imagen obtenida de la técnica).
  - Datos *procedentes de análisis computacionales*: estructura secundaria de una proteína es un ejemplo de dato obtenido computacionalmente a partir de la secuencia. En este caso el dato biológico está asociado a una probabilidad. Existe un gran abanico de aplicaciones que proporcionan este tipo de datos.
  - Datos *confirmados experimentalmente*: la interacción entre dos proteínas observada mediante el experimento del doble híbrido.

### 1.1.5. Tipos de datos

Los centros de secuenciación a gran escala, las instalaciones de análisis de alto rendimiento y los laboratorios individuales producen una extensa cantidad de datos como secuencias de nucleótidos y proteínas, estructuras proteicas cristalinas, medidas de expresión génica, interacciones proteína-gen y estudios fenotípicos.

¿Cómo almacenamos los datos biológicos en el ordenador? Como se ha comentado anteriormente, los datos biológicos son complejos y heterogéneos por lo que no existe una forma común de representarlos. Haciendo un esfuerzo de síntesis podríamos hablar de *seis tipos de datos biológicos* y por lo tanto seis formas diferentes de representación [4]. Dentro de cada forma de representación existen matices que dan lugar a los diferentes tipos de formatos que discutiremos más adelante en este capítulo.

El dato más simple a representar en bioinformática es el *dato escalar*. Un ejemplo de este tipo de dato

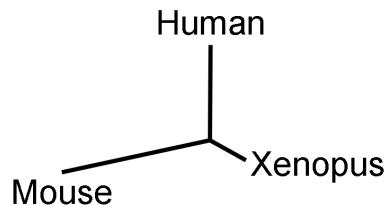
sería el punto isoeléctrico de una proteína. Las secuencias tanto de proteínas como de ácidos nucleicos se almacenan como *cadenas de caracteres* donde cada carácter es un amino ácido o un nucleótido (Figura 1.1).

```
MALWMRLPLLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTR  
REAEDLQVGQVELGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
```

**Figura 1.1:** Secuencia correspondiente a la insulina humana.

La representación se complica cuando se trata de estructuras atómicas ya que para describir la organización de una proteína en el espacio es necesario detallar la posición de cada uno de sus átomos en las coordenadas X, Y y Z. Por lo tanto el fichero con las coordenadas atómicas de una proteína es un *dato espacial*. Existen determinadas técnicas como la hibridación *in situ* que nos permiten medir el nivel de expresión de un gen en un órgano o tejido durante el desarrollo de un organismo (*dato temporal*).

Los *grafos* se utilizan para representar árboles filogenéticos de los cuales se hablará en detalle en el Capítulo 9 (Figura 1.2).



**Figura 1.2:** Árbol filogenético.

La *representación vectorial* es muy útil cuando se trata de datos resultantes de experimentos de dinámica molecular y celular.

#### 1.1.6. Ciclo de vida de los datos biológicos

El dato crudo se genera en el contexto de un proyecto de investigación a través de un experimento determinado. Este dato normalmente se deposita en una base de datos. Existen dos tipos de perfil de usuario de las bases de datos bioinformáticas: aquellos interesados en el dato tal y como está almacenado en la base de datos y los que utilizan el dato crudo para derivar información y conocimiento a través de diferentes tipos de análisis (Figura 1.3).



**Figura 1.3:** Ciclo de vida del dato biológico.

## 1.2. Historia de las bases de datos

Si hay un nombre ligado a la Bioinformática es sin lugar a dudas el de Margaret Dayhoff (1925-1983). Esta fisicoquímica estadounidense fue pionera en el campo de la Bioinformática. Su visión de futuro le llevó a publicar un libro en 1965 que reunía las 65 secuencias proteicas existentes hasta esa fecha con el objetivo de facilitar el análisis de las mismas. Este libro titulado “Atlas of Protein Sequence and Structure” fue el precursor de PIR (Protein Information Resource), la primera base de datos en el campo de la Bioinformática.

### 1.2.1. ¿Por qué son esenciales las bases de datos?

Durante las dos últimas décadas se han producido una gran variedad de desarrollos tecnológicos que han acelerado el ritmo de producción de datos biológicos. La inversión en tecnologías de alto rendimiento, comenzando por los análisis de expresión de *microarrays* a mitad de los años 90 y continuando con el conocimiento cada vez más detallado del genoma, transcriptoma, proteoma y metaboloma, ha dado lugar a un tremendo escalado en el ritmo de producción de datos biológicos crudos. Según Burge et al [3] si se continúa el ritmo de generación de datos que se tiene hoy en día, en el 2020 habrá un millón de veces más datos que actualmente. Por lo tanto el próximo reto en la investigación es hacer el análisis, la gestión y el acceso a los datos tan eficiente como la generación de los mismos [10].

Las *bases de datos* surgen por tanto de manera natural ante la necesidad de preservar y organizar la avalancha de datos que se estaba generando solucionando así el problema del archivo de los datos pero al mismo tiempo permitiendo el análisis de los mismos y su reutilización para otros propósitos. Los científicos dependen de la disponibilidad de los datos de otros científicos por lo que el dato biológico no tiene interés por sí mismo sino que adquiere su valor en la medida en que la comunidad científica sea capaz de localizarlo, integrarlo y accederlo. Los primeros esfuerzos destinados a la creación de bases de datos bioinformáticas los realizaron grupos de investigación interesados en organizar y compartir los datos generados en su propio laboratorio. Conforme las bases de datos iban creciendo, el perfil de gestor de las mismas se fue profesionalizando, pasando a ser cada vez más computacional. A partir de ese momento surgieron *proyectos internacionales* que se hicieron cargo de estas bases de datos, dos de estas iniciativas son:

- European Bioinformatics Institute (EBI).

- National Centre for Biotechnology Information (NCBI).

El EBI forma parte del EMBL (European Molecular Biology Laboratory) y se construyó en el año 1992 en el campus Wellcome Trust Genome en Hinxton (Inglaterra) para dar soporte a la gran cantidad de datos que se estaban generando con los proyectos de secuenciación del Instituto Sanger. Actualmente además de otras muchas bases de datos, el EBI alberga ENA (*European Nucleotide Archive*) para secuencias de nucleótidos y UniprotKB de secuencias de proteínas [2, 11].

El NCBI es parte de la Biblioteca Nacional de Medicina de los Estados Unidos (NLM) que es a su vez parte del Instituto Nacional de Salud (NIH). El NCBI está en Maryland y se fundó en 1988 para desarrollar sistemas de información en el campo de la biología molecular. Como bases de datos más relevantes tiene la de secuencias de nucleótidos *GenBank* y la de bibliografía biomédica *PubMed*. Estas bases de datos junto con muchas otras más, se pueden consultar a través del motor de búsqueda *Entrez* [17].

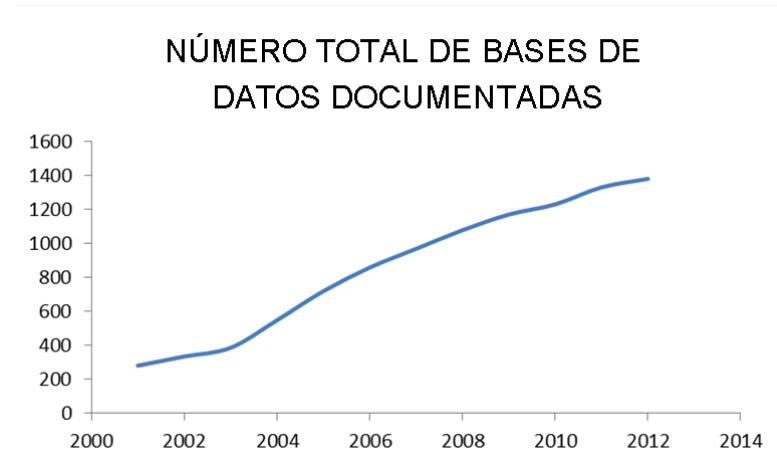
Las *bases de datos online* se han convertido en importantes vías para publicar los datos biológicos. De hecho, es requisito imprescindible para la publicación en determinadas revistas, el archivo en la correspondiente base de datos. Como consecuencia de esto cada vez es más común encontrar en las publicaciones un apartado dentro de la sección de métodos que haga referencia a un identificador de secuencia, estructura tridimensional, gel bidimensional, etc.

Las bases de datos comienzan a ser la piedra angular de la investigación biomédica moderna, citándose en la literatura miles de veces al año. Actualmente no se concibe ningún flujo de trabajo en investigación que no suponga en alguno de sus pasos la “consulta a” o la “integración con” alguna base de datos bioinformática. Por esta razón me gustaría hacer hincapié en la *accesibilidad de los datos*: una base de datos bioinformática, para que sea realmente de utilidad, ha de estar públicamente accesible *online* o ha de poder descargarse en su totalidad.

### 1.2.2. Diversidad de las bases de datos

Es difícil de averiguar el número exacto de bases de datos. Aunque no es exhaustivo, el número especial online de la revista *Nucleic Acids Research (NAR)* sobre bases de datos de 2012 listó en 1380 las bases de datos publicadas [8] y estimaciones derivadas de un estudio llevado a cabo en el contexto del proyecto ELIXIR sugirió un ritmo crecimiento anual aproximado de un 12 % (Figura 1.4). Globalmente parecería que existen más bases de datos que las mencionadas en la colección online, y esto se debe a que muchas bases de datos no se han publicado, o no se han publicado en el número de *NAR* dedicado a bases de datos. *NAR* clasifica las bases de datos de manera jerárquica en 14 categorías y 41 subcategorías. La existencia de muchas bases de datos en áreas estrechamente relacionadas hace útil la inclusión de referencias cruzadas entre entradas relacionadas en bases de datos diferentes (ver Sección 1.4). Dado el alto número de bases de datos, en este capítulo trataremos únicamente las más emblemáticas.

Los laboratorios de investigación y las técnicas experimentales dependen de recursos como las bases de datos y del *software* bioinformático que proporcionan herramientas integradas para facilitar su flujo de trabajo en investigación.



**Figura 1.4:** Gráfico crecimiento bases de datos según el número especial de bases de datos de la revista Nucleic Acids Research.

### 1.3. Estructura de las bases de datos

Hemos visto anteriormente que una BD es un conjunto de datos biológicos pertenecientes a un mismo contexto que tiene una estructura específica que permite añadir, eliminar y extraer datos y en muchos casos también ayuda al análisis de los mismos. Desde el punto de vista estructural, el objetivo principal del desarrollo de una BD es organizar los datos en un conjunto de *registros* estructurados que permitan recuperar fácilmente la información. Cada registro está compuesto por un número determinado de *campos* que contienen datos específicos, por ejemplo: DNI, direcciones de correo electrónico, etc. El proceso de *consulta* consiste en recuperar un registro concreto de la base de datos a través de la especificación de una pieza de información (valor) que será encontrada en un campo en especial (Figura 1.5). Para facilitar el acceso y recuperación de datos, se han desarrollado programas denominados *sistemas de gestión de bases de datos*.



**Figura 1.5:** Proceso de consulta en una base de datos.

Existen numerosas formas de organizar los datos en las BD pero en la actualidad, los tres tipos principales de BD utilizadas para almacenar datos biológicos son las BD constituidas por archivos planos, las bases de datos relacionales y las bases de datos orientadas a objetos.

Las *bases de datos de ficheros planos* almacenan los datos como ficheros de texto (Figura 1.6). Originalmente todas las bases de datos utilizaban este formato de almacenamiento donde los campos se encuentran separados por algún tipo de carácter especial (tabuladores, comas...). El inconveniente obvio que tiene este tipo de BD es que las consultas se hacen ineficientes ya que es necesario rastrear el fichero de principio a fin para encontrar la información solicitada por el usuario. Pero a pesar de los inconvenientes, muchas BD biológicas siguen utilizando este sistema ya que no implica mucho diseño y los resultados de la búsqueda son fácilmente entendibles y manipulables sin necesidad de herramientas especializadas.

```
ID      INS_BOVIN          Reviewed;      105 AA.
AC      P01317;
DT      21-JUL-1986, integrated into UniProtKB/Swiss-Prot.
DT      01-AUG-1992, sequence version 2.
DT      18-APR-2012, entry version 127.
DE      RecName: Full=Insulin;
DE      Contains:
DE          RecName: Full=Insulin B chain;
DE      Contains:
DE          RecName: Full=Insulin A chain;
DE      Flags: Precursor;
GN      Name=INS;
OS      Bos taurus (Bovine).
```

**Figura 1.6:** Fichero de texto correspondiente a la insulina bovina procedente de UniProt (número de acceso P01317).

El tipo más común de base de datos usado para almacenar información biológica es la *base de datos relacional* donde los datos se organizan en *tablas* consistentes en *registros* y *campos*. Estas tablas se relacionan a través de campos especiales llamados *claves*. En una *base de datos orientada a objetos*, la información se representa mediante objetos como los presentes en la programación orientada a objetos.

## 1.4. Los datos de diferentes bases de datos están interrelacionados

Como hemos visto anteriormente, existe una gran cantidad de bases de datos, y muchas de ellas coexisten en la misma área, por lo que se hace necesaria la interoperabilidad de las mismas a través de las referencias cruzadas. Las *referencias cruzadas* no son más que enlaces entre las bases de datos que permiten al usuario ir de una BD a otra haciendo una única consulta (Figura 1.7). Si bien el concepto es de referencia cruzada es muy sencillo de entender, el proceso por el cuál un dato de una BD se relaciona con otro dato perteneciente otra BD puede llegar a ser una tarea bastante compleja llevada a cabo por profesionales altamente cualificados llamados revisores de los datos biológicos.



**Figura 1.7:** Referencias cruzadas: a partir de la entrada correspondiente a una secuencia de UniProt se llega a la estructura resuelta por rayos X (PDB) y de ahí a la publicación donde se describe la estructura (PubMed).

## 1.5. Calidad y revisión de los datos

Según Burge et al. [3], la *revisión* consiste en “el análisis, interpretación e integración de la información biológica que se encuentra almacenada en las BD para añadir valor mediante la anotación y la interconexión de los datos dentro de un marco biológico común”. La calidad de los datos depende totalmente de la supervisión de los mismos. Las *bases de datos supervisadas* son la piedra angular de la investigación en ciencias de la vida ya que proporcionan información esencial de referencia sobre genes, sus productos e interacciones, estándares de nomenclatura, secuencias genómicas de referencia y sus anotaciones.

### 1.5.1. Anotación de los datos

La definición de *anotación* según la Wikipedia es: “información extra insertada en un punto específico dentro de un documento o cualquier otra pieza de información”. En nuestro contexto, la pieza de información es el dato crudo y la anotación es el procesamiento de estos datos crudos con la finalidad de extraer información adicional. Así, los datos derivan directamente de los experimentos y las anotaciones de las interpretaciones realizadas por los revisores. Emplearemos UniProtKB para ilustrar este concepto. Esta BD está centrada en las secuencias de las proteínas (dato crudo) pero añade valor a cada registro (anotaciones) mediante la inclusión de información relacionada con el papel de la proteína (función, estructura, localización subcelular, interacciones con otras proteínas y dominios que la componen) así como un amplio rango de características relativas a la secuencia como sitios activos y modificaciones postraduccionales.

La *anotación manual* es un proceso lento y complicado. Hoy en día los revisores se encuentran desbordados por la cantidad de datos que se incorporan a las bases de datos. Para lidiar con esta cantidad de datos se han creado herramientas para la automatización del proceso de anotación (*anotación automática*).

### 1.5.2. Vocabularios controlados y ontologías

Los revisores no solamente anotan sino que también son los encargados de gestionar la “avalancha de datos” controlando la incorporación y la estandarización de los mismos con el fin de facilitar el proceso de integración entre datos pertenecientes a diversas bases de datos. En esta conexión coherente juegan un papel fundamental los vocabularios controlados y las ontologías ya que proporcionan reglamentaciones terminológicas que apoyan la reusabilidad y la integración de los datos y, por lo tanto, favorecen el desarrollo de sistemas útiles para el proceso de anotación, la extracción de información o el procesamiento del lenguaje natural.

Los *vocabularios controlados* (VC) ofrecen un conjunto de términos formales dentro de un campo determinado de aplicación. Una *ontología* es un tipo de vocabulario controlado donde los términos se encuentran definidos y existen relaciones entre los mismos. La interpretación de estos términos puede ser procesada por un ordenador o interpretada por una persona. Los VC y las ontologías disponibles a la comunidad científica proporcionan un entorno dentro del cuál los grandes conjuntos de datos pueden ser anotados y consultados sistemáticamente. Estas herramientas aseguran la consistencia e incrementan la eficiencia y la precisión de las consultas mediante la estandarización de las amplias variaciones terminológicas que existen en un campo específico de estudio. Podemos ver tres ejemplos de diferentes ontologías en la Figura 1.8.

Actualmente existen muchos vocabularios y ontologías reconocidas como estándares por la comunidad científica que cubren un amplio abanico de campos de conocimiento. Incluso en el año 2004 se creó el National Centre for Biomedical Ontologies, un consorcio dependiente del NIH (National Institute of Health) para impulsar el desarrollo y uso de las ontologías en el campo de la Biomedicina. Este consorcio da acceso a BioPortal<sup>1</sup>, un sitio web donde se pueden encontrar más de 300 ontologías relacionadas con las ciencias de la vida [6, 14]. Sin embargo estas herramientas aún no están presentes en el proceso de escritura de un artículo científico: un autor cuando describe genes, proteínas, fenotipos... etcétera, utiliza normalmente sus propios términos sin ser consciente de que la utilización de una terminología estándar sería de gran utilidad para la posterior inclusión sus datos en una base de datos.

La incorporación de los resultados de una investigación en una base de datos se puede hacer de dos maneras. Imaginemos por ejemplo el caso de un investigador que descubre un nuevo gen implicado en la resistencia a metales tóxicos en microrganismos extremófilos. Tendríamos estos dos escenarios:

1. El investigador conoce la base de datos GenBank y está interesado en incorporar este gen. En este caso el autor accederá a una interfaz web que le permitirá integrar su dato en esta BD a través de una serie de pasos sencillos.
2. El investigador no conoce la existencia de la base de datos GenBank ni de ninguna otra base de datos biológica. Nuestro investigador se centraría por tanto únicamente en la redacción del artículo que describe la nueva función de este gen. Una vez publicado, los revisores de la base de datos GenBank, realizarán el proceso de incorporación de los datos en la BD, mediante la extracción de forma no ambigua de las entidades que se discuten en el artículo. Los revisores cotejarían la terminología utilizada en el artículo con los vocabularios/ontologías de la base de datos. Como se desprende de este punto, los revisores no sólo desarrollan y la gestionan los

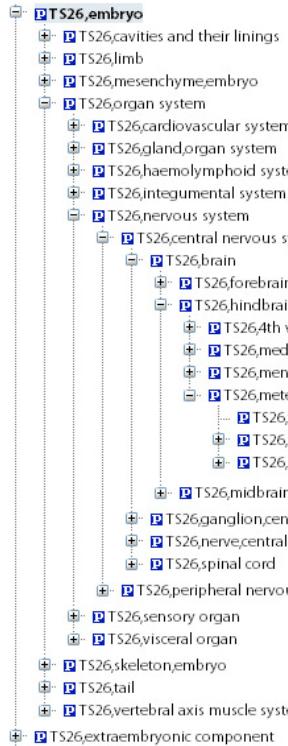
---

<sup>1</sup>BioPortal, repositorio de ontologías biomédicas. <http://bioportal.bioontology.org>

vocabularios y ontologías y sino que también las utilizan durante el proceso de incorporación de datos a una base de datos.

Como hemos visto, la integración de los datos en las BDs normalmente requiere de la colaboración entre dos actores: el autor de los datos y el revisor de la base de datos donde se integrarán estos resultados. Fruto de esta colaboración es el almacenamiento del dato biológico en una base de datos como parte del proceso de publicación y la inclusión del identificador correspondiente como referencia cruzada en el artículo.

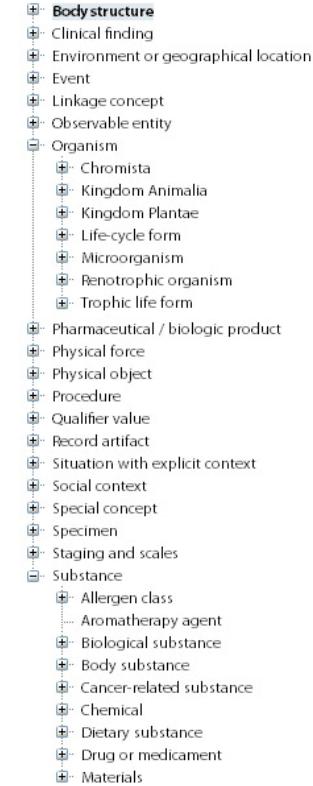
#### A) Ontología EMAP TS26



#### B) Ontología GO Proceso Biológico



#### C) Ontología SNOMED



**Figura 1.8:** Ontologías. A) Ontología que describe las estructuras anatómicas de ratón en cada uno de los estadios del desarrollo embrionario (en este ejemplo, TS26) creada dentro del proyecto e-Mouse Atlas Project; B) Ontología de Gene Ontology de procesos biológicos (serie de funciones moleculares reconocidas); C) SNOMED es una ontología de términos médicos que proporciona códigos, términos, sinónimos y definiciones de enfermedades, hallazgos, protocolos, microrganismos, sustancias, etc.

### 1.5.3. Revisores ¿quiénes son y qué hacen?

El proceso de revisión de los datos se ha convertido en una parte esencial del descubrimiento biológico que permite organizar, representar y hacer la información biológica accesible tanto a los humanos como a los computadores.

A lo largo de esta sección hemos visto cuatro de las *tareas de los revisores*: anotación, desarrollo de

vocabularios y ontologías, control en el proceso de incorporación de datos y extracción de información de la literatura. Además, los revisores se encargan de corregir las inconsistencias y errores en la representación de los datos.

La revisión requiere una mezcla de habilidades y experiencia que incluye un nivel de investigación científica avanzada, competencias en sistemas de gestión de bases de datos, sistemas operativos múltiples y lenguajes de *scripting*. Este tipo de formación se adquiere a través de la combinación de autoaprendizaje y experiencia.

## 1.6. Números de acceso

Un número de acceso identifica de manera única un registro en una base de datos. Cada base de datos tiene su propio formato para los números de acceso que incluso depende de las entidades a las que identifique como en el caso de GenBank:

Nucleótidos: 1 letra + 5 números ó 2 letras + 6 números.

Proteínas: 3 letras + 5 números.

Los números de acceso de UniProtKB consisten en 6 caracteres alfanuméricos en el siguiente formato:

1	2	3	4	5	6
[A-N,R-Z] [O,P,Q]	[0-9]	[A-Z] [A-Z, 0-9]	[A-Z, 0-9]	[A-Z, 0-9]	[0-9]

## 1.7. Gestión de las bases de datos

Para que una base de datos sea considerada de utilidad en el campo de la Bioinformática, ha de cumplir una serie de requisitos:

- *Accesibilidad*: es requisito indispensable que todos los datos estén disponibles a la comunidad científica de manera gratuita y sin restricción ya que el progreso de la investigación en ciencias de la vida dependen del acceso a estos datos. Por lo tanto, las bases de datos bioinformáticas deben ser accesibles online.
- Han de proporcionar conjuntos de *datos completos y actualizados*: para que una base de datos tenga realmente interés, ésta ha de actualizarse periódicamente para incluir nuevos datos y ha de proporcionar referencias a otras bases de datos (referencias cruzadas, ver Sección 1.4).
- *Interfaces de acceso intuitivas* y amigables que permitan el acceso y la incorporación de datos así como herramientas que permitan hacer análisis de los mismos.
- *Adoptar en lo posible estándares* en el campo de la bioinformática para favorecer el intercambio de datos entre diversas colecciones.
- Cuando sea posible, los conjuntos de datos proporcionados por las bases de datos, estarán *disponibles para su descarga*.
- *Calidad*: los datos biológicos almacenados en las bases de datos deberían estar revisados manual o automáticamente para asegurar la calidad y la extracción de anotaciones de los mismos.

Por lo tanto dejemos a un lado la visión que se tiene de las bases de datos como herramientas estadísticas. Las bases de datos requieren un gran esfuerzo a nivel diseño, implementación, mantenimiento, organización, anotación, supervisión, depósito y almacenamiento.

## 1.8. Tipos de bases de datos

Como hemos visto en secciones anteriores, existen multitud de bases de datos por lo que resulta útil clasificarlas en grupos. Se pueden hacer diferentes clasificaciones de las BD dependiendo de diferentes factores. En este capítulo vamos a estudiar las clasificaciones que atienden a la procedencia de los datos y a la revisión de los mismos. Luego hablaremos de una iniciativa encaminada a etiquetar las bases de datos de manera estándar (BioDBCore) [9] y por último veremos cómo las instituciones de referencia en bioinformática como la revista Nucleic Acids Research, el EBI o el NCBI hacen su propia clasificación de los datos.

### 1.8.1. Procedencia de los datos

En términos generales podríamos decir que las *bases de datos primarias* tienen datos originales que proceden directamente del experimento y normalmente son incorporadas por los autores a las bases de datos. Sin embargo las *bases de datos secundarias* contienen datos derivados del análisis de los datos de las bases de datos primarias. Estos análisis se hacen computacionalmente o manualmente por un revisor.

Bases de datos primarias: GenBank (secuencias nucleotídicas) y PDB (estructuras procedentes de las técnicas de rayos X y de Resonancia Magnética nuclear).

Bases de datos secundarias: UniProt/SwissProt y PROSITE.

Las *bases de datos especializadas* están centradas en un campo de investigación concreto como por ejemplo ZFIN (pez cebra), MGI (ratón), TRANSFAC (factores de transcripción).

### 1.8.2. Revisión de los datos

Las *bases de datos revisadas* están sujetas a un control de la calidad de los datos por parte de los revisores que se encargan de controlar el proceso de incorporación de los datos, la actualización y la consistencia de los mismos. Un buen ejemplo de base de datos sometida a un exhaustivo sistema de revisión tanto manual como computacional es UniProtKB, de la que hablaremos en detalle más adelante.

Por otro lado las *bases de datos no revisadas* no cuentan con la figura de los revisores sino que son bases de datos donde los autores introducen libremente sus datos sin la guía de un experto ni de un vocabulario controlado u ontología. A las bases de datos no revisadas se las llama también *repositorios*. Los repositorios normalmente no incluyen información adicional al dato experimental y además suelen ser redundantes. Actualmente existen muy pocas bases de datos de este tipo.

### 1.8.3. Clasificación EBI

El European Bioinformatics Institute (EBI) clasifica las bases de datos en *bases de datos clave* y *bases de datos especializadas*. Las primeras proporcionan colecciones completas de valor genérico para las ciencias de la vida. Estas se dividen de forma grosera en dos categorías: aquellas que describen los componentes moleculares de los sistemas biológicos (por ejemplo secuencias de nucleótidos, secuencias de proteínas, estructuras macromoleculares y moléculas pequeñas) y las que describen sus comportamientos o los resultados de estos comportamientos (por ejemplo transcripción, traducción e interacción). Como ejemplos de la primera tendríamos Ensembl de genomas [7], ENA de secuencias nucleotídicas,

PDBe de estructuras proteicas o chEBI de entidades químicas. Sin embargo las bases de datos PRIDE (proteomas), Reactome (rutas), ArrayExpress (expresión génica) o IntAct (interacciones moleculares) formarían parte de la segunda categoría.

Además de las bases de datos clave, el EBI alberga un gran número de bases de datos especializadas como European Mutant Mouse Archive que proporciona información sobre las cepas mutantes de ratón para permitir a los investigadores enlazar la información fenotípica con las mutaciones o la Non-Redundant Patent-Sequence Database Collection que proporciona acceso a secuencias con aplicaciones patentadas.

#### **1.8.4. Clasificación NCBI**

El National Centre for Biotechnology Information (NCBI) clasifica las bases de datos en las siguientes categorías:

- Referencias bibliográficas (PubMed).
- Genes y secuencias asociadas (Entrez Gene, UniGene).
- Genomas (Entrez Genome, Sequence Read Archive).
- Genotipos y fenotipos (dbGaP, dbSNP, OMIM).
- Expresión génica (GEO, GENSAT).
- Estructura molecular y proteómica (MMDB).

#### **1.8.5. Clasificación NAR**

La revista Nucleic Acids Research publica desde el año 2001 un número anual dedicado a bases de datos que clasifica en 15 categorías:

- Secuencias de Nucleótidos
- Secuencias de RNA
- Secuencias de proteínas
- Estructura
- Bases de Datos Genómicas (invertebrados)
- Rutas Metabólicas y de Señalización
- Genomas de Humano y otros Vertebrados
- Genes humanos y Enfermedades
- Datos de Microarrays y otras bases de datos de Expresión
- Bases de datos de Proteómica
- Otras bases de datos de Biología Molecular
- Bases de datos de organelas
- Bases de datos de Plantas
- Bases de datos Inmunológicas
- Biología celular

#### **1.8.6. Redundancia**

En una base de datos podemos encontrar una misma entrada repetida varias veces. Esto puede deberse a que el autor del dato repita el experimento y lo actualice o bien a que varios autores realicen el mismo trabajo. Cuando una base de datos tiene datos idénticos, decimos que es redundante. GenBank

es un ejemplo de *base de datos redundante* ya que contiene entradas diferentes para cada proyecto de secuenciación de nucleótidos aunque esto suponga la inclusión de secuencias duplicadas del mismo gen obtenido por laboratorios diferentes. Los datos duplicados pueden ser coincidentes cuando varios laboratorios llegan al mismo resultado. Pero la mayoría de las veces los datos redundantes son discrepantes y pueden hacer que los resultados del análisis sea inconsistente. Además, la redundancia hace que la base de datos necesite más espacio de almacenamiento y afectan a la eficiencia de la consulta. Por estas razones los revisores tratan de eliminarla en lo posible.

Hasta ahora hemos hablado de la redundancia dentro de una base de datos (*redundancia intra-BD*). Pero sin embargo se pueden encontrar datos duplicados en diferentes bases de datos. En este caso hablaríamos de *redundancia inter-BD*. Podemos destacar dos *bases de datos no redundantes* que recogen información de diferentes archivos: RefSeq y UniParc.

### **RefSeq**

La base de datos RefSeq se construye a partir de las secuencias proporcionadas por las bases de datos de secuencias de International Nucleotide Sequence Database Collaboration (INSDC) que incluye el banco de DNA de Japón, el archivo europeo de nucleótidos (ENA) y GenBank. RefSeq integra información sobre genes, transcritos y proteínas de cada organismo junto con anotaciones e información bibliográfica [16].

### **UniParc**

UniParc es una base de datos no redundante que contiene la mayoría de las secuencias proteicas que hay en el mundo ya que recoge datos de las siguientes BDs: UniProtKB/Swiss-Prot, Uni-ProtKB/TrEMBL, PIR-PSD, EMBL, EMBL WGS, Ensembl, IPI, PDB, PIR-PSD, RefSeq, FlyBase, WormBase, H-Invitational Database, TROME databases, European Patent Office proteins, United States Patent and Trademark Office proteins y Japan Patent Office proteins. UniParc dispone únicamente de secuencias proteicas. Cualquier otro tipo de información sobre la proteína debe ser extraído directamente de las bases de datos de origen a través de referencias cruzadas.

#### **1.8.7. Acceso**

Podemos hablar de tres maneras de acceder a los datos proporcionados por una base de datos. El sistema más común de acceso a una base de datos es a través de su *interfaz web específica* (Figura 1.9A). Sin embargo, en los últimos años el crecimiento en el número de bases de datos originó la necesidad de unificar las interfaces de búsqueda para facilitar al usuario el proceso de consulta. Para ello se crearon los llamados entornos integrados como el EBI o el NCBI de los que hemos hablado en la Subsección 1.2.1. La incorporación de las bases de datos en entornos integrados llevó aparejado el desarrollo de motores de búsqueda. Así tenemos Entrez como motor de búsqueda del NCBI o el EB-Eye como motor de búsqueda del EBI (Sección 1.11). A través de los *motores de búsqueda*, el usuario puede obtener una panorámica de los resultados de un amplio conjunto de bases de datos pertenecientes a un repositorio haciendo una única consulta (Figura 1.9B).

Otro tipo de acceso, el *programático*, permite a una máquina y no una persona ejecutar la consulta y recibir del resultado de la misma (Figura 1.9C). El acceso programático a las bases de datos permite que el resultado de la consulta a una base de datos sea la consulta a la siguiente sin la intervención humana. De esta manera se puede automatizar las consultas construyendo los llamados flujos de trabajo.

**A) Interfaz de UniProtKB**

**B) Motor de Búsqueda EB-eye**

**C) Acceso a través de Servicio Web**

```

ID INS_HUMAN Reviewed: 110 AA.
AC P01308; QSEER2;
DT 21-JUL-1996; integrated into UniProtKB/Swiss-Prot.
DT 21-JUL-1996; sequence version 1.
DT 18-APR-2012; entry version 169.
DE RecName: Full=Insulin;
DE Contains:
DE RecName: Full=Insulin B chain;
DE Contains:
DE RecName: Full=Insulin A chain;
DE Flags: Precursor;
GN Name=INS;
CS Homo sapiens (Human).
OC Diketopiperazine Merozeta; Chordata; Craniata; Vertebrata; Euteleostomi;
OC Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
OC Catarrhini; Hominoidea; Homo.
OX NCBI_TaxID=9606;
RN [1]

```

**Figura 1.9:** A) Ejemplo de interfaz web específica correspondiente a la base de datos UniProt; B) Entorno integrado del EBI conocido como EB-eye; C) Acceso programático a una entrada de UniProt a través de los servicios web del EBI.

El sitio web del EBI es visitado por 300.000 usuarios aproximadamente cada mes y recibe diariamente 3,5 millones de peticiones. Además cerca de un millón de trabajos se ejecutan al mes a través de los servicios web.

### 1.8.8. Búsqueda

La búsqueda de información en las bases de datos bioinformáticas se suele hacer por analogía. Pongamos el caso de tener una secuencia proteica sin caracterizar. La secuencia es una sucesión de caracteres que por sí misma no proporciona mucha información. Sin embargo si comparamos esta secuencia con las casi 23 millones secuencias proteicas existentes en UniProtKB (alineamiento de secuencias), encontraríamos un listado de secuencias similares a la nuestra. En algunos casos la similitud sería tan grande que podríamos atribuir a nuestra secuencia las anotaciones de las secuencias similares. Estaríamos de esta manera caracterizando nuestra secuencia a través de la *búsqueda por semejanza* (ver Capítulo 8).

Imagina ahora que, con las veinte secuencias más parecidas a tu secuencia problema (extraídas a partir de la búsqueda por semejanza), generamos una secuencia genérica (patrón de secuencia) que reúna las características de todas ellas. Esta secuencia a su vez podría servir para buscar en bases de datos de patrones para encontrar otras secuencias genéricas similares a la que tú has generado. Este tipo de búsqueda se encuadraría dentro del *descubrimiento de patrones*.

La búsqueda en las bases de datos es un paso fundamental en la construcción de flujos de trabajo. Por ejemplo, para hacer la predicción de la estructura proteica a partir de su secuencia por el método de modelado por homología, el primer paso sería consultar a una base de datos de secuencias para extraer la secuencia más similar a la secuencia problema.

Pero también se pueden hacer *búsquedas más especializadas* en las bases de datos. Este es el caso del análisis de la huella peptídica resultante de un experimento de espectrometría de masas donde normalmente se utilizan bases de datos centradas en el organismo que se esté estudiando.

## 1.9. Bases de datos más relevantes

El elevado número de bases de datos bioinformáticas existentes actualmente hace imposible que se pueda hablar de cada una de ellas en un solo capítulo. Por esta razón describiremos únicamente dos

bases de datos: GenBank de secuencias nucleotídicas y UniProt de secuencias proteicas.

### 1.9.1. UniProt Knowledgebase

*UniProtKnowledgeBase* (UniProtKB) es una base de datos que proporciona una visión unificada de las secuencias proteicas junto con su información funcional [11]. UniProtKB tiene una amplia cobertura ya que incorpora, interpreta, anota, integra y estandariza datos procedentes de un gran número de bases de datos dispersas [5]. Esta base de datos está producida por el consorcio UniProt formado por grupos del EBI, del Instituto Suizo de Bioinformática (SIB) y de PIR (Protein Information Resource).

UniProtKB consta de dos secciones, UniProtKB/Swiss-Prot y UniProtKB/TrEMBL. La diferencia entre ambas secciones es que la primera de ellas está curada manualmente, lo que significa que la información de cada entrada es anotada y revisada por un revisor, mientras que UniProtKB/TrEMBL se genera automáticamente y está enriquecida mediante anotación y clasificación automática.

Las anotaciones de UniProtKB/Swiss-Prot consisten en la descripción de lo siguiente: función, información específica sobre enzimas, dominios y sitios biológicamente relevantes, modificaciones post-transducciónales, localización subcelular, especificidad de tejido, expresión específica en el desarrollo, estructura, interacciones, isoformas, enfermedades asociadas o deficiencias, etc. [15]. UniProtKB tiene actualmente casi 24 millones de registros. Alrededor de un 98 % de las secuencias almacenadas en UniProtKB proceden directamente de traducciones de las secuencias codificantes (CDS) submitidas al INSDC (International Nucleotide Sequence Database Collaboration) que está compuesto por ENA (European Nucleotide Archive), la base de datos de DNA de Japón (DDBJ) y GenBank de la cuál hablaremos también en esta sección. Los CDS se generan a través de programas de predicción o se prueban experimentalmente. Los CDS traducidos se transfieren automáticamente a la sección TrEMBL de UniProtKB. Luego los registros de TrEMBL pueden ser seleccionados para una posterior anotación manual e integración en la sección UniProtKB/Swiss-Prot. UniProtKB también acepta la integración de secuencias proteicas directamente secuenciadas a través de una herramienta web que permite a los investigadores añadir directamente las proteínas secuenciadas junto con los datos biológicos asociados.

En una entrada de UniProtKB podemos encontrar dos tipos de anotaciones: anotaciones generales (o comentarios) y anotaciones sobre la secuencia. La sección de comentarios alberga cualquier información útil que no encaja en ningún otro sitio de la entrada. Los comentarios son textos libres agrupados en bloques. Un bloque contiene una o más líneas de comentarios que se introducen por tema (Figura 1.10A). En la sección de anotaciones sobre la secuencia, la información sobre la proteína se mapea de forma precisa en la secuencia. La sección está organizada en una tabla con unos títulos auto-explicativos (Figura 1.10B).

Analicemos en profundidad la entrada correspondiente a la insulina humana (Figura 1.10).

- *Parte superior:* tenemos en la primera línea el nombre de la entrada INS\_HUMAN y en la segunda el identificador único P01308 (llamado número de acceso primario). La siguiente sección ofrece la descripción bioquímica de la proteína, incluyendo su nombre estándar, el nombre del gen que la codifica, el número de cadenas y el organismo de donde procede la proteína con su identificador taxonómico y su linaje.
- *Sección intermedia:* en la sección intermedia podremos encontrar las anotaciones generales (función, localización subcelular, uso farmacéutico, etc), las anotaciones sobre la secuencia (procesamiento de la molécula, amino ácidos modificados, variantes, estructura secundaria, etc.) y las palabras clave que describen la proteína (transporte de glucosa, hormona, etc). Las palabras clave proporcionan un resumen del contenido de la entrada. Estas palabras clave pertenecen a un vo-

**A)****General annotation (Comments)**

Function	Insulin decreases blood glucose concentration. It increases cell permeability to monosaccharides, amino acids and fatty acids. It accelerates glycolysis, the pentose phosphate cycle, and glycogen synthesis in liver.
Subunit structure	Heterodimer of a B chain and an A chain linked by two disulfide bonds.
Subcellular location	Secreted.

**B)****Sequence annotation (Features)**

Feature key	Position(s)	Length	Description	Graphical view	Feature identifier
<b>Molecule processing</b>					
Signal peptide	1 – 24	24	(Ref.12)		PRO_0000015819
Peptide	25 – 54	30	Insulin B chain (Ref.12)		PRO_0000015820
Propeptide	57 – 87	31	C peptide (Ref.13) (Ref.14)		PRO_0000015821
Peptide	90 – 110	21	Insulin A chain		PRO_0000015821
<b>Amino acid modifications</b>					
Disulfide bond	31 ↔ 96		Interchain (between B and A chains)		
Disulfide bond	43 ↔ 109		Interchain (between B and A chains)		
Disulfide bond	95 ↔ 100		(Ref.13)		

**Figura 1.10:** En esta figura se muestran algunas de las anotaciones sobre la secuencia y generales referidas a la insulina humana (P01308). A) Los comentarios sobre la insulina están divididos en tres bloques: función (descenso de los niveles de glucosa en sangre), estructura de la subunidad (heterodímero unido por dos puentes disulfuro) y localización sub-cellular (secretada). B) En la sección de anotaciones sobre la secuencia encontramos los títulos procesamiento de la molécula y aminoácidos modificados.

cabulario controlado desarrollado por UniProtKB donde existen 10 categorías: proceso biológico, componente celular, diversidad de la secuencia codificante, estadio del desarrollo, enfermedad, dominio, ligando, función molecular, modificaciones post-traduccionales y términos técnicos. Cada uno de estos términos se atribuyen de forma manual a las entradas de UniProt/SwissProt y automáticamente a las de UniProt/TrEMBL [11].

- *Sección inferior* (sección secuencia): proporciona la secuencia de amino ácidos de la proteína. Se puede ver como la secuencia correspondiente a la insulina humana contiene 110 amino ácidos, con un peso molecular total de 11,981 Da. Para estudios posteriores, será necesario copiar la secuencia en formato FASTA, un formato muy simple que veremos en la Sección 1.10.

A continuación encontraremos las referencias bibliográficas y las referencias cruzadas. El consorcio UniProt está muy comprometido en impulsar la interacción y el intercambio de datos con la comunidad científica, asegurando un amplio acceso a los recursos de UniProt y promoviendo la interoperabilidad entre los mismos. Prueba de ello son los más de 10 millones de enlaces que UniProt proporciona a más de 120 bases de datos [15].

### 1.9.2. GenBank

*GenBank*<sup>2</sup> es una base de datos que contiene secuencias nucleotídicas correspondientes a más de 250.000 especies además de información bibliográfica y anotaciones biológicas. Estas secuencias se obtienen principalmente de envíos por parte de autores o de envíos masivos procedentes de Etiquetas de Secuencias Expresadas (Expressed Sequence Tags –EST-), secuencia del genoma encuesta (Genome Survey Sequence –GSS-), secuenciación aleatoria del genoma (Whole-Genome Shotgun –WGS-) y otros datos de centros de secuenciación masiva. Los datos de GenBank se intercambian diariamente con el Archivo Europeo de Nucleótidos (ENA) y con la Base de Datos de DNA de Japón (DDBJ) ya que los tres son socios dentro de la Colaboración Internacional de la Base de Datos de Secuencias Nucleotídicas (INSDC) [1].

La base de datos GenBank fue concebida por el Centro Nacional de Información en Biotecnología (National Centre for Biotechnology Information –NCBI-), una división de la Biblioteca Nacional de Medicina (National Library of Medicine –NLM-), ubicada en el campus de los Institutos Nacionales de Salud (National Institutes of Health –NIH- ) en Bethesda, MD, USA.

GenBank agrupa las secuencias en varias divisiones de acuerdo con la taxonomía o la estrategia de secuenciación llevada a cabo para obtener los datos. Actualmente hay doce divisiones taxonómicas y seis divisiones de secuenciación masiva (Tabla 1.1. La división PAT tiene entradas enviadas por las oficinas de patentes y la división WGS tiene secuencias procedentes de los proyectos WGS.

División	Descripción	Pares de bases
TSA	Transcriptome shotgun data	1 874 047 448
ENV	Muestras ambientales	2 553 693 157
PHG	Fagos	62 579 756
PAT	Secuencias patentadas	11 154 487 762
BCT	Bacterias	6 975 597 755
INV	Invertebrados	2 535 336 197
WGS	Whole Genome Shotgun data	208 315 831 132
VRL	Virus 1	180 083 600
MAM	Otros mamíferos	807 098 397
PLN	Plantas 4	741 991 057
GSS	Genome Survey Sequences	20 770 772 329
SYN	Sintético	156 218 063
VRT	Otros vertebrados	22 105 250 711
EST	Expressed Sequence Tags	39 018 185 344
UNA	No anotados	125 912
PRI	Primates	6 116 546 725
ROD	Roedores	4 396 957 541
HTC	cDNA masivo	662 320 919
STS	Sequence tagged sites	653 872 683
HTG	Hight-throughput genomic	338 987 064 933

**Tabla 1.1:** Divisiones de GenBank

<sup>2</sup>GenBank. <http://www.ncbi.nlm.nih.gov/genbank>

## 1.10. Obtención de secuencias y formatos

Normalmente el resultado de consulta a una base de datos bioinformática es una página web que detalla toda la información sobre el dato junto con las anotaciones correspondientes. Esta información además se puede descargar como un fichero con diversos formatos. En bioinformática, el formato estándar para las secuencias tanto de proteínas como de ácidos nucleicos es el *formato FASTA*. Una secuencia en formato FASTA comienza con una línea descriptiva, seguida por las líneas correspondientes a la secuencia (siempre representada en códigos de una única letra). La línea descriptiva se distingue de la de la secuencia en que tiene un símbolo mayor que (>) en la primera columna. La palabra que sigue al símbolo es el *identificador* de secuencia. No debe haber espacio entre el símbolo ">" y la primera letra del identificador (Figura 1.11B). Como se puede observar, el formato FASTA es un formato abreviado, es decir, da la información esencial sobre la secuencia y la cadena de caracteres correspondiente a la secuencia en sí. Además, la información sobre la secuencia puede omitirse: podríamos tener una secuencia en formato FASTA donde solamente encontráramos el signo ">", retorno de línea y la secuencia de caracteres.

### A) Formato GenBank

```
LOCUS      JK340485          291 bp  mRNA    linear   EST 12-AUG-2011
DEFINITION CdL15 Common bermudagrass (Cynodon dactylon cv. C299) drought 5
days suppression subtractive hybridization cDNA library Cynodon
dactylon cDNA clone CdL15 similar to putative insulin degrading
enzyme, mRNA sequence.
ACCESSION JK340485
VERSION  JK340485.1 GI:343410571
KEYWORDS EST.
SOURCE   Cynodon dactylon (Bermuda grass)
ORGANISM Cynodon dactylon
           Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
           Spermatophyta; Magnoliophyta; Liliopsida; Poales; Poaceae; PACMAD
           clade; Chloridoideae; Cydonioideae; Eleusininae; Cynodon.
REFERENCE 1 (bases 1 to 291)
AUTHORS Zhou,P., Hu,L., Wang,Z., Du,H., An,Y. and Huang,B.
TITLE Identification of stress-responsive genes for tolerance to
short-term and long-term water deficit in hybrid bermudagrass
[Cynodon dactylon (L.) Pers. x C. transvaalensis Burtt Davy] and
common bermudagrass (C. dactylon) contrasting in drought tolerance
JOURNAL Unpublished (2011)
COMMENT Contact: Zhou Peng
Key Laboratory of Urban Agriculture (South)
School of Biological and Agricultural Science, Shanghai Jiaotong
University
Building of Agriculture & Biology, 800 Dongchuan Road, Shanghai
200240, P.R. China
Tel: +86 21 3420 5976
Fax: +86 21 3420 5976
Email: pzhou0063@sjtu.edu.cn
The sequence was obtained from common bermudagrass 'C299' leaves
under 5 days drought.
Seq primer: M13 Forward
POLYA>No.
FEATURES
source      Location/Qualifiers
           1..291
           /organism="Cynodon dactylon"
           /mol_type="mRNA"
           /cultivar="C299"
           /db_xref="taxon:28909"
           /db_xref="biosample:706886"
           /clone="CdL15"
           /tissue_type="leaves"
           /dev_stage="Vegetative stage"
           /lab_host="E. coli DH5-alpha"
           /clone_lib="LIBEST_027413 Common bermudagrass (Cynodon
dactylon cv. C299) drought 5 days suppression subtractive
hybridization cDNA library"
           /note="Vector: pMD18-T simple vector; Site_1: EcoRV; Total
RNA was extracted from fully-expanded leaves of common
...
each ligated with two different cDNA adaptors. Two rounds
of hybridization and PCR amplification was used to screen
the different expression gene. And the products were
directly cloned into the pMD18-T vector."
ORIGIN
1 ctggacgcta gaggtaaga gttcttaag atgttgaaa gaaatatcca tgaactgtcc
61 gacaaggatt tcaagatgc ttgtaaatcg cttaattgtt ccaaactggaa gaaattccaa
121 aacttgtggg aggtatctca cttttactgg ggagagatcg aggtctggaaatcgttca
181 gatcggttg agactqagggt ggctctctgtt agagaactgtt agaaagaga attcatgttt
241 tttttcgacc ggcacataaa agttgacgtt cttgtaaaggaa ggactataag c
//
```

### B) Formato FASTA

```
>gi|343410571|gb|JK340485.1|JK340485 CdL15 Common bermudagrass (Cynodon dactylon
cv. C299) drought 5 days suppression subtractive hybridization cDNA library
Cynodon dactylon cDNA clone CdL15 similar to putative insulin degrading enzyme,
mRNA sequence
CTTGACGCTAGAGTGTGAAGAGTCTTTAAGATGTTGAAAGAAAATATCCATGAACTGTCCGACRAAGGATT
TCAAGAGATAATGTGAATTCGCTTATGTGATTCCAACACTGGAGAAATTCAAAACACTTGTGGGAGGTAACTCTCA
CTTTACTGGGGAGAGATCGAGGCTGGAACTCTCAAGTTGATCGGGTTGAGACTGAGGTGCTCTCTG
AGAGAACTGAAGAAAGAAGAAATCAICGAATTITTCGACCGCACATAAAAGTTGACGGCTCTGAAAGGA
GGACTATAGC
```

Figura 1.11: Entrada JK34485 en formato GenBank (A) y en formato FASTA (B).

En cambio, los formatos propios de las bases de datos de origen como son los *formatos GenBank o UniProt*, incluyen todas las anotaciones sobre la secuencia y anotaciones generales (Figura 1.11A).

### 1.11. Ejercicio: acceso a las bases de datos a través de los motores de búsqueda del NCBI (Entrez) y del EBI (EB-Eye)

Entrez es un sistema de recuperación de información de bases de datos que proporciona acceso a un conjunto de 40 bases de datos muy diversas. Entrez es el principal sistema de búsqueda en el NCBI que integra la base de datos de literatura biomédica PubMed (Capítulo 2) con otras 39 bases de datos moleculares y de referencias bibliográficas que proporcionan información sobre secuencias de DNA y proteínas, estructura, genes, genomas, variaciones y expresión génica. Entrez permite al usuario las consultas por texto utilizando booleanos, bajar los datos en varios formatos y enlazar los registros entre bases de datos basándose en relaciones biológicas [17].

La Figura 1.12 muestra la página global de la consulta que contiene el resultado de la selección de todas las bases de datos. Esta página lista las bases de datos de Entrez junto con el número de registros encontrados por la consulta en cada base de datos. Las bases de datos están organizadas en tres secciones: la sección en la parte de arriba contiene las bases de datos de literatura: PubMed, PubMed Central, Books, OMIM y OMIA. Las 26 bases de datos moleculares ocupan la sección central de la página. La sección inferior alberga las bases de datos accesorias de literatura como MeSH y el catálogo NLM.

La página global de consulta por sí misma puede usarse para buscar en todas las bases de datos mencionadas anteriormente mediante la introducción de un término o frase en la caja de búsqueda. Si hacemos clic en el número o en el nombre de la base de datos adyacente, obtendremos los resultados en esa base de datos. Las consultas en Entrez pueden ser palabras sueltas, frases cortas, identificadores de bases de datos, símbolos de genes, nombres de genes o de proteínas... etc.

Vamos a comenzar nuestra búsqueda en Entrez con el nombre de una proteína: insulina. Para ello introducimos el término ‘insulin’ en la caja de búsqueda. Si no le ponemos ninguna etiqueta al término, el sistema buscará este término en todos los campos de todas las bases de datos que tiene indexadas. Esto nos da un número bastante elevado de registros. Observa por ejemplo que tenemos más de 290.000 artículos de PubMed donde se menciona esta proteína. Además, la secuencia de DNA que codifica para esta proteína ha sido mapeada en 100 genomas diferentes (completamente secuenciados o no). Sin embargo si marcamos el término ‘insulin’ con la etiqueta ‘protein’ (insulin[Protein]), entonces el sistema únicamente buscará este término en los campos relacionados con proteínas. Como consecuencia de esto, no encontraremos registros en las bases de datos GSS o EST. Pero ¿qué ocurre con la base de datos Nucleotide? Observamos en esta base de datos un descenso en el número de registros: de 26134 a 81. La base de datos Nucleotide contiene todas las secuencias nucleotídicas procedentes de GenBank, EMBL y DDBJ. Por lo tanto, si Nucleotide sólo contiene secuencias nucleotídicas y nosotros hemos puesto la etiqueta [Protein] a la insulina, ¿cómo es que nos aparecen 81 registros como resultado de la búsqueda? Haz clic en el enlace a esta base de datos. Observarás que efectivamente todos los registros corresponden a secuencias nucleotídicas. Sin embargo, si observas con detenimiento cualquiera de estos registros (por ejemplo NM\_131056), te darás cuenta de que en ellos se proporciona la información sobre las secuencias que codifican proteínas (CDS). En el registro NM\_131056 tenemos un CDS que codifica para una proteína llamada ‘insulin preprotein’ con número de acceso NP\_571131.1. Entrez también permite acotar la búsqueda a un único organismo, por ejemplo humano. En este caso, hay que tener cuidado de poner la búsqueda entre paréntesis (comprueba qué ocurre si no lo haces): (insulin[Protein] AND human[Organism]).

The screenshot shows the NCBI Entrez search interface. At the top, there's a navigation bar with links for HOME, SEARCH, and SITE MAP. Below this is a main search bar labeled "Search across databases" with a dropdown menu set to "all[filter]". To the right of the search bar are buttons for GO, Clear, and Help.

**Welcome to the Entrez cross-database search page**

The page displays a grid of database entries, each with an icon, name, and brief description. The entries are organized into several sections:

- PubMed:** biomedical literature citations and abstracts
- PubMed Central:** free, full text journal articles
- Site Search:** NCBI web and FTP sites
- Books:** online books
- OMIM:** online Mendelian Inheritance in Man

**Large section:**

- Nucleotide:** Core subset of nucleotide sequence records
- EST:** Expressed Sequence Tag records
- GSS:** Genome Survey Sequence records
- Protein:** sequence database
- Genome:** whole genome sequences
- Structure:** three-dimensional macromolecular structures
- Taxonomy:** organisms in GenBank
- SNP:** short genetic variations
- dbVar:** Genomic structural variation
- Gene:** gene-centered information
- SRA:** Sequence Read Archive
- BioSystems:** Pathways and systems of interacting molecules
- HomoloGene:** eukaryotic homology groups
- Probe:** sequence-specific reagents
- BioProject:** aggregated biological research project data
- dbGaP:** genotype and phenotype
- UniGene:** gene-oriented clusters of transcript sequences
- CDD:** conserved protein domain database
- Clone:** integrated data for clone resources
- UniSTS:** markers and mapping data
- PopSet:** population study data sets
- GEO Profiles:** expression and molecular abundance profiles
- GEO DataSets:** experimental sets of GEO data
- Epigenomics:** Epigenetic maps and data sets
- PubChem BioAssay:** bioactivity screens of chemical substances
- PubChem Compound:** unique small molecule chemical structures
- PubChem Substance:** deposited chemical substance records
- Protein Clusters:** a collection of related protein sequences
- OMIA:** online Mendelian Inheritance in Animals
- BioSample:** biological material descriptions

**Small section:**

- NLM Catalog:** catalog of books, journals, and audiovisuals in the NLM collections
- MeSH:** detailed information about NLM's controlled vocabulary

At the bottom of the page, there are links for Counts in XML, Entrez Utilities, Disclaimer, Privacy statement, and Accessibility.

**Figura 1.12:** Página de resultados global de Entrez que muestra los resultados de una búsqueda por todos los campos en todas las bases de datos (all[Filter]). La búsqueda ha sido realizada a partir de la caja de búsqueda de la página home del NCBI con todas las bases de datos seleccionadas.

El EBI indexa y actualiza más de 300 millones de entradas cada día. El acceso a estos datos se hace a través de un motor de búsqueda llamado EB-eye. EB-eye está accesible desde cualquier página del EBI y está diseñado para permitir a los usuarios realizar búsquedas basadas en texto a través de los campos más comúnmente usados de todas las colecciones de datos clave sin necesidad de ningún conocimiento previo de las mismas. Los resultados se presentan como una lista expandible de “dominios de conocimiento” que cubren diferentes tipos de datos: secuencias nucleotídicas, secuencias proteicas, macromoléculas, etc. Cada dominio de conocimiento puede ser expandido, lo que permite al usuario examinar a fondo una base de datos individual, o incluso un campo de una base de datos específica. Además, las referencias cruzadas que existen entre las diferentes bases de datos permiten la navegación de una base de datos a otra [18].

**A) Página de resultados**

Showing 18 results out of 363,197 in All results

Filter your results

**Source**

- All results (363,197)
- Genomes (1,604)
- Nucleotide Sequences (25,538)
- Protein Sequences (8,625)
- Macromolecular Structures (647)
- Small molecules (929)
- Gene Expression (16,749)
- Molecular Interactions (167)
- Reactions, Pathways & Diseases (1,322)
- Protein Families (175)
- Enzymes (50)
- Literature (307,047)

**D) Listado de categorías**

**E) Acotación de los registros**

**B) secciones individuales**

**C) resultados principales de la búsqueda**

Showing 18 results out of 363,197 in All results → Genomes → Ensembl Gene

Filter your results

**Source**

- All results (363,197)
- Genomes (1,604)
- Ensembl Gene (1,431)

**Ensembl Gene (1,431 results found)**

- OTTHUMG00000005588 insulin Species homo sapiens
- ENSDORG00000003743 Insulin-like 6 [Source: MGI Symbol:Acc:MGI:1351595] Species Drosophila ordi

**Figura 1.13:** Página de resultados del motor de búsqueda EB-eye. A: Página de resultados de la búsqueda del término ‘insulin’. B: Sumario de uno de resultados. C: Listado de resultados por categoría y base de datos. D: Listado de categorías para el filtrado de los resultados. E: Acotación de los registros.

Veamos cuál es el resultado de preguntar en el motor del EBI por el término ‘insulin’. Como puedes observar en la Figura 1.13A, la manera en que está organizada la página de resultados es muy diferente a la del NCBI. En la parte superior derecha encontraremos la sección ‘Genes & Proteins summaries’. Esta sección proporciona una manera muy útil de explorar los datos del EBI desde la perspectiva de un gen o proteína para ciertas especies clave (humano, ratón, levadura, nematodo y mosca de la fruta). El sumario recopila datos de varias bases de datos del EBI para presentar la vista de un gen de una manera organizada, intuitiva y relevante. Para cada gen/proteína, el sumario comprende cinco secciones individuales y la posibilidad de pasar de una a otra: gen, expresión, proteína, estructura proteica, y literatura (Figura 1.13B). También puedes desplegar la información equivalente para un gen ortólogo en otra especie. La sección más extensa corresponde a los resultados principales de la búsqueda y consiste en un listado de registros o páginas encontradas en la búsqueda organizadas por categoría y base de datos (Figura 1.13C). La columna de la izquierda es el listado de categorías (Figura 1.13D). Estas categorías son a su vez enlaces que permiten navegar en la lista de resultados principales limitándolos a registros dentro de una categoría particular. Por ejemplo, si selecciono la categoría Genomes/Ensembl Gene, pasará de 363.197 registros a 1431, correspondientes únicamente a entradas de Ensembl Gene (Figura 1.13E).



## 1.12. Bibliografía

- [1] D. A. Benson, I. Karsch-Mizrachi, K. Clark, D. J. Lipman, J. Ostell, and E. W. Sayers. Genbank. *Nucleic Acids Res*, 40(Database issue):D48–53, 2012.
- [2] C. Brocksbank, G. Cameron, and J. Thornton. The european bioinformatics institute’s data resources. *Nucleic Acids Res*, 38(Database issue):D17–25, 2010.
- [3] S. Burge, T. K. Attwood, A. Bateman, T. Z. Berardini, M. Cherry, C. O’Donovan, L. Xenarios, and P. Gaudet. Biocurators and biocuration: surveying the 21st century challenges. *Database (Oxford)*, 2012:bar059, 2012.
- [4] M. Chagoyen. *Integration of biological data: systems, infrastructures and programmable tools*. PhD thesis, Universidad Autónoma de Madrid, 2005.
- [5] U. Consortium. Reorganizing the protein space at the universal protein resource (uniprot). *Nucleic Acids Res*, 40(Database issue):D71–5, 2012.
- [6] R. G. Cote, P. Jones, R. Apweiler, and H. Hermjakob. The ontology lookup service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC Bioinformatics*, 7:97, 2006.
- [7] P. Flicek, M. R. Amode, D. Barrell, K. Beal, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gil, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. K. Kahari, D. Keefe, S. Keenan, R. Kinsella, M. Komorowska, G. Koscielny, E. Kulesha, P. Larsson, I. Longden, W. McLaren, M. Muffato, B. Overduin, M. Pignatelli, B. Pritchard, H. S. Riat, G. R. Ritchie, M. Ruffier, M. Schuster, D. Sobral, Y. A. Tang, K. Taylor, S. Trevanion, J. Vandurovcova, S. White, M. Wilson, S. P. Wilder, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernandez-Suarez, J. Harrow, J. Herrero, T. J. Hubbard, A. Parker, G. Proctor, G. Spudich, J. Vogel, A. Yates, A. Zadissa, and S. M. Searle. Ensembl 2012. *Nucleic Acids Res*, 40(Database issue):D84–90, 2012.
- [8] M. Y. Galperin and X. M. Fernandez-Suarez. The 2012 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Res*, 40(Database issue):D1–8, 2012.
- [9] P. Gaudet, A. Bairoch, D. Field, S. A. Sansone, C. Taylor, T. K. Attwood, A. Bateman, J. A. Blake, C. J. Bult, J. M. Cherry, R. L. Chisholm, G. Cochrane, C. E. Cook, J. T. Eppig, M. Y. Galperin, R. Gentleman, C. A. Goble, T. Gojobori, J. M. Hancock, D. G. Howe, T. Imanishi, J. Kelso, D. Landsman, S. E. Lewis, I. K. Mizrachi, S. Orchard, B. F. Ouellette, S. Ranganathan, L. Richardson, P. Rocca-Serra, P. N. Schofield, D. Smedley, C. Southan, T. W. Tan, T. Tatusova, P. L. Whetzel, O. White, and C. Yamasaki. Towards biobdbcore: a community-defined information specification for biological databases. *Nucleic Acids Res*, 39(Database issue):D7–10, 2011.
- [10] P. Gaudet and R. Mazumder. Biocuration virtual issue 2012. *Database (Oxford)*, 2012:bas011, 2012.
- [11] M. Magrane and U. Consortium. Uniprot knowledgebase: a hub of integrated protein data. *Database (Oxford)*, 2011:bar009, 2011.
- [12] A. Mitra. Classifying data for successful modeling, 2012.
- [13] A. Mitra. Data mining, a simple guide for beginners, 2012.
- [14] M. A. Musen, N. F. Noy, N. H. Shah, P. L. Whetzel, C. G. Chute, M. A. Story, and B. Smith. The national center for biomedical ontology. *J Am Med Inform Assoc*, 19(2):190–5, 2012.
- [15] C. O’Donovan and R. Apweiler. A guide to uniprot for protein scientists. *Methods Mol Biol*, 694:25–35, 2011.
- [16] K. D. Pruitt, T. Tatusova, G. R. Brown, and D. R. Maglott. Ncbi reference sequences (refseq): current status, new features and genome annotation policy. *Nucleic Acids Res*, 40(Database issue):D130–5, 2012.
- [17] E. W. Sayers, T. Barrett, D. A. Benson, E. Bolton, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, S. Federhen, M. Feolo, I. M. Fingerman, L. Y. Geer, W. Helmberg, Y. Kapustin, S. Krasnov, D. Landsman, D. J. Lipman, Z. Lu, T. L. Madden, T. Madej, D. R. Maglott, A. Marchler-Bauer, V. Miller, I. Karsch-Mizrachi, J. Ostell, A. Panchenko, L. Phan, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, M. Shumway, K. Sirotnik, D. Slotta, A. Souvorov, G. Starchenko, T. A. Tatusova, L. Wagner, Y. Wang, W. J. Wilbur, E. Yaschenko, and J. Ye. Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 40(Database issue):D13–25, 2012.
- [18] F. Valentin, S. Squizzato, M. Goujon, H. McWilliam, J. Paern, and R. Lopez. Fast and efficient searching of biological data resources—using eb-eye. *Brief Bioinform*, 11(4):375–84, 2010.



## **Capítulo 2**

# **Gestión de bibliografía e información y escritura de artículos científicos**

*José María Fernández y Álvaro Sebastián*

### **2.1. Introducción**

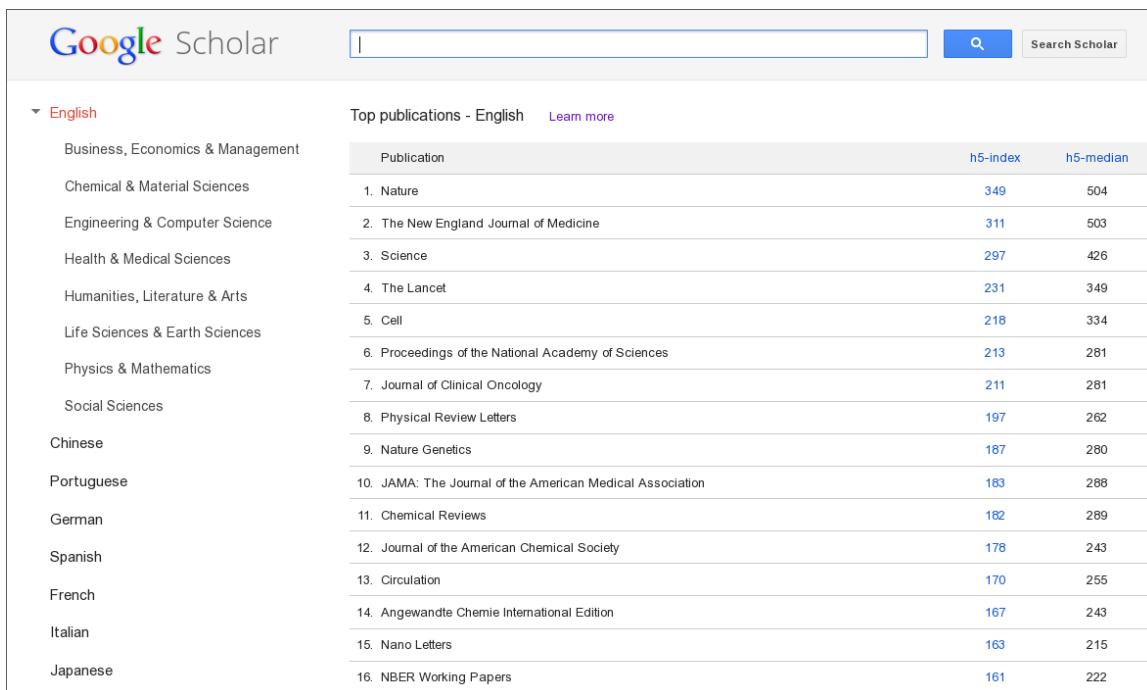
Aunque en el mundo de la ciencia los descubrimientos y los resultados científicos obtenidos se difundan mediante ponencias (por ejemplo, en congresos), principalmente se dan a conocer mediante publicaciones científicas. La cantidad de conocimiento acumulado en publicaciones científicas es descomunal, y sólo en el área de las ciencias de la vida, a finales de 2013 había registrados más de 23 millones de artículos científicos en la base de datos PubMed (ver Subsección 2.2.2), provenientes de más de 5000 revistas científicas.

Una parte básica del trabajo de cualquier investigador de un área tan cambiante como la de las Ciencias de la Vida es estar al día sobre las publicaciones relevantes para los proyectos en los que está trabajando: ver cuáles de esas publicaciones tienen información relevante para los proyectos; ver si se nos han adelantado a la hora de publicar un descubrimiento o desarrollo, etc. Por eso es vital realizar una gestión documentada, como por ejemplo qué artículos han sido leídos, anotaciones de esos artículos o qué trabajos existentes se han tomado como base para realizar los desarrollos, entre otros.

Por otro lado, una forma de reconocer el mérito de los trabajos realizados es mediante la publicación de dichos trabajos. Para que un trabajo científico salga publicado en una revista científica debe normalmente superar una revisión por pares realizada por otros científicos del campo, coordinada y aprobada por uno de los editores de la revista. La relevancia de las publicaciones científicas de un investigador determinan el transcurso de su carrera científica. A pesar de que no es la métrica más adecuada, a la hora de optar por una plaza o solicitar un proyecto científico se tiene en cuenta el número de publicaciones científicas, cuándo se publicaron y su impacto desde entonces. Y muchas veces, independientemente de la calidad científica del trabajo a publicar, la diferencia entre publicar o no publicar es elegir la revista más adecuada para ese trabajo.

## 2.2. Principales bases de datos de literatura científica y ejemplos de búsqueda

Al principio, la única manera de tener acceso a los artículos de una revista científica era teniendo una *suscripción física* a la misma. El inconveniente de este modelo de negocio es que el acceso al conocimiento dentro de las publicaciones de esas revistas estaba muy restringido, lo cuál es un freno para el desarrollo científico presente y futuro. Además, se ha dado en más de una ocasión la paradoja de que, sin saberlo y sin haber plagio de por medio, dos o más investigadores hayan publicado el mismo descubrimiento encontrado de forma independiente.



**Figura 2.1:** Revistas científicas más relevantes en 2013, según Google Scholar.

En la primera década del siglo XXI surgió una alternativa a este modelo de negocio, que es el modelo de *libre acceso*. En este modelo los artículos publicados en una revista están disponibles para todo el mundo pasado un tiempo (ej. 6 meses o un año) o incluso inmediatamente. Este modelo es viable económicamente sustituyendo parte de los ingresos obtenidos por las suscripciones por aportaciones de los autores (si te admiten un artículo, para que salga publicado tienes que pagar). Actualmente, varios organismos públicos de investigación, entre ellos, el NIH, obligan a que todos los artículos que surjan de proyectos financiados parcial o totalmente por ellos se publiquen en revistas que proporcionen libre acceso a los artículos.

Como se puede ver en la Figura 2.1, algunas de las revistas más conocidas en el ámbito de las ciencias de la vida son *New England Journal of Medicine*, *Cell*, *Nature*, *Science* y *Proceedings of the National Academy of Sciences*. En el campo de la bioinformática revistas que tienen un gran índice de impacto son *Nucleic Acid Research*, *Bioinformatics*, *BMC Bioinformatics* y *Briefings in Bioinformatics*. Buena parte de ellas (pero no todas) ya siguen el modelo de libre acceso a los artículos publicados.

Los principales problemas a los que nos enfrentamos al realizar una búsqueda en la literatura científica son las *ambigüedades*, los *homónimos* (términos o acrónimos que se escriben igual pero que tienen un

significado totalmente diferente), los *sinónimos* (diversas palabras o expresiones que se refieren a un mismo concepto) y las *erratas y errores de redacción*. Por ello, para hacer búsquedas más detalladas y precisas se suele recurrir a alguno de los muchos sistemas especializados de *minería de textos* (similar a la minería de datos explicada en el Capítulo 5), como por ejemplo iHOP<sup>1</sup> [1], PubReMiner<sup>2</sup>, o a búsquedas en PubMed, Google Scholar o incluso usando buscadores generales como Google o Bing.

### 2.2.1. Servidores de artículos por publicar

Tal como es el sistema científico actual, sólo se da reconocimiento a los investigadores por sus descubrimientos publicados, no antes. Esto hace que, en el mejor de los casos, desde que se terminan los experimentos y se empieza a escribir el artículo, hasta que dicho artículo sale por fin publicado, pueda pasar mínimo un año, si no más. En el peor y bastante común de los casos, muchos descubrimientos no llegan a ser publicados, o no se encuentran lo suficientemente contrastados, condenando a toda la comunidad científica a “redescubrirlos”. Para paliar de alguna manera este problema, se han creado *repositorios abiertos de artículos*, como arXiv<sup>3</sup>, donde los investigadores pueden depositar sus artículos para que cualquiera pueda dar su opinión constructiva, y que quede constancia de que han hecho un descubrimiento. Salvo en casos muy excepcionales, el envío como autor de artículos no publicados a arXiv no impide que posteriormente se pueda mandar una versión revisada de dichos artículos a cualquier revista científica para intentar publicarlos.

arXiv es una iniciativa mantenida actualmente en la Universidad de Cornell que comenzó a funcionar en Agosto de 1991, cubriendo áreas de investigación como matemáticas o ciencias de la computación, donde es bastante común la práctica de compartir las ideas antes de ser publicadas para contrastarlas. Desde entonces, poco a poco más áreas de investigación (como la biología molecular) se han ido sumando a esta práctica, almacenando a finales de 2013 más de 900000 artículos. Una de las peculiaridades de arXiv es que permite a los autores de los artículos ir subiendo actualizaciones de los mismos, manteniendo almacenadas al mismo tiempo las versiones anteriores. De esta manera se puede mantener sobre qué versión de un artículo añadió la comunidad científica comentarios y críticas, y cualquiera de ellos puede revisar a posteriori si las razones a dichas críticas han sido debidamente subsanadas.

### 2.2.2. PubMed y PubMed Central

La base de datos *PubMed*<sup>4</sup> es un recurso de libre acceso desarrollado y mantenido por el National Center for Biotechnology Information (NCBI), dentro de la National Library of Medicine (NLM) de Estados Unidos, ubicada en el National Institutes of Health (NIH). Dentro de esta base de datos *se almacenan citas a publicaciones científicas de diverso tipo*, como referencias a literatura biomédica que ha sido anotada en MEDLINE, publicaciones científicas del área de ciencias de la vida y libros disponibles *online*.

La base de datos PubMed puede ser consultada de forma programática, siendo capaz de devolver los registros encontrados en diversos formatos: textual, ASN.1, referencias BibTeX, XML, etc. Además, el contenido completo de la base de datos está disponible para descarga, bajo petición formal al NLM/NIH declarando cuál va a ser su uso, ya que no es gratuita para uso comercial. Buena parte de las entradas de PubMed, al tener su origen en MEDLINE, se encuentran anotadas mediante términos de la ontología MeSH (*Medical Subject Headings*), que es revisada y actualizada anualmente. A su vez, las anotaciones

<sup>1</sup>iHOP. <http://www.ihop-net.org/UniPub/iHOP>

<sup>2</sup>PubReMiner. <http://hgserver2.amc.nl/cgi-bin/miner/miner2.cgi>

<sup>3</sup>arXiv. <http://www.arxiv.org>

<sup>4</sup>PubMed. <http://www.ncbi.nlm.nih.gov/pubmed>

son periódicamente revisadas, sobre todo para los artículos recientemente registrados. Para todos los artículos que hay en PubMed provenientes de revistas se almacena el resumen del artículo, lo cuál sirve para hacerse una idea del contenido del mismo, e incluso para buscar por su contenido.

*PubMed Central*<sup>5</sup> se creó tras el auge del modelo de negocios de libre acceso a los artículos en las revistas científicas. Es una *biblioteca electrónica de libre disposición* de literatura biomédica y de ciencias de la vida, también gestionado en el NLM/NIH. Esta biblioteca almacena y clasifica de forma homogénea los artículos (y el material suplementario asociado a ellos) de aquellas revistas que han optado por proporcionarlo. Una de las premisas de PubMed Central es que, todo artículo almacenado ahí estará disponible de forma gratuita de forma permanente. La propiedad intelectual de cada artículo almacenado en PubMed Central sigue perteneciendo a la revista donde fue publicado y a sus autores. Los artículos de PubMed que están disponibles en PubMed Central están enlazados entre sí en ambas webs, además de a la publicación original. A finales de 2013 había más de 2.9 millones de artículos completos almacenados en PubMed Central, con casi 1400 revistas que han decidido proporcionar la totalidad de sus artículos a PubMed Central y más de 2400 publicaciones que proporcionan parte de sus artículos.

Aquellas revistas que se comprometen a que sus artículos aparezcan en PubMed o PubMed Central mandan periódicamente la información de los nuevos artículos a la biblioteca siguiendo las normativas y las normas de calidad de los proveedores de datos de PubMed y PubMed Central. Dicha información proporcionada debe seguir una estructura XML muy cercana a la usada para los registros de PubMed, y se sube a una cuenta FTP previamente proporcionada por los administradores de PubMed. Una vez enviada la información, si es correcta, se integra en PubMed pasadas 48 horas desde su subida. Si la revista además tiene interés en proporcionar un enlace al artículo completo en PubMed Central (por ejemplo, desde su propia web), entonces tiene además que usar el servicio LinkOut<sup>6</sup> del NCBI.

## Uso básico de PubMed

El método principal de búsqueda en PubMed es usando una expresión en texto libre que contenga las *palabras clave más representativas* de la búsqueda que estamos iniciando, como los apellidos de algunos de los autores, nombre de enfermedad, síntoma o gen, etc. Por ejemplo, si buscáramos artículos relacionados con la enfermedad de Huntington, usando la palabra ‘huntington’, PubMed nos devolvería los artículos que la tengan en el título, resumen, e incluso autores. Saldría un resultado similar al de 15296 resultados de la Figura 2.2.

Al ser una búsqueda tan general, de un solo término, en la página aparece incluso una distribución del término de búsqueda por año. Si nuestro interés está relacionado con las mutaciones relacionadas con la enfermedad, podríamos *refinar la búsqueda* buscando ‘huntington mutant’, bajando el número de resultados a 1193. Pero, como los artículos indexados en PubMed pueden referir no sólo a la enfermedad de Huntington humana, al *filtrar por especie* (como se puede ver en la Figura 2.3) se reducirían los resultados a 826.

Aunque hemos aplicado el *filtro en el interfaz web*, el motor de búsquedas de PubMed permite incluir dicho filtro (y otros) a mano en la cadena de búsqueda. La forma de expresar esta misma búsqueda de forma escrita sería:

```
huntington mutant AND "humans" [MeSH Terms]
```

---

<sup>5</sup>PubMed Central. <http://www.ncbi.nlm.nih.gov/pmc>

<sup>6</sup>LinkOut. <http://www.ncbi.nlm.nih.gov/projects/linkout>

The screenshot shows the PubMed search interface with the query 'huntington'. The results page displays 1 to 20 of 15296 articles. The sidebar on the left includes filters for Article types (Clinical Trial, Review, More...), Text availability (Abstract available, Free full text available, Full text available), PubMed Commons (Reader comments), Publication dates (5 years, 10 years, Custom range...), and Species (Humans, Other Animals). The right side features a 'New feature' section about display settings, a 'Results by year' chart, and a 'Related searches' section listing terms like 'huntington disease review', 'huntington chorea', 'huntington disease treatment', 'huntington disease genetic', and 'juvenile huntington'.

**Figura 2.2:** Búsqueda inicial con la palabra clave 'huntington'.

This screenshot shows the same search results as Figure 2.2, but with a specific filter applied: 'Species' set to 'Humans'. The results now show 1 to 20 of 826 articles. The 'Species' filter option is circled in red. The right side of the screen includes a 'New feature' section, a 'Titles with your search terms' section listing 'Synaptic mutant huntingtin inhibits synapsin-1 phosphorylation and causes neurological symptoms.', 'Glutathione peroxidase activity is neuroprotective in models of Huntington's disease.', and 'HTRF analysis of soluble huntingtin in PHAROS PBMCs.', and a '416 free full-text articles in PubMed Central' section.

**Figura 2.3:** Búsqueda usando 'huntington mutant', filtrando por especie.

Si, además, quisiéramos introducir un filtrado por fechas, para sólo ver los artículos más modernos, sólo sería necesario escribir:

```
huntington mutant AND "humans" [MeSH Terms]
AND ("2012/01/01" [PDAT] : "3000/12/31" [PDAT])
```

reduciéndose la búsqueda a tan sólo 152 artículos. Al consultar un artículo en concreto, nos llevará a una página como la de la Figura 2.4, donde aparecen el título y resumen del artículo, y marcadas en estos textos las palabras clave usadas en la búsqueda. Adicionalmente, también aparecen el *enlace al artículo original* en la revista como a la posible copia del mismo en PubMed Central (si estuviera), enlaces a artículos relacionados, y en cuadros desplegables las anotaciones de *MeSH terms* del artículo. Todas las búsquedas en PubMed se pueden grabar en la sesión actual, para comparar los resultados,

así como *exportar parte o todos los resultados* a ficheros en diversos formatos, incluidos muchos de los manejados por gestores de bibliografía.

**Figura 2.4:** Ejemplo del resumen de un artículo dentro de PubMed.

### 2.2.3. Google Scholar

Una forma cada vez más popular de buscar artículos científicos es mediante *Google Scholar*<sup>7</sup>. Este sistema está basado en la base de datos y motor de búsquedas de Google, de manera que problemas como el de los sinónimos o las erratas en los textos o los términos de búsqueda quedan más o menos mitigados. Pero a diferencia de PubMed, *Google Scholar* integra no sólo artículos científicos, sino también tesis doctorales, libros, textos legales (como por ejemplo, patentes), resúmenes e introducciones para congresos y ponencias. Además, la fuente de estos documentos no son sólo las revistas y editoriales científicas, sino también universidades o asociaciones profesionales.

La forma de ponderar los documentos encontrados tiene en cuenta factores como el contenido completo del texto, dónde fue publicado, por quién fue escrito y la frecuencia y últimas veces que fueron referenciados desde otros textos. El sistema además proporciona para la publicación en la que aparece cada artículo una serie de métricas:

- El *h-index* de una publicación es el mayor número *h* para el cuál al menos *h* artículos de la publicación fueron citados al menos *h* veces cada uno.
- El *h-core* de una publicación es el conjunto artículos que cumplen la condición del *h-index* calculado.
- El *h-median* de una publicación es la mediana del número de citas de los artículos incluidos en el *h-core*.

<sup>7</sup>Google Scholar. <http://scholar.google.es>

En el escenario de un número de la revista ‘ArTi’, que tenga 7 artículos, que han sido referenciados 19, 10, 5, 2, 1, 0 y 0 veces desde su publicación hasta el momento actual, el *h-index* de ese número de la revista ArTi sería 3 y el *h-core* estaría compuesto por los artículos que han sido citados 19, 10 y 5 veces. Al ser la media de citas de los artículos en el *h-core* de 11.333..., el *h-median* sería 10.

En lugar de estas métricas, en los resultados de las búsquedas Google Scholar muestra las métricas *h5-index* y *h5-median*. Su cálculo es idéntico al de las métricas anteriormente explicadas, pero teniendo en cuenta sólo los artículos publicados en los pasados 5 años, pudiéndose inspeccionar el *h5-core*. La precisión de los valores de todas estas métricas se ven influidos por la calidad y la precisión a la hora de identificar correctamente las referencias a una revista o artículo.

The screenshot shows a Google Scholar search results page for the query "huntington mutant". The search bar contains the query. On the left, there's a sidebar with filters: "Articles", "Case law", "My library New!", and a date range selector. The date range selector is circled in red and includes options like "Any time", "Since 2013", "Since 2012", "Since 2009", and "Custom range...". Below this are "Sort by relevance" and "Sort by date" buttons, followed by checkboxes for "include patents" and "include citations", both of which are checked. A "Create alert" button is also present. The main results list includes several entries, such as "Ganglioside GM1 induces phosphorylation of mutant huntingtin and restores normal motor behavior in [HTML] from pnas.org" and "Neurodegenerative disease: Preventing SIRTain'death by mutant huntingtin". Each result has a link to the full text, citation counts, and "Cite Save" buttons. The results are paginated at the bottom.

**Figura 2.5:** Búsqueda en Google Scholar, filtrando por fecha.

Como se puede ver en la Figura 2.5, en Google Scholar se pueden realizar búsquedas similares a las que se pueden hacer en PubMed, pudiendo filtrarse las búsquedas por palabras clave en el título o a determinados autores. Sin embargo, tiene algunas limitaciones, como que *no se pueda filtrar fácilmente por organismo*. Se pueden *grabar y exportar referencias* a formatos BibTeX o EndNote, usados por casi todos los gestores de referencias bibliográficas.

Google Scholar está *integrado con el sistema de cuentas de usuario de Google*, de manera que se pueden *grabar las búsquedas* más interesantes, e incluso usarlas como avisadoras para *configurar alertas* cada vez que aparezcan documentos nuevos relativos a dichas búsquedas. Así mismo, Google Scholar se puede usar para *gestionar los artículos de los que sea autor o coautor*, y poder ver la evolución del impacto de dichos artículos, o incluso recibir un aviso cada vez que un artículo tuyo haya sido citado en nuevos artículos aparecidos.

Una forma muy sencilla de incluir en Google Scholar un documento científico que no se encuentre ya catalogado y almacenado en otra parte, como por ejemplo, una tesis doctoral, es poniendo dicho documento accesible desde la red, en formato PDF y enlazado desde una página web. Salvo que la página que referencia a ese documento esté inaccesible, los motores de actualización de contenidos

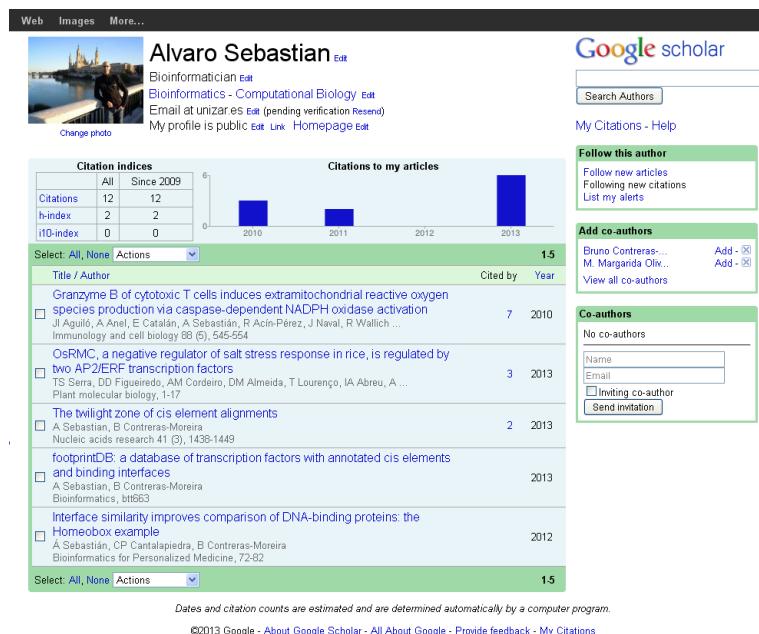
de Google la terminarán encontrando, encontrarán el PDF y se encargarán de indexar su contenido. Para publicar un gran volumen documentos científicos es recomendable usar un repositorio digital local basado en EPrints<sup>8</sup> o DSpace<sup>9</sup>.

## Perfil en Google Scholar

Google Scholar permite crear nuestro *perfil personal* donde mostrar y gestionar nuestras propias publicaciones. Para crear el perfil deberemos entrar en nuestra cuenta Google y acceder a la página del perfil<sup>10</sup>. Tras completar nuestros datos, afiliación y e-mail académico (opcional), podremos añadir automáticamente a nuestro perfil las publicaciones indexadas por Google Scholar y manualmente aquellas que no lo estén.

Adicionalmente podremos *añadir coautores* a nuestras publicaciones, *crear alertas* para saber cuándo nuestras publicaciones tienen citas nuevas o alguien publica algo que nos interesa. También podremos seguir las publicaciones de otros investigadores y recibir un correo cada vez que publiquen algo nuevo. Finalmente nos permitirá consultar las *métricas de nuestros artículos*, como el número de citas, *h-index* e *i10-index*<sup>11</sup>.

Nuestro *perfil* podrá ser *público* o *privado*, según deseemos. En la Figura 2.6 se muestra cómo es el perfil público de uno de los autores del capítulo.



**Figura 2.6:** Ejemplo de perfil público en Google Scholar.

<sup>8</sup>EPrints. <http://www.eprints.org>

<sup>9</sup>DSpace. <http://dspace.org>

<sup>10</sup>Perfil en Google Scholar. <http://scholar.google.com/citations?hl=es>

<sup>11</sup>I10-index. <http://en.wikipedia.org/wiki/I10-index>

## 2.3. Índices y medidas de impacto científico

El *impacto de un artículo* se puede medir de varias maneras: por el número de lectores del artículo (que es difícil de calcular), por el número de artículos que lo referencian, y por el índice de impacto que tenía en el momento de la publicación la revista donde fue publicado. El índice de impacto de una revista intenta medir cómo de popular es dicha revista, y cuánto caso se le hace dentro de la comunidad científica.

Un índice de impacto ampliamente admitido es el calculado por la empresa Thomson Reuters en sus *Journal Citation Reports* (JCR). Tal como se describe en su sitio web<sup>12</sup>, el *índice de impacto* de una revista en un determinado período es una medida de la frecuencia con la que un “artículo típico” de dicha revista ha sido citado en ese período. El índice de impacto JCR anual es la razón entre publicaciones y entradas referenciables publicadas recientemente. Por ello el índice de impacto de una revista para un determinado año se calcula dividiendo el número artículos en todas las revistas publicados ese año que referencian a artículos de la revista publicados en los dos años anteriores, entre el número de artículos publicados en esa revista en los dos años anteriores al año que es evaluado.

Sin embargo, no todos los artículos son tenidos en cuenta a la hora de calcular este índice de impacto, ya que las editoriales pueden decidir qué artículos entran o no en el cómputo. Por ello, a pesar de que este índice de impacto es tenido en cuenta por muchas revistas e instituciones científicas a la hora de proclamar cómo de buena es una publicación, han ido surgiendo cada vez más críticas hacia el mismo, por un lado por las tácticas que se pueden seguir para influir en ese índice de impacto, y por otro porque en varias ocasiones ha sido imposible por parte de grupos independientes reproducir dichos números calculados.

Una forma de medir el impacto inmediato de un artículo es usando los correspondientes sistema de alertas de las principales revistas científicas, que suelen informar cuándo uno de sus artículos está teniendo un índice de impacto superior a la media. Otra manera es mediante el uso de medios más afines a las redes sociales, como Twitter o Research Gate<sup>13</sup>.

## 2.4. Gestores de bibliografía

Un *gestor de bibliografía* es una base de datos de artículos y materiales científicos que contiene, al menos, la referencia a cada documento registrado (el título, los autores, fecha de publicación, revista y número en los que fue publicado, resumen, palabras clave...), y todo el material adicional relacionado con ese documento registrado (el texto completo del artículo, los materiales suplementarios, las anotaciones que estemos realizando sobre el mismo, etc.).

Los gestores de bibliografía nacieron de la necesidad de mantener un registro de qué artículos habían sido leídos a la hora de elaborar un artículo o de escribir un libro, un informe o una tesis. Una de las tareas de los gestores de bibliografía es *exportar las referencias bibliográficas* que seleccionemos siguiendo el formato requerido por la revista a la que vayamos a enviar un artículo. Por ejemplo, la referencia [2] está formateada para las revistas Nature, Science y Nucleic Acid Research:

- Rubio-Camarillo, M., Gómez-López, G., Fernández, J. M., Valencia, A. & Pisano, D. G. RUBioSeq: a suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses. *Bioinformatics* (2013).
- M. Rubio-Camarillo, G. Gómez-López, J. M. Fernández, A. Valencia, D. G. Pisano, RUBioSeq: a suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses, *Bioinformatics* (2013).

<sup>12</sup>The Thomson Reuters Impact Factor. <http://wokinfo.com/essays/impact-factor>

<sup>13</sup>Research Gate. <http://www.researchgate.net>

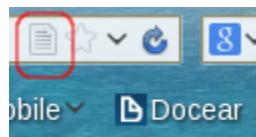
- Rubio-Camarillo,M., Gómez-López,G., Fernández,J.M., Valencia,A. and Pisano,D.G. (2013) RUbioSeq: a suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses. Bioinformatics.

Usando un gestor de bibliografía, formatear las referencias de forma adecuada para una revista en un manuscrito en el que estemos citando 50 artículos diferentes es una tarea trivial. Los gestores de bibliografía actuales, como por ejemplo Zotero, Mendeley o EndNote, se integran con los sistemas de elaboración de textos más empleados a nivel científico, como son LaTeX, Microsoft Word y LibreOffice/OpenOffice. Por eso, en caso de que por alguna razón decidamos mandar dicho manuscrito a otra revista, el reformateo es inmediato.

Los estándares *de facto* usados para el intercambio de referencias bibliográficas son, entre otros, BibTeX (ligado al sistema LaTeX), EndNote, RIS y MODS. Por ello, todos los gestores de bibliografía, ya sean locales o web, permiten importar y exportar las referencias a uno o varios de estos formatos. Algunos de los gestores bibliográficos más usados en ciencia son Docear<sup>14</sup>, Papers<sup>15</sup>, EndNote<sup>16</sup>, CiteULike<sup>17</sup>, Zotero<sup>18</sup> y Mendeley<sup>19</sup>.

#### 2.4.1. Zotero

Zotero<sup>18</sup> es un gestor de bibliografía gratuito mantenido en el *Roy Rosenzweig Center for History and New Media*, que nació inicialmente como una extensión del navegador Firefox, y que actualmente se puede usar tanto en varios navegadores (instalando una extensión en cada navegador) como de forma independiente del navegador. Su primer objetivo siempre ha sido facilitar lo más posible la captura durante el proceso de documentación de referencias a material científico disponible en la red. Por ello, como se puede ver en la Figura 2.7, Zotero muestra un ícono característico cada vez que se visita una página web que pueda contener material referenciable, como un artículo científico o un conjunto de ellos encontrados en PubMed, Google Scholar o la web de cualquier revista científica, la referencia a uno o más libros encontrados en bibliotecas electrónicas o sitios de compra de libros, etc.



**Figura 2.7:** Ícono de Zotero en la barra de URLs, indicando que hay información sobre un artículo en esa página.

Al pinchar en ese ícono, Zotero añade automáticamente los datos de la referencia bibliográfica a su biblioteca local (Figura 2.8). Esa referencia se almacena en la carpeta que esté activa de la biblioteca, y lleva asociada todo el material adicional que se encuentre referenciado en la página. Por ejemplo, si tenemos suscripción a una revista, buscamos un artículo en concreto y pinchamos en ese ícono, se añaden no sólo los datos de la referencia y un *snapshot* de la página HTML, sino también el PDF con el artículo completo (aunque no lo hace automáticamente con el material suplementario). Todo el contenido de los materiales añadidos a Zotero es indexado, de manera que se pueda localizar cualquier referencia por el contenido de sus documentos.

<sup>14</sup>Docear. <https://www.docear.org>

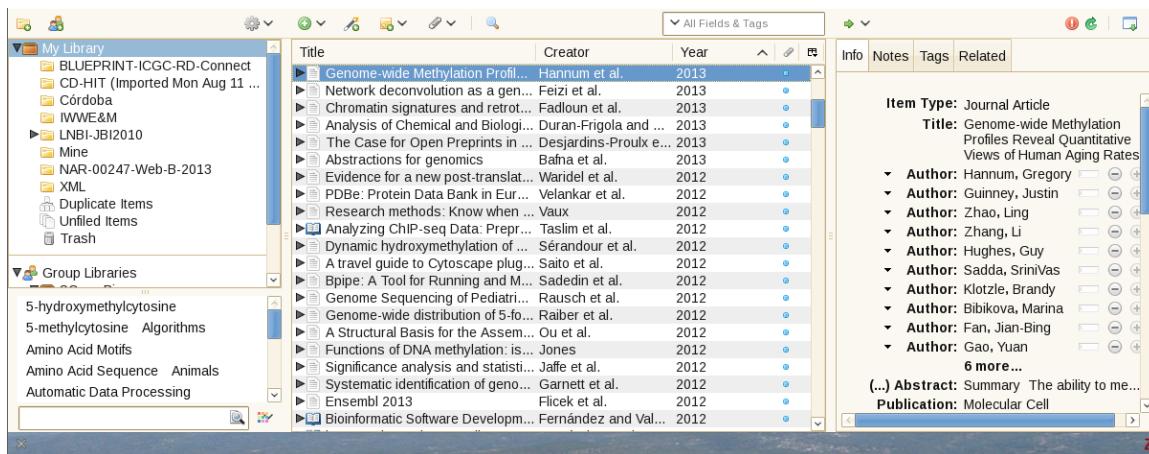
<sup>15</sup>Papers. <http://www.papersapp.com>

<sup>16</sup>EndNote. <http://endnote.com>

<sup>17</sup>CiteULike. <http://www.citeulike.org>

<sup>18</sup>Zotero. <http://www.zotero.org>

<sup>19</sup>Mendeley. <http://www.mendeley.org>



**Figura 2.8:** Pestaña principal de Zotero en Firefox.

Dentro del sitio web de Zotero se puede crear una cuenta gratuita, que permite, entre otras cosas, la creación de grupos, y que varios usuarios de Zotero sean capaces de compartir y gestionar una biblioteca de grupo. Otro uso de la cuenta es permitir que todas las instalaciones de Zotero vinculadas a esa cuenta, que muy probablemente estén en distintos dispositivos, puedan sincronizar sus bibliotecas. Hay dos niveles de sincronización disponibles, correspondiendo el nivel básico a las referencias bibliográficas. La cuenta gratuita proporciona una cuota de 300MB para el nivel completo de sincronización: documentos PDF, *snapshots* HTML, imágenes, ... Pero al ser tan sencillo alcanzar esa cuota, por un lado se puede pagar anualmente una cantidad de dinero para ampliar dicho espacio, o bien usar un servidor WebDAV (ya sea propio o público) para almacenar esos documentos.

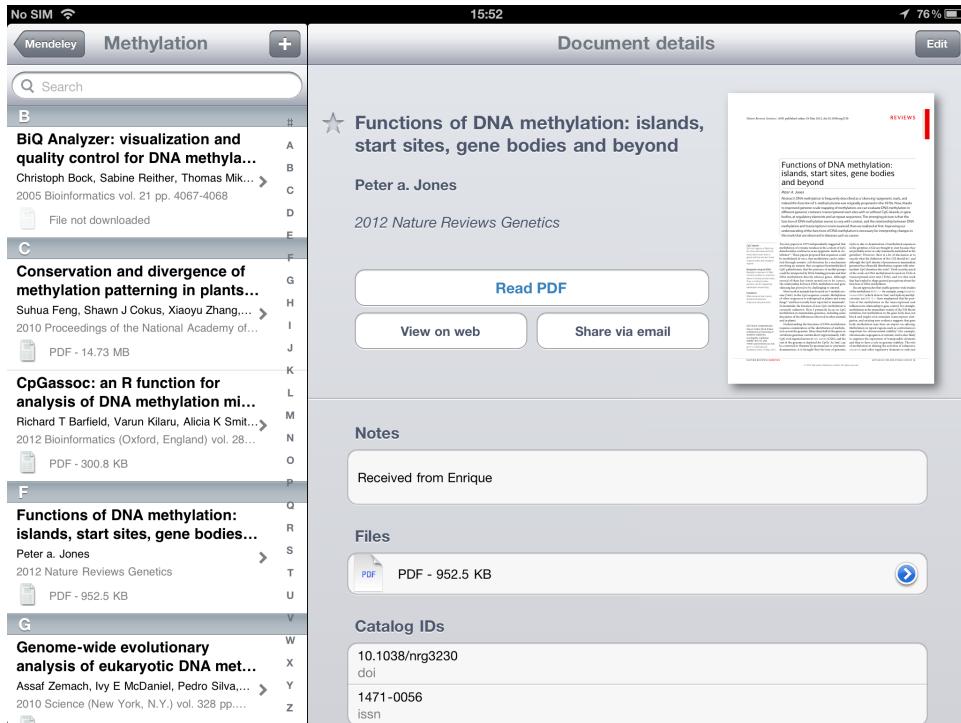
Zotero permite tanto importar como exportar referencias en bloque usando los formatos abiertos más comunes (BibTeX, RIS, MODS, etc.). También tiene asociados una serie de extensiones para Microsoft Word y LibreOffice/OpenOffice, permitiendo integrarse con ellos y gestionar las referencias que aparezcan en un manuscrito que se encuentre en elaboración, así como el formato de dichas referencias. Zotero también tiene la capacidad de elaborar *timelines* e informes de un conjunto de artículos seleccionados, lo que permite observar detalles como las interrelaciones entre ellos a nivel de citaciones, qué palabras clave comparten, etc.

El servidor de Zotero usa un protocolo abierto y documentado, con lo que cualquiera puede crear aplicaciones capaces de sincronizarse con los servidores de Zotero. Actualmente existen varios clientes de Zotero para distintas plataformas, como *Zandy* y *Scanner for Zotero* para Android, y *BibUp*, *PaperShift* o *ZotPad* (Figura 2.9) para iOS.

#### 2.4.2. Mendeley

*Mendeley*<sup>19</sup> (Figura 2.10) es una aplicación gratuita de gestión de material bibliográfico, cuya virtud más destacada es la *capacidad de extraer* de los documentos (ya sea PDF o documento Word) de artículos científicos los *metadatos* necesarios para identificarlos. La manera principal de trabajar con Mendeley es copiar el PDF de un manuscrito en una carpeta de documentos configurada dentro de Mendeley, iniciándose de esa manera el proceso de extracción de metadatos. Con esos metadatos consulta las bases de datos bibliográficas públicas, como por ejemplo PubMed o DOI<sup>20</sup> para luego complementar

<sup>19</sup>International Digital Object Identifier Foundation. <http://www.doi.org>



**Figura 2.9:** ZotPad, un cliente de Zotero para iPad.

la referencia bibliográfica asociada a dicho documento, que se añade a la biblioteca local. Todo el contenido de los documentos añadidos que reconozca Mendeley se indexa, para facilitar posteriores búsquedas por contenido o por palabra clave. Dispone de un *visor integrado de PDFs* que permite no sólo ver, sino anotar dichos documentos, quedando también esas notas registradas.

En el sitio web de Mendeley se puede crear una cuenta gratuita que permite almacenar las referencias bibliográficas de una instalación local, y con espacio de 2GB para los documentos asociados. Mediante esa cuenta se pueden sincronizar dichas referencias y documentos con otros dispositivos, como por ejemplo un tablet (Figura 2.11). Otra manera de añadir nuevas referencias bibliográficas es mediante el importador web, que es un *bookmark* especial para navegadores que envía la página actual al servidor de Mendeley, para que añada a tu biblioteca las referencias bibliográficas que encuentre.

Al igual que muchos otros gestores de bibliografía, permite tanto importar como exportar parte o todas las referencias a formatos estándar, como BibTeX, EndNote o RIS. Un punto adicional que tiene Mendeley es que *se puede sincronizar con una instalación local de Zotero*, tanto a nivel de referencias como de etiquetas, carpetas y documentos, de forma que se puede trabajar con ambos gestores de bibliografía de forma simultánea. De la misma manera, también hay *plugins* para Microsoft Word y LibreOffice/OpenOffice que permiten integrar Mendeley dentro del proceso de escritura y gestión de referencias de un manuscrito.

La cuenta de usuario en Mendeley también tiene *funcionalidades de redes sociales*, ya que permite buscar artículos, anotaciones e incluso documentos completos por temáticas en las bibliotecas públicas de referencias de otros usuarios de Mendeley. Dentro de Mendeley también existe el concepto de *grupos y equipos*, y de *bibliotecas* de referencias asociadas a ellos, de manera que cada usuario del grupo puede contribuir con referencias y documentos a dicha biblioteca. Sin embargo, para las cuentas gratuitas el número de grupos que se pueden crear está limitado a 1.

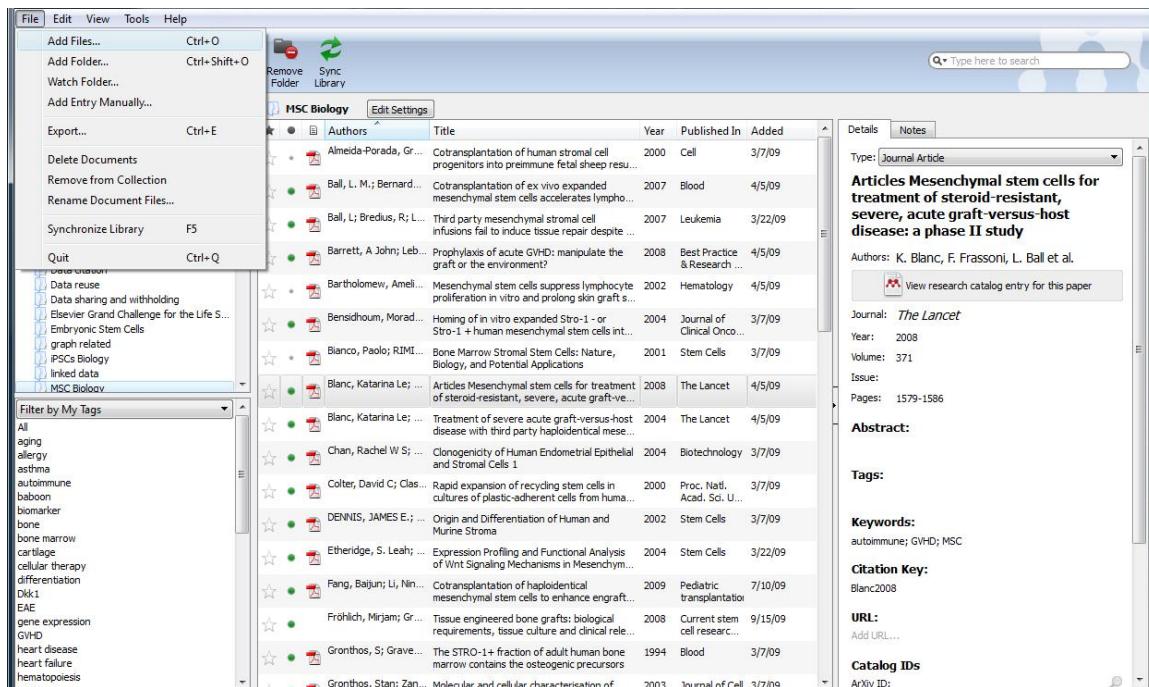


Figura 2.10: Mendeley Desktop, disponible para Windows, Linux y Mac OS X.

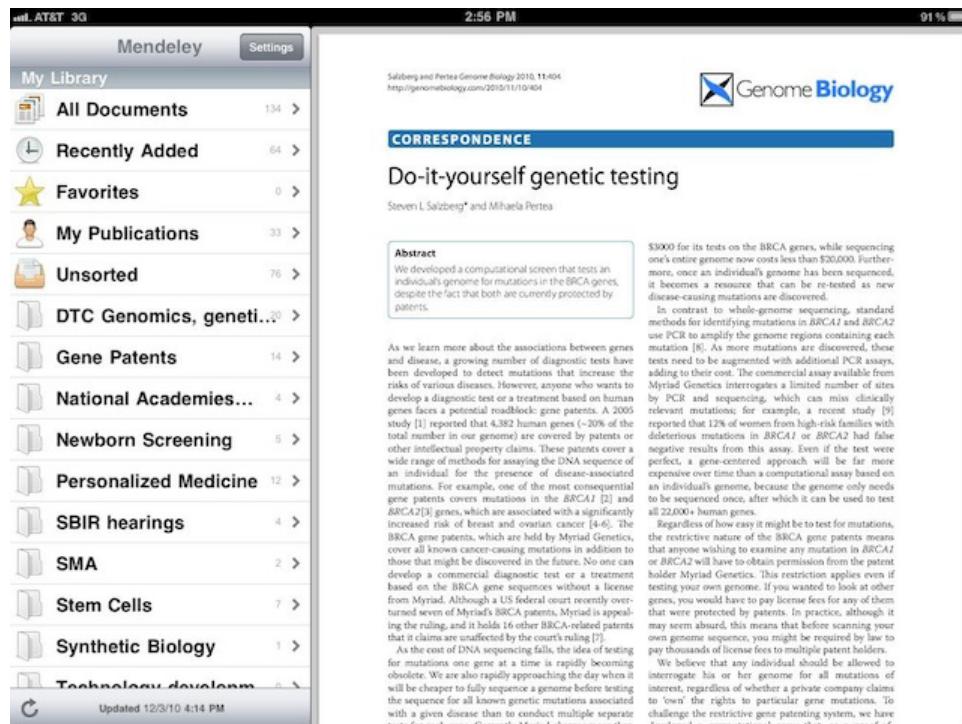


Figura 2.11: Cliente de Mendeley para iPad.

Por último, toda la información almacenada dentro del perfil de Mendeley puede ser accedida mediante un protocolo abierto y documentado, lo que permite que se puedan desarrollar aplicaciones no oficiales capaces de consultar y gestionar las bibliotecas de referencias mantenidas en el perfil, así como sus

documentos.

### 2.4.3. Biblioteca de Google Scholar

El servicio ‘*Mi Biblioteca*’<sup>21</sup> de Google Scholar ha sido implementado recientemente. Consiste en una colección personal de artículos vinculada a nuestra cuenta Google. Los artículos indexados en Google Scholar pueden ser añadidos a nuestra biblioteca con un simple clic en la opción ‘Guardar’ bajo el enlace al artículo (ver Figura 2.12). Nuestra biblioteca de artículos puede ser *organizada por tema* y se puede aprovechar el poder del *motor de búsqueda de Google* para encontrar fácilmente el artículo que nos interese en nuestra colección.

The screenshot shows a Google Scholar search results page. The search term is 'database transcription factors'. The results are filtered to 'Académico' (Academic). There are approximately 644,000 results. The first result is 'JASPAR: an open-access database for eukaryotic transcription factor binding profiles' by A Sandelin et al. The abstract and citation information are shown. Below the abstract, there are several options: 'Artículos', 'Mi biblioteca', 'Nuevo', 'Cualquier momento', 'Desde 2014', 'Desde 2013', 'Desde 2010', 'Intervalo específico', 'Ordenar por relevancia', 'Ordenar por fecha', 'Buscar en la Web', and 'Buscar sólo páginas en español'. The 'Guardar' button next to the citation information is circled in red.

Figura 2.12: Opción ‘Guardar’ bajo una búsqueda con Google Scholar.

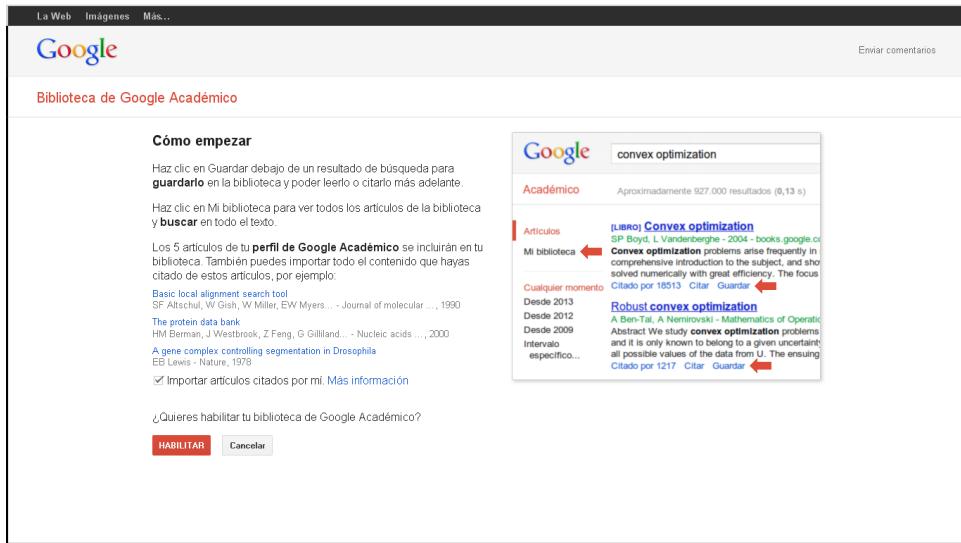
Para comenzar a usar nuestra biblioteca Google Scholar, deberemos acceder al servicio de biblioteca de Google Scholar<sup>22</sup> y pulsar sobre el botón ‘Habilitar’ en la pantalla de bienvenida (ver Figura 2.13). Google automáticamente añadirá a nuestra biblioteca nuestros artículos y todos los que hemos citado.

Dentro de ‘Mi biblioteca’ están las acciones de *etiquetar* y *clasificar* los artículos guardados, *visionar por fecha*, *consultar las referencias* que citan alguno de mis artículos y ver qué referencias he citado en los mismos. Los artículos de la biblioteca no aparecerán en mi perfil público (ver Sección 2.2.3).

Además al pulsar sobre cada artículo de la biblioteca tendremos las mismas opciones que nos daba Google Scholar de visitar la fuente original del artículo, exportar la referencia al artículo en varios formatos (BibTeX, EndNote, RefMan y CSV), así como de consultar en qué otros artículos ha sido citado.

<sup>21</sup>Google Scholar library help page. <http://scholar.google.com/intl/en/scholar/help.html#library>

<sup>22</sup>Servicio de biblioteca de Google Scholar. <http://scholar.google.com/scholar?hl=en&scilib=1>



**Figura 2.13:** Habilitar por primera vez nuestra biblioteca en Google Scholar.

## 2.5. Escritura de artículos

A la hora de preparar un manuscrito para su publicación, hay que tener muy claras cuáles son las ideas y los resultados que se quieren presentar. Un artículo científico suele estar compuesto por el *título*, los *autores* y sus afiliaciones, las *palabras clave* que lo definen, su *resumen*, y las siguientes secciones: *introducción*, *resultados*, *materiales y métodos*, *discusión*, *agradecimientos* y patrocinios, y *referencias o bibliografía*. A la hora de preparar un manuscrito se suelen incluir al final o en un documento separado todas las tablas y figuras en baja resolución junto con sus pies correspondientes. Finalmente se han de adjuntar en ficheros separados versiones en alta resolución (normalmente, 600 puntos por pulgada) o vectoriales de cada una de las figuras.

El manuscrito tiene que estar estructurado en torno a las ideas y resultados que se presentan. Se debe seguir una *línea argumental* lo más clara y concisa posible, explicando los conceptos que sean necesarios para comprender el manuscrito, ya sea directamente o mediante el apoyo en referencias a bibliografía. El texto del manuscrito tiene que ser correcto y conciso, libre de erratas y ambigüedades. Todas las figuras, tablas y citas a otros artículos deben estar justificadas por el hilo argumental del manuscrito, y aparecer referenciadas en el texto. Por ello, se debe evitar todo lo posible la inclusión o referencia a material superfluo. El idioma internacional de la ciencia es desde hace muchos años el *inglés*, y a la hora de redactar el manuscrito hay que tomar consideración si la editorial de la revista donde se intenta publicar prefiere el inglés americano o el inglés británico.

Las personas que hayan colaborado en un artículo deben aparecer como *autores* o en la sección de agradecimientos. Cuantos menos autores aparezcan en un manuscrito, más relevante es el orden en el que aparecen, y en algunos casos puede ser fuente de graves disputas. Como normal general deberán aparecer en la lista de autores todos los investigadores que hayan realizado algún experimento o trabajo para el artículo y en la lista de agradecimientos las personas o entidades que hayan colaborado asesorando, revisando o aportando medios para la realización de los trabajos. En las primeras posiciones deben figurar los autores que más trabajo han hecho para el artículo, mientras que en las posiciones finales suelen aparecer los jefes de grupo y directores de proyecto. Existe además la figura del *corresponding author*, que puede coincidir o no con el primer autor, y que es quien ha coordinado o supervisado la investigación que ha llevado a la escritura del manuscrito. Hasta hace no mucho existía la costumbre de

incluir sin razón a jefes de departamento o catedráticos en las posiciones finales como autores, a veces sin que dichas personas se hubieran leído el artículo ni hubieran participado en él. En cambio, se excluía a estudiantes que habían hecho experimentos y deberían ser legítimos autores, pero que sin embargo nadie se acordaba de incluirlos al haber dejado el laboratorio antes de la elaboración del manuscrito.

La preparación de un manuscrito suele requerir de la coordinación por parte de uno de los autores, normalmente el *corresponding author*, que es el que se debe encargar de mantener la copia maestra del manuscrito. Este coordinador recibe todos los cambios y comentarios sobre el manuscrito por parte de los autores, debe decidir qué cambios se integran en la copia maestra del manuscrito y debe distribuir entre los autores la nueva versión de la copia maestra una vez hechos los cambios, y las tareas de reescritura que les podría corresponder. *Herramientas colaborativas de escritura de documentos*, como Google Docs, o *herramientas de gestión de cambios* en Word o LibreOffice/OpenOffice permiten agilizar el proceso de escritura de los artículos, y facilitar el trabajo del coordinador.

### 2.5.1. Formato y plantillas de artículos

Normalmente se prepara inicialmente un *artículo genérico* que resume la investigación en unos apartados estándar como los anteriormente mencionados: introducción, materiales y métodos, resultados y discusión. Entonces se decide entre todos los autores a qué revista enviarlo de acuerdo a su temática, calidad y relevancia. La revista de destino restringe las temáticas que puede seguir el manuscrito, para que todos artículos publicados vayan en consonancia con la *línea editorial de la revista*. Una vez seleccionada la revista, habrá que adaptar el artículo a las normas de la misma.

La revista de destino para el manuscrito concreta una serie de *requisitos técnicos* que debe seguir todo artículo que sea enviado a los editores<sup>23</sup><sup>24</sup> (Figura 2.14). Estos requisitos deben ser cumplidos al pie de la letra, porque son un primer filtro a la hora de rechazar la admisión de artículos. Los parámetros suelen ser: la longitud máxima en palabras del resumen y en páginas del artículo; las secciones que deben aparecer en el manuscrito, y su denominación y orden; el formato y el orden de las referencias bibliográficas y sus citas desde el texto; el formato del manuscrito (documento Word, RTF, PDF, LaTeX...); el formato de las figuras (TIFF, PNG, JPEG, PDF), su resolución mínima en puntos por pulgada y sus dimensiones máximas, etc. Por todo ello habrá que adaptar el artículo genérico escrito previamente para que cumpla las reglas de la revista. El cambio de formato en las referencias se podrá realizar fácilmente con ayuda de los gestores bibliográficos explicados en la Sección 2.4.

Muchos grupos editoriales suelen proporcionar a los autores de manuscritos una serie de *plantillas* que facilitan la tarea de preparación de un manuscrito para su envío a los editores. Dependiendo de la revista, esas plantillas son para documentos Word o para LaTeX, y especifican el tamaño, la disposición y el nombre de cada una de las secciones del manuscrito. Así mismo, esas plantillas especifican los márgenes, espaciado, interlineado, fuente a usar y tamaño de la fuente en las distintas secciones. La primera versión del artículo se puede enviar en formato PDF sin usar las plantillas requeridas por la revista, pero si el artículo es aceptado para publicación habrá que formatearlo y enviarlo usando dichas plantillas.

<sup>23</sup>Instructions to Authors for *Bioinformatics* journal.

[http://www.oxfordjournals.org/our\\_journals/bioinformatics/for\\_authors/general.html](http://www.oxfordjournals.org/our_journals/bioinformatics/for_authors/general.html)

<sup>24</sup>Instructions to Authors for *Nucleic Acids Research* journal.

[http://www.oxfordjournals.org/our\\_journals/nar/for\\_authors/msprep\\_submission.html](http://www.oxfordjournals.org/our_journals/nar/for_authors/msprep_submission.html)

#### INITIAL SUBMISSION ONLINE

Prepare your manuscript as detailed above and then submit online via the web site <http://mc.manuscriptcentral.com/nar>.

##### CHECKLIST prior to initial submission:

You must ensure that you have available the following files. Acceptable file formats are listed above. Please ensure that all files are named carefully and unambiguously so that their content is clear:

- A file of your complete manuscript text including title page and abstract
- Your high-resolution figure files containing your figures, schemes, equations and, if complex, tables. Please make sure that each figure etc is clearly labelled with its number
- Any supplementary data which MUST be in a file(s) separate from the main manuscript file
- A file of any related manuscript currently under consideration by another journal (see Author responsibilities)
- Your manuscript title and abstract text for cutting and pasting into the system
- The email addresses of all of your co-authors [Please note that the journal reserves the right to contact the Senior Author of the manuscript if his/her contact details are not included.]
- Names, institutes and email addresses of at least four suggested referees [Please note that four names are now mandatory for standard category and methods manuscripts, but six are preferred. Submissions for the special issues (Database and Web Server issues) must supply six names. They should be scientists working independently (i.e. not a recent collaborator) in areas similar to your own who have relevant expertise, such as those included in your reference list. If you have any queries please contact the Senior Editorial Office.]
- Your letter to the editor, either as a file to upload or for cutting and pasting into the system. You may upload more than one such file, so you may include supporting material that is not for publication. The letter must contain details of any previous submissions of the work to NAR (see Author responsibilities)
- An estimate of approximate final paper length using the equation provided above under Length
- Between two and five keywords or short relevant phrases

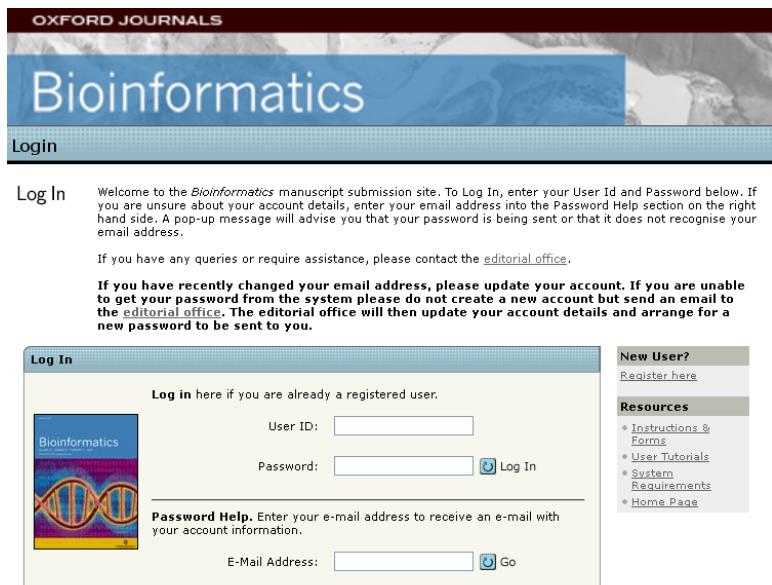
**Figura 2.14:** Detalle de la información *online* para autores de la revista *Nucleic Acids Research*.

### 2.5.2. Proceso de envío y revisión de artículos

Cada revista o grupo editor tiene su propio *sistema de envío y revisión* de artículos. La URL de acceso a dicho sistema se encuentra en las instrucciones para los autores de la revista (Figura 2.15). El primer paso para enviar un artículo es encontrar el enlace a dicho sistema y crear una cuenta de usuario introduciendo nuestros datos personales y de nuestro grupo de investigación. Una vez autenticados en el sistema, podremos enviar un nuevo artículo llenando un completo formulario (Figura 2.16), donde deberemos insertar datos sobre los autores y afiliación, título y resumen del artículo, palabras clave, temática... Además, dependiendo de la revista, deberemos sugerir y dar datos de contacto de varios posibles revisores del artículo (normalmente entre 3 y 5), éstos deberán ser investigadores que trabajen en la misma temática del artículo, que no hayan intervenido ni colaborado en nuestra investigación. Finalmente, se deberán adjuntar los archivos con el texto del artículo, figuras, tablas y materiales suplementarios. Para una primera revisión del artículo se podrán incluir las figuras y tablas en el texto del artículo sin ser necesario usar las plantillas de la revista, pero sí respetando las reglas respecto a contenido, secciones y tamaño.

Tras enviar el artículo deberemos esperar la respuesta de un *editor* que examinará el artículo, verá si cumple los requisitos técnicos básicos (estructura, longitud máxima, etc.), decidirá si su temática se corresponde a la de la revista y valorará su impacto potencial en la disciplina científica de la revista. De acuerdo a la importancia y calidad de la revista, el editor seleccionará los mejores artículos recibidos para su publicación y desechará el resto. El filtro del editor es el primer obstáculo en la publicación de un artículo, normalmente el editor nos mandará un email con su decisión en un plazo de entre 7 y 30 días dependiendo de la revista.

Si el artículo pasa el filtro del editor, éste enviará el artículo a los *revisores* que elija, ya sean los sugeridos durante el envío del artículo, u otros que estime más efectivos. El sistema de revisión de artículos científicos se denomina '*revisión por pares*' (*peer review* en inglés). Éste consiste en que dos o más revisores leen y analizan los artículos de forma independiente para determinar la validez e importancia de la investigación y los resultados. Los evaluadores rara vez reciben reconocimiento



**Figura 2.15:** Portal de acceso para revisores y autores de la revista *Bioinformatics*.

económico, aunque sí cierto prestigio y acceso privilegiado a información. El proceso de revisión por pares suele durar entre 20-40 días y es el momento más temido por todo científico. Tras la revisión, ya sea positiva o negativa, por parte de los revisores, el editor pondera esas evaluaciones y tiene que tomar la decisión de si el artículo tiene que ser revisado, si es aceptado o si es rechazado. En caso de ser revisado puede requerir una ‘*major revision*’ o ‘*minor revision*’. En el primer caso los revisores y/o el editor pedirán realizar cambios sustanciales en el artículo y/o nuevos experimentos antes de volver a revisar el artículo y decidir sobre su aceptación. En el segundo caso los revisores sugerirán una serie de cambios menores y mejoras al artículo, tras los cuales estará listo para ser publicado sin ninguna revisión adicional. En ambos casos habrá que realizar las modificaciones oportunas (marcándolas normalmente con otro color de letra), reenviar el artículo como una ‘nueva revisión’ y esperar el veredicto de los revisores nuevamente (en caso de ser necesario). En la Figura 2.17 hay un ejemplo del proceso de revisión de un artículo.

Cuando el artículo es definitivamente aceptado, pasará a la etapa de ‘*producción*’. Se deberán aceptar las condiciones de licencia y publicación de la revista por parte de todos los autores y el pago de las tarifas de publicación, común en las revistas *open access* y en aquellas que exigen una tasa adicional por figuras en color o sobrepasarse de la longitud máxima del artículo. Habrá que adaptar el texto a las plantillas proporcionadas por la revista, si no se ha hecho previamente, y corregir pequeñas erratas y fallos que detecte el equipo de producción de la revista. Finalmente el artículo será publicado, pero si la revista tiene edición en papel, el artículo puede tardar 1 o 2 meses más en ser publicado definitivamente aunque exista una versión ‘*advance access online*’.

El tiempo total entre el primer envío y publicación de un artículo puede variar mucho, un tiempo mínimo sería de 2 meses, aunque la media oscila entre 5-6 meses y algunos artículos pueden tardar incluso años, especialmente cuando el artículo es rechazado por varias revistas y hay que reenviarlo, adaptarlo o incluir nuevos experimentos para darle más valor.

Author Center  
Submit a Manuscript

Select your manuscript type and subject category. Enter your title and abstract into the appropriate boxes below. The running heading will appear as the header of the published paper. This heading should not exceed fifty characters, including spaces. If you need to insert a special character, click the "Special Characters" button. When you are finished, click "Save and Continue." [Read More ...](#)

**1 Type, Title & Abstract**

**2 Attributes**

**3 Authors & Institutions** ✓

**4 Reviewers**

**5 Details & Comments**

**6 File Upload**

**7 Review & Submit**

**Manuscript Type**

Category

**Title** (Limit 512 characters)

Press Control-V (or Cmd-V) to Paste

**Abstract** (Limit 250 words)

Press Control-V (or Cmd-V) to Paste

**Subject Category**

Please select your manuscript subject category:

**Does your manuscript involve Next Generation Sequencing data analysis?**

Yes  No

**Figura 2.16:** Formulario de envío de artículos de la revista *Bioinformatics*.

Manuscripts with Decisions						
Manuscript ID	Manuscript Title	Date Submitted	Date Decisioned	Status	Actions	
BIOINF-2013-1569.R1	footprintDB: a database of transcription factors with annotated cis elements and binding interfaces <a href="#">[View Submission]</a>	22-Oct-2013	08-Nov-2013	<ul style="list-style-type: none"> <li>• Accept after Review (08-Nov-2013)</li> </ul> <a href="#">view decision letter</a>		
BIOINF-2013-1569	footprintDB: a database of transcription factors with annotated cis elements and binding interfaces <a href="#">[View Submission]</a>	04-Sep-2013	13-Oct-2013	<ul style="list-style-type: none"> <li>• Major Revision (13-Oct-2013)</li> <li>• a revision has been submitted</li> </ul> <a href="#">view decision letter</a>	a revision has been submitted (BIOINF-2013-1569.R1)	<a href="#">top</a>

**Figura 2.17:** Ejemplo del proceso de revisión de un artículo en la revista *Bioinformatics*.

## 2.6. Primeros pasos con L<sup>A</sup>T<sub>E</sub>X

LaTeX es un sistema de preparación de documentos creado inicialmente por Leslie Lamport, y al mismo tiempo, un *lenguaje de marcas* para la escritura de documentos. LaTeX está basado en el sistema de escritura de documentos basado en macros TeX, creado por Donald Knut en 1978. Como tal, LaTeX es un *conjunto de macros y convenciones* que definen los distintos elementos de un documento: fuente y tamaño de la misma, párrafos, secciones, tablas, figuras, listas de elementos, referencias bibliográficas, etc. Actualmente existen en la red multitud de recursos que describen el sistema LaTeX<sup>25</sup>.

El paradigma de LaTeX es que los autores se deben centrar en el contenido de lo que quieren escribir, sin estar distraídos por su representación visual. En ese punto difiere bastante de las *suites* de ofimática actuales, enfocadas en el paradigma WYSIWYG (*What You See Is What You Get*). Otro de los puntos fuertes de LaTeX es su potente sistema de *escritura de fórmulas matemáticas*, razón por la cuál este sistema es usado por toda la comunidad científica de los campos de la matemática, la física y otras disciplinas. Este mismo sistema ha hecho que se ha convertido en el estándar *de facto* de definición de fórmulas para sitios web como Wikipedia<sup>26</sup>.

Existen multitud de distribuciones del sistema LaTeX, cada una de ellas enfocada o en una plataforma o en la solución de algún problema en concreto. Las herramientas TeX y LaTeX originales están enfocadas en la generación de documentos DVI (*device independent*), que luego se convierten en ficheros PostScript o PDF mediante los comandos ‘dvips’, ‘dvipdf’, ‘dvipdfm’ o ‘dvipdfmx’.

Estas herramientas originalmente tenían una serie de limitaciones: el motor interno era bastante rígido, con lo que la inclusión de una característica nueva era bastante complicada; sólo trabajaban con sus propias fuentes tipográficas, instaladas como paquetes TeX; las imágenes que se podían incluir como figuras tenían que estar en formato EPS (*Encapsulated PostScript*); no soportaban documentos escritos en idiomas con distinto sentido de escritura al del inglés (japonés, chino, hindú, árabe ...), ni la representación en formato Unicode de los caracteres no existentes en el alfabeto inglés. Por ello, surgieron ramas y variantes de las herramientas originales:

- **pdflatex**: Soporta la generación de documentos PDF, la definición de hiperenlaces internos y externos al documento y la inclusión de imágenes en formato PDF, PNG y JPEG.
- **lualatex**: Usa el lenguaje de scripting *lua*, lo cuál permite definir, por ejemplo, paquetes de generación de gráficas parametrizadas. Soporta además fuentes TrueType. Al ser una evolución de pdflatex, también integra todas sus mejoras.
- **xelatex**: Es una reescritura del sistema LaTeX para soportar por completo el procesamiento y flujo de texto de todos los idiomas cuyos caracteres se pueden representar en Unicode. Está basado en el sistema de fuentes OpenType (un superconjunto de TrueType), de manera que soporta características avanzadas de escritura como ligaduras de caracteres. Y al haber tomado esa reescritura como base a pdflatex, también integra todas sus mejoras.

### 2.6.1. Crear un artículo con LaTeX

Todo documento LaTeX suele tener un nombre fichero con extensión ‘.tex’ o ‘.latex’, y tiene una estructura similar a la del Código 2.1. Comienza con la definición de la *plantilla de documento* que vamos a usar (libro, artículo, presentación...), el idioma (para definir las normas de división de palabras) y el tamaño base de las fuentes. Esa plantilla define además las macros específicas del tipo de documento que definen.

<sup>25</sup>LaTeX Wikibook. <http://en.wikibooks.org/wiki/LaTeX>

<sup>26</sup>Displaying a formula en Wikipedia. [http://en.wikipedia.org/wiki/Help:Displaying\\_a\\_formula](http://en.wikipedia.org/wiki/Help:Displaying_a_formula)

Todas las *declaraciones de macros* comienzan con el carácter ‘\’. Por ejemplo, la macro ‘\section’ es común a muchas plantillas, mientras que la plantilla ‘book’ define la macro ‘\bookchapter’, que sólo tiene sentido a la hora de ir definiendo capítulos. Todo comentario que se quiera incluir en un documento LaTeX tiene que comenzar por el carácter ‘%’. Dichos comentarios no aparecerán en el documento PDF generado a partir del fichero fuente original.

Tras la declaración de la plantilla del documento se declaran todos los *paquetes* que van a ser necesarios mediante la macro ‘\usepackage’ (ver Código 2.1). Por ejemplo, se carga el paquete que define las macros de manipulación de imágenes externas con ‘\usepackage{graphicx}’. Además de la carga de los paquetes, en esta parte inicial se realizan las inicializaciones necesarias de los mismos, como por ejemplo declarar el título y autores para la plantilla ‘article’, declarar qué fuentes se van a usar, qué márgenes va a tener el documento, etc.

El sistema de gestión de *referencias bibliográficas* de LaTeX permite definir las referencias dentro del cuerpo del manuscrito (ya sea usando primitivas LaTeX o el paquete ‘filecontents’, como en el Código 2.1), o fuera de él en un fichero en formato BibTeX (.bib). Mediante la macro ‘\cite’ se define una cita a una referencia declarada en la bibliografía. Al final del documento las macros ‘\bibliographystyle’ y ‘\bibliography’ permiten por un lado definir el estilo para citar y mostrar las referencias, y por otro cargar el fichero de referencias en formato BibTeX y obligar a que las referencias usadas aparezcan embebidas en el documento. La forma en la que gestiona LaTeX las referencias bibliográficas hace que sólo se muestren aquellas referencias que han sido citadas a lo largo del texto, con lo que es bastante común hacer uso de un fichero con la biblioteca de referencias recopiladas.

Una vez hechas todas las declaraciones iniciales, se incluye el *cuerpo del documento* dentro de una declaración del estilo ‘\begin{document} ... \end{document}’. Cada *sección* de la plantilla artículo se inicia con la macro ‘\section’ y se puede insertar una *etiqueta* (‘\label’) para más tarde referenciarlas mediante ‘\ref’. Para evitar confusiones, el *sistema de referencias cruzadas* es distinto del de referencias bibliográficas (‘\cite’).

Dentro del documento, la separación entre un párrafo y otro se produce al introducir una línea en blanco. De esta manera, un conjunto de líneas consecutivas son tratadas como integrantes de un mismo párrafo, y a la hora de ser representadas aparecen como si sólo estuvieran separadas mediante un espacio, ignorando los saltos de línea existentes. También se pueden insertar saltos de línea con las macros ‘\\’ o ‘\vspace’.

La *inserción de figuras* en el texto se realiza con ayuda del paquete ‘graphicx’ previamente cargado. Los formatos aceptados por el compilador ‘pdflatex’, ‘lualatex’ y ‘xelatex’ (ver Sección 2.6) son PNG, JPG o PDF. Si se requiere insertar imágenes desde otros formatos habrá que convertirlas previamente a uno de estos formatos o cargar paquetes adicionales. Hay que especificar correctamente la ruta al fichero con la imagen o el compilador fallará.

Dentro de LaTeX existen multitud de paquetes que gestionan el tema de las *tablas*, siendo el entorno más sencillo de usar el de ‘\begin{table}\begin{tabular} ... \end{tabular}\end{table}’. Al comienzo de ‘\begin{tabular}’ se definen el número de columnas con las iniciales del alineamiento deseado (‘l’ *left*, ‘c’ *center*, ‘r’ *right*) y las líneas verticales que las separan mediante ‘|’. Posteriormente cada fila estará delimitada por el carácter ‘\\’ y cada columna por ‘&’. Se podrán insertar líneas horizontales entre filas mediante ‘\hline’.

Como ya se ha comentado, LaTeX es el sistema de edición preferido para publicaciones de campos científicos que requieren la escritura de fórmulas matemáticas complejas. Para ello, LaTeX tiene implementado un sistema de *escritura de ecuaciones* que es más o menos sencillo con un *output* muy profesional. No se entrará en detalle a explicar su funcionamiento, pero se puede ver un ejemplo en el

Código 2.1 y el Wikibook de LaTeX está muy bien documentado<sup>27</sup>.

En el Código 2.1 se han utilizado los paquetes y macros explicados en esta sección, se recomienda probar y modificar el código en un editor de LaTeX para comprender su funcionamiento.

Código 2.1: Ejemplo de documento LaTeX, basado en la plantilla ‘article’.

```
1 % Definir el tipo/plantilla de documento
2 \documentclass[12pt,spanish]{article}
3
4 % Definir los paquetes adicionales a usar
5 \usepackage{graphicx} % Paquete para gestionar imágenes
6 \usepackage[utf8]{inputenc} % Paquete para escribir código LaTeX con caracteres UTF8 (e
   ñes, tildes...)
7 \usepackage[a4paper, margin=1.5cm]{geometry} % Paquete para dar tamaño y definir
   márgenes del documento
8 \usepackage[spanish, es-tabla]{babel} % Paquete para traducir el nombre de las
   secciones y leyendas al español
9 \usepackage{filecontents} % Permite crear el fichero de referencias .bib desde el código
   LaTeX
10
11 % Título, autores y fecha
12 \title{Ejemplo de documento en \LaTeX{}}
13 \author{J. M. Fernández}
14 \date{\today}
15
16 % Generamos un fichero 'ejemplo.bib' con las referencias a citar
17 \begin{filecontents}{ejemplo.bib}
18 @article{cit:Hoffmann2005,
19   author = {Hoffmann, R. and Valencia, A.},
20   title = {Implementing the iHOP concept for navigation of biomedical literature},
21   journal = {Bioinformatics},
22   volume = {21 Suppl 2},
23   pages = {ii252-8},
24   year = {2005}
25 }
26 @article{cit:RubioCamarillo2013,
27   author = {Rubio-Camarillo, M. and Gomez-Lopez, G. and Fernandez, J. M. and Valencia,
   A. and Pisano, D. G.},
28   title = {RUbioSeq: a suite of parallelized pipelines to automate exome variation and
   bisulfite-seq analyses},
29   journal = {Bioinformatics},
30   volume = {29},
31   number = {13},
32   pages = {1687-9},
33   year = {2013}
34 }
35 \end{filecontents}
36
37 % Comienza el cuerpo del documento
38 \begin{document}
39
40 % Genera el título automáticamente
41 \maketitle
42
43 \begin{abstract}
44 Espacio para escribir el resumen del artículo.
45 \end{abstract}
46
```

<sup>27</sup>LaTeX/Mathematics. <http://en.wikibooks.org/wiki/LaTeX/Mathematics>

```

47 % Insertamos secciones con una etiqueta o 'label' para referenciarlas
48 \section{Introducción}
49 \label{sec:intro}
50 Espacio para escribir el texto de la sección 'Introducción'.
51
52 \section{...}
53
54 \section{Resultados}
55 \label{sec:intro}
56 Espacio para escribir el texto de la sección 'Resultados'.
57 Como se comentó en la Sección \ref{sec:intro} y se puede observar en la Figura \ref{fig:imagen1}, la Tabla \ref{tab:tabla1}, la Ecuación \ref{eq:ecuacion1} y las Referencias \cite{cit:Hoffmann2005,cit:RubioCamarillo2013}.
58
59 % Insertamos una figura desde el archivo 'imagen1.png'
60 \begin{figure}[h!]
61     \centering
62     \includegraphics[width=0.6\textwidth]{imagen1.png}
63     \caption{Descripción de la imagen.}
64     \label{fig:imagen1}
65 \end{figure}
66
67 % Insertamos una tabla
68 \begin{table}[h]
69     \centering
70     \begin{tabular}{l | c | r}
71         1 & 2 & 3 \\
72         \hline
73         4 & 5 & 6 \\
74         \hline
75         7 & 8 & 9 \\
76     \end{tabular}
77     \caption{Descripción de la tabla.}
78     \label{tab:tabla1}
79 \end{table}
80
81 % Insertamos una ecuación
82 \begin{equation}
83     \centering
84     \cos(2\theta) = \cos^2 \theta - \sin^2 \theta
85     \label{eq:ecuacion1}
86 \end{equation}
87
88 % Imprime las referencias bibliográficas del fichero creado 'ejemplo.bib'
89 \bibliographystyle{plain}
90 \bibliography{ejemplo}
91
92 % Termina el documento
93 \end{document}

```

## 2.6.2. Compilar código LaTeX

Una vez escrito el documento, hay que *compilar el código* mediante ‘pdflatex’ (recomendado) o alguno de sus derivados para verificar su corrección y su consistencia a nivel de referencias, así como para *generar el documento PDF resultante*. Debido al funcionamiento del sistema LaTeX a la hora de resolver las referencias cruzadas y bibliográficas, y de algunos paquetes de LaTeX, es bastante común tener que ejecutar varias veces ‘pdflatex’ hasta obtener el documento PDF terminado. Por ejemplo:

```
$ pdflatex documento.tex
$ bibtex documento.tex
$ pdflatex documento.tex
$ pdflatex documento.tex
```

Todo este esfuerzo a bajo nivel se puede evitar instalando y utilizando el comando ‘`latexmk`’, que se encarga de detectar si hay que ejecutar ‘`pdflatex`’ una o más veces, si es necesario ejecutar ‘`bibtex`’, etc. Dicho comando es compatible con todas las variantes de LaTeX, permitiendo incluso la integración con sistemas más especializados, como MikTeX.

```
$ latexmk -pdf documento.tex
```

También la mayoría de editores especializados de LaTeX que se citarán en la siguiente sección suelen ejecutar automáticamente dichos comandos. Si es necesario realizar algún paso adicional especial, otra alternativa es incluir los anteriores comandos en un *script* (archivo .bat en Windows o .sh en Linux).

### 2.6.3. Editores LaTeX

Existen multitud de editores de TeX y LaTeX como se puede comprobar en la página de Wikipedia dedicada en exclusiva<sup>28</sup>. Por ejemplo, este libro ha sido editado en parte usando *Texmaker*<sup>29</sup> (ver Figura 2.18). A continuación una lista con los editores más recomendados:

- Texmaker: probablemente sea la opción más popular de la lista, es libre y multiplataforma. Muy fácil de usar y es ideal para novatos.
- Kile: similar a Texmaker pero ligeramente más completo y diseñado para entornos KDE de Linux. Tiene muchos asistentes, menús e iconos que lo hacen perfecto para el que recién empieza. Para la mayoría se trata del mejor editor para Linux.
- LaTeXila: Similar a Kile pero para entornos Gnome de Linux.
- TeXworks: editor libre y multiplataforma basado en TeXshop, un editor LaTeX para Mac.
- LyX: editor es un editor libre y multiplataforma WYSIWYM (“*What You See Is What You Mean*”) que permite introducir y formatear texto sin saber absolutamente nada de LaTeX. Para introducir fórmulas se puede utilizar su editor similar a ‘MathType’.
- Texlipse: un completo *plugin* para el famoso entorno de programación Eclipse.
- Gedit: el editor de textos de Gnome junto con su complemento para LaTeX (‘`gedit-latex-plugin`’).
- Gummi: se trata básicamente de una terminal y un visor, así que para un novato seguramente no es lo mejor.
- JLatexEditor: editor Open Source bastante completo y multiplataforma.
- Medit: destaca por tener un terminal empotrado. Similar a Gedit con su plugin para LaTeX.

Para más información y enlaces a los editores consultar el listado publicado en el blog ‘Cristal Polarizado’<sup>30</sup>.

---

<sup>28</sup>Comparison of TeX editors. [http://en.wikipedia.org/wiki/Comparison\\_of\\_TeX\\_editors](http://en.wikipedia.org/wiki/Comparison_of_TeX_editors)

<sup>29</sup>Texmaker. <http://www.xmlmath.net/texmaker>

<sup>30</sup>12 Editores LaTeX para GNU/Linux.

<http://cristalpolarizador.blogspot.com/2011/04/12-editores-latex-para-gnulinux.html>

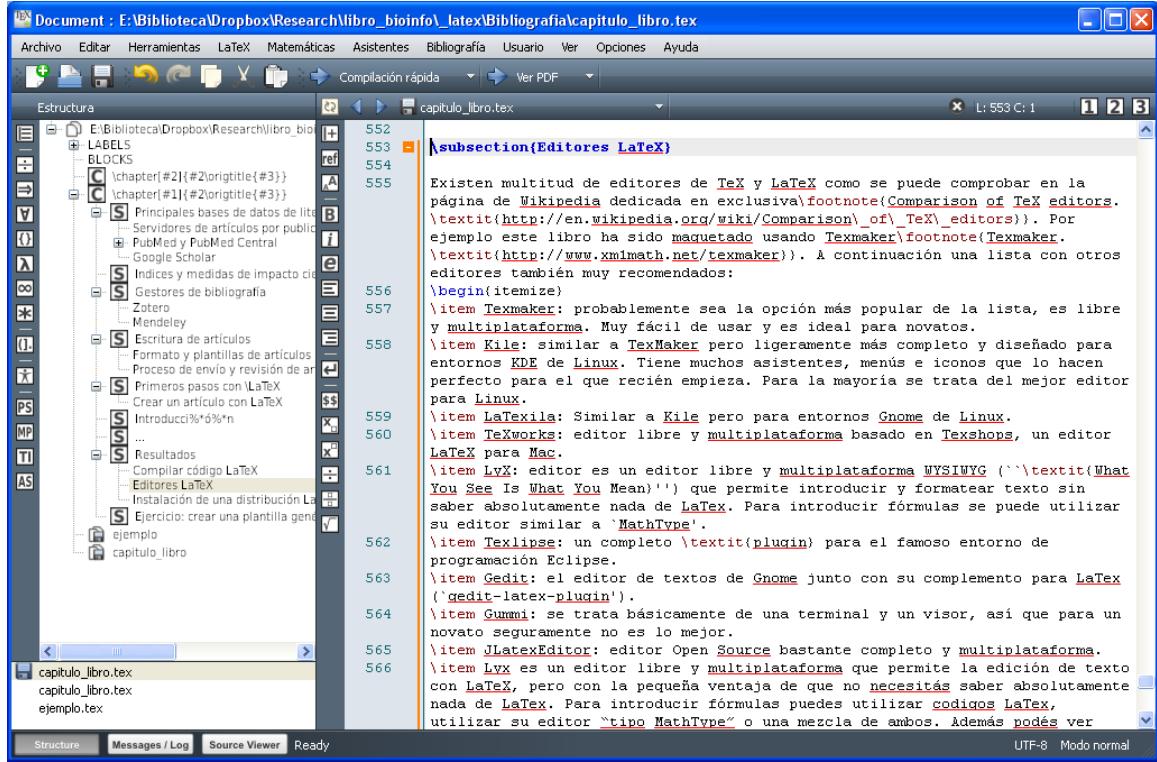


Figura 2.18: Captura de pantalla del editor Texmaker.

#### 2.6.4. Instalación de una distribución LaTeX

Existen multitud de distribuciones del sistema LaTeX, dentro de las cuáles destacan *TeX Live*<sup>31</sup> (principalmente para los sistemas UNIX/Linux), *MiKTeX*<sup>32</sup> (para Microsoft Windows) y *MacTeX*<sup>33</sup> (superconjunto de TeX Live para Mac OS X). Todas estas distribuciones incluyen, entre otras cosas, las herramientas actuales del sistema LaTeX ('tex', 'latex', 'bibtex', 'pdflatex', 'xelatex', etc.), los paquetes y fuentes LaTeX estándar, y las plantillas para la elaboración de documentos estándar y de manuscritos para diversas revistas científicas, como por ejemplo IEEE. TeX Live se encuentra disponible como paquete estándar para casi todas las distribuciones Linux existentes, con lo que una instalación completa mediante el *sistema de paquetes* de la distribución suele ser bastante trivial, por ejemplo en Ubuntu:

```
$ sudo apt-get install texlive-latex-base texlive-latex-recommended texlive-fonts-
recommended texlive-latex-extra
```

MiKTeX y MacTeX poseen *instaladores gráficos* para Windows y Mac OS X respectivamente, que se pueden descargar desde sus páginas web<sup>32,33</sup>. Se recomienda *instalar la distribución deseada antes de instalar el editor*, debido a que durante la instalación del editor, éste debe reconocer la ruta de la distribución LaTeX previamente instalada y configurarse automáticamente para poder compilar los documentos LaTeX. En caso contrario podría ser necesario definir las rutas de los comandos de LaTeX manualmente en las opciones de configuración del editor.

<sup>31</sup>TeX Live. <http://www.tug.org/texlive>

<sup>32</sup>MiKTeX. <http://miktex.org/>

<sup>33</sup>MacTeX. <http://tug.org/mactex>



## 2.7. Bibliografía

- [1] R. Hoffmann and A. Valencia. Implementing the iHOP concept for navigation of biomedical literature. *Bioinformatics*, 21(Suppl 2):ii252–ii258, Oct. 2005.
- [2] M. Rubio-Camarillo, G. Gómez-López, J. M. Fernández, A. Valencia, and D. G. Pisano. RUbioSeq: a suite of parallelized pipelines to automate exome variation and bisulfite-seq analyses. *Bioinformatics*, Apr. 2013. PMID: 23630175.



# Capítulo 3

## Estadística y R

*Sonia Tarazona*

### 3.1. Introducción

La estadística es la disciplina que se encarga de la obtención y análisis de datos mediante el uso de herramientas informáticas y modelos matemáticos, con la finalidad última de responder las preguntas del investigador ante un problema no determinista y ayudarle en la toma de decisiones. Para ello, es necesario en primer lugar diseñar correctamente el experimento o procedimiento de muestreo con tal de no introducir sesgos no deseados en nuestros datos. Una vez obtenidos los datos, la estadística se encarga de resumir la información contenida en ellos de forma eficiente y comprensible (*estadística descriptiva*). Si queremos extender las conclusiones de nuestro estudio no sólo a los individuos de nuestra muestra sino a toda la población, recurrirímos a la *inferencia estadística*, que medirá el error que estamos cometiendo al hacer este tipo de estimaciones.

El objetivo de este capítulo es dar una visión general de la estadística y ofrecer una perspectiva de algunos de los métodos más empleados en bioinformática, intercalando ejemplos e indicaciones de cómo realizar los cálculos mediante el *lenguaje de programación estadística R*. No pretendemos desarrollar detalladamente todos los conceptos y métodos estadísticos que aquí se presentan, ya que no es este un libro de estadística.

#### 3.1.1. Población y muestra

En terminología estadística, una *población* es el conjunto de todos los individuos que son objeto de nuestro estudio y a los que queremos extender la conclusiones del mismo. La mayor parte de las veces es imposible obtener información de toda nuestra población objetivo (por falta de tiempo o recursos, porque la población es infinita o bien porque tomar una muestra significa destruir al propio individuo). Por ello, nos debemos limitar a recoger datos de un pequeño subconjunto representativo de la población total, al que llamamos *muestra*. Por ejemplo, supongamos que queremos estudiar qué genes se expresan o inhiben en enfermos de cáncer de pulmón en comparación con pacientes sanos, y tenemos un grupo de 20 pacientes con dicho cáncer y un grupo de 25 personas sanas (controles) para realizar el estudio. El grupo de 20 pacientes enfermos constituye una muestra, mientras que la población estaría constituida por todos los enfermos de cáncer de pulmón. Análogamente, se definiría la población y muestra en los

controles. Nuestro propósito es comparar ambas poblaciones (sanos y enfermos), pero sólo disponemos de la información que nos proporcionan las muestras.

Es importante que la muestra sea *representativa* de la población, ya que el objetivo será hacer estimaciones sobre la población desconocida basándonos en dicha muestra. Una forma de garantizar la representatividad de la muestra es seleccionar al azar los elementos de la misma (*muestreo aleatorio*). También juega un papel fundamental el tamaño de la muestra. A mayor tamaño muestral, menor será el error de nuestras estimaciones.

### 3.1.2. Variables aleatorias

Una *variable aleatoria* es cualquier característica que podamos medir en los individuos de nuestra población y cuyo valor no conocemos de antemano sino que viene determinado por el azar.

En estadística, las variables aleatorias se clasifican en dos tipos: *discretas* y *continuas*. Una *variable discreta* es aquella que sólo puede tomar unos valores determinados, normalmente enteros. Es decir, toma valores en un conjunto discreto (finito o infinito numerable) de puntos. Una *variable continua*, por contra, puede tomar cualquier valor. Según sea el tipo de la variable aleatoria con la que estamos trabajando, debemos elegir los estadísticos, gráficos o métodos apropiados.

La media de una variable aleatoria  $X$  también se denomina *esperanza matemática* y se denota por  $E(X)$ , mientras que su *varianza* se denota por  $Var(X)$ .

### 3.1.3. Introducción a R

*R*<sup>1</sup> es un lenguaje de programación pensado para la estadística y, por tanto, muchos de los métodos, funciones o gráficos que podamos necesitar están ya implementados. R es un *software* libre y funciona en cualquier sistema operativo. Además, según las necesidades del usuario, se pueden utilizar librerías con funcionalidades de cálculo o generación de gráficos más especializadas. En la página web de R, podemos acceder al repositorio general CRAN<sup>2</sup>. Otro repositorio más específico para bioinformática es Bioconductor<sup>3</sup>.

En la propia web de R<sup>1</sup>, encontraremos manuales que nos ayudarán a manejarlo en el uso de este lenguaje. No obstante, a lo largo de este capítulo presentaremos aquellas funciones que consideremos más útiles para llevar a cabo los análisis estadísticos descritos. No pretendemos citar exhaustivamente todas las funciones implementadas en R, ni todas las opciones que contempla cada función de las que mencionemos. Simplemente ofrecemos un sencillo “recetario” que ayude al usuario no experimentado a realizar en R los análisis más elementales y a entender su funcionamiento.

En R, el símbolo ‘>’ nos indica que el programa está preparado para recibir nuestras órdenes. Podemos realizar cualquier operación básica como por ejemplo:

```
> 2+5  
[1] 7  
> log2(8)  
[1] 3  
> sqrt(36)*45^3  
[1] 546750
```

<sup>1</sup>R Project for Statistical Computing. <http://www.r-project.org>

<sup>2</sup>The Comprehensive R Archive Network. <http://cran.r-project.org>

<sup>3</sup>Bioconductor. <http://www.bioconductor.org>

Para crear el objeto ‘dato1’ que sea igual a 5, el objeto ‘dato2’ que sea igual a ‘ $\log_{10}(27)$ ’ y el objeto ‘miresultado’ que recoja la diferencia entre ambos:

```
> dato1 <- 5
> dato2 <- log10(27)
> miresultado <- dato1-dato2
> miresultado
[1] 3.568636
```

La asignación de un valor o valores a un objeto de R, también se puede realizar utilizando ‘=’ en lugar de ‘<-’.

Veamos algunos ejemplos con otros tipos de objetos en R. Notar que se pueden escribir comentarios en R, precediendo el texto del símbolo '#':

```
# Vectores
> mivector1 <- c(2,3,5,8)
> mivector1
[1] 2 3 5 8
> mivector2 <- 7:10
> mivector2
[1] 7 8 9 10
> miresultado <- mivector1 + mivector2
> miresultado
[1] 9 11 14 18

# Matrices
> mimatriz <- matrix(1:15, ncol = 3)
> mimatriz
 [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> t(mimatriz) # matriz traspuesta
 [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
> mimatriz[,2] # columna 2 de la matriz
[1] 6 7 8 9 10

# Listas
> milista = vector("list", length = 2)
# Los elementos de una lista pueden ser de distinta naturaleza
# y dimensiones
> milista[[1]] = c(1:5,20:30)
# El segundo elemento de la lista es una muestra aleatoria
# de 30 valores del 1 al 100 sin reemplazamiento
> milista[[2]] = sample(x = 1:100, size = 30, replace = FALSE)
# Aplicamos la misma función (media en este caso)
# a todos los elementos de la lista
> sapply(milista, mean)
[1] 18.12500 46.86667
```

En bioinformática, los datos suelen estar generados por otra aplicación o extraídos de una base de datos. Por tanto, lo más habitual es disponer de uno o más ficheros de texto que recojan dicha información. Como ejemplo, vamos a leer en R el fichero ‘SupplementaryTable2.txt’<sup>4</sup>, que contiene datos de

---

<sup>4</sup>Supplemental Research Data from [12]. <http://genome.cshlp.org/content/18/9/1509/suppl/DC1>

expresión génica (medida mediante RNA-seq) para 32000 genes de humano, en 2 tejidos (riñón e hígado) y con 7 réplicas técnicas para cada tejido [12]:

```
> marioni = read.delim("SupplementaryTable2.txt", header = TRUE,
+ sep = '\t', as.is = TRUE)
```

Para obtener ayuda sobre cualquier función (p.e. ‘`read.delim`’), así como sus posibles opciones o parámetros, basta anteponer el símbolo ‘?’ delante del comando:

```
> ?read.delim
```

## 3.2. Estadística descriptiva

Las herramientas utilizadas en *estadística descriptiva* para resumir la información contenida en los datos son los parámetros o estadísticos (media, varianza,...), los gráficos y las tablas de frecuencias. A continuación describimos los más utilizados.

### 3.2.1. Parámetros de posición y dispersión

Los *parametros de posición* nos aportan información sobre la localización de los datos. Algunos de los parámetros de posición más utilizados son la media, la mediana y los percentiles.

La *media* (aritmética) se suele representar con la letra griega  $\mu$  cuando se refiere a la *media poblacional* y con  $\bar{x}$  si se refiere a la *media muestral*. Dada la muestra  $\{x_1, x_2, \dots, x_n\}$ , la media muestral se define como:

$$\bar{x} = \frac{\sum x_i}{n} \quad (3.1)$$

Análogamente se definiría la media poblacional.

Para obtener la mediana o los percentiles, se deben ordenar los datos de menor a mayor. La *mediana* será el valor que divide a los datos en dos partes iguales, dejando a la mitad de los datos por debajo de ella y a la otra mitad por encima. Si el número de datos es impar, la mediana es el valor central. Si es par, será la media de los dos datos centrales. En el caso de los *percentiles*, el percentil  $P$  es el valor que deja un  $P\%$  de los datos por debajo de él y el resto por encima.

Veamos cómo calcular estos parámetros con R, utilizando los datos de Marioni (ver Subsección 3.1.3). En concreto, calculamos la media, la mediana y el percentil 90 de los valores de expresión de todos los genes para la réplica 1 de riñón:

```
> mean(marioni[, "R1L1Kidney"]) # media
[1] 56.40553
> median(marioni[, "R1L1Kidney"]) # mediana
[1] 2
> quantile(marioni[, "R1L1Kidney"], probs = 0.9) # percentil 90
90 %
129
```

Los *parámetros de dispersión* sirven para cuantificar la variabilidad de nuestros datos, es decir, cómo de diferentes son entre sí. Los parámetros de dispersión más utilizados son la varianza y desviación típica (también denominada desviación estándar), el rango y el rango intercuartílico.

La *varianza* de una población de tamaño  $N$  se define de la siguiente forma:

$$\sigma^2 = \frac{\sum(x_i - \mu)^2}{N} \quad (3.2)$$

Aunque la *varianza muestral* también se puede calcular mediante una fórmula análoga a la anterior pero utilizando el tamaño muestral  $n$ , es más habitual obtenerla de la siguiente forma, ya que resulta ser un mejor estimador de la varianza poblacional:

$$s^2 = \frac{\sum(x_i - \mu)^2}{n - 1} \quad (3.3)$$

Dado que la *varianza* es un promedio de las desviaciones al cuadrado de cada uno de los datos respecto de la media, se mediría en unidades al cuadrado. Por ello, la *desviación típica* o *desviación estándar* ( $\sigma$  para la población y  $s$  para la muestra) se calcula como la raíz cuadrada de la varianza. La desviación típica nos da una idea más intuitiva de la dispersión de nuestros datos respecto de la media.

El *rango* se define como la diferencia entre el valor máximo y mínimo de nuestros datos. El *rango intercuartílico* es la diferencia entre el percentil 75 (también denominado tercer cuartil) y el percentil 25 (primer cuartil).

A continuación se indica cómo calcular estos cuatro parámetros de dispersión en R para los valores de expresión de todos los genes en la réplica 1 de riñón (datos de Marioni):

```
> var(marioni[, "R1L1Kidney"]) # varianza
[1] 149135.7
> sd(marioni[, "R1L1Kidney"]) # desviación típica
[1] 386.181
> diff(range(marioni[, "R1L1Kidney"])) # rango
[1] 36493
> IQR(marioni[, "R1L1Kidney"]) # rango intercuartílico
[1] 40
```

Se debe tener en cuenta que la varianza y desviación típica calculada por R son las muestrales, según la fórmula presentada anteriormente.

Notar que los estadísticos muestrales son a su vez variables aleatorias, ya que el valor que tomen variará según la muestra aleatoria obtenida. En concreto, se puede demostrar que si una variable aleatoria  $X$  tiene media  $E(X) = \mu$  y varianza  $Var(X) = \sigma^2$ , sus estadísticos muestrales para una muestra de tamaño  $n$  cumplen lo siguiente:

$$E(\bar{X}) = \mu; Var(\bar{X}) = \sigma^2/n$$

$$E(s^2) = \sigma^2$$

### 3.2.2. Gráficos en R

#### Función ‘plot’

Una función básica en R para dibujar gráficos es ‘plot’. Los múltiples argumentos de esta función permiten la obtención de gráficos muy variados. En la Figura 3.1 se muestran un par de ejemplos de utilización de esta función. A continuación se indica el código de R empleado para dibujar dichas gráficas, que están basadas en los datos de Marioni. Para el gráfico (A), se ha calculado los logaritmos en base 2 de la suma de las 7 réplicas (técnicas) de riñón y de las 7 de hígado para cada uno de los genes. Después, se ha obtenido un gráfico de dispersión que nos permitirá observar la relación entre la expresión de los genes en hígado y en riñón. En el ejemplo (B), se representan los perfiles de expresión del gen ORM1 de la  $\alpha 1$ -glicoproteína ácida (marcador tumoral) a lo largo de todas las réplicas de los dos tejidos.

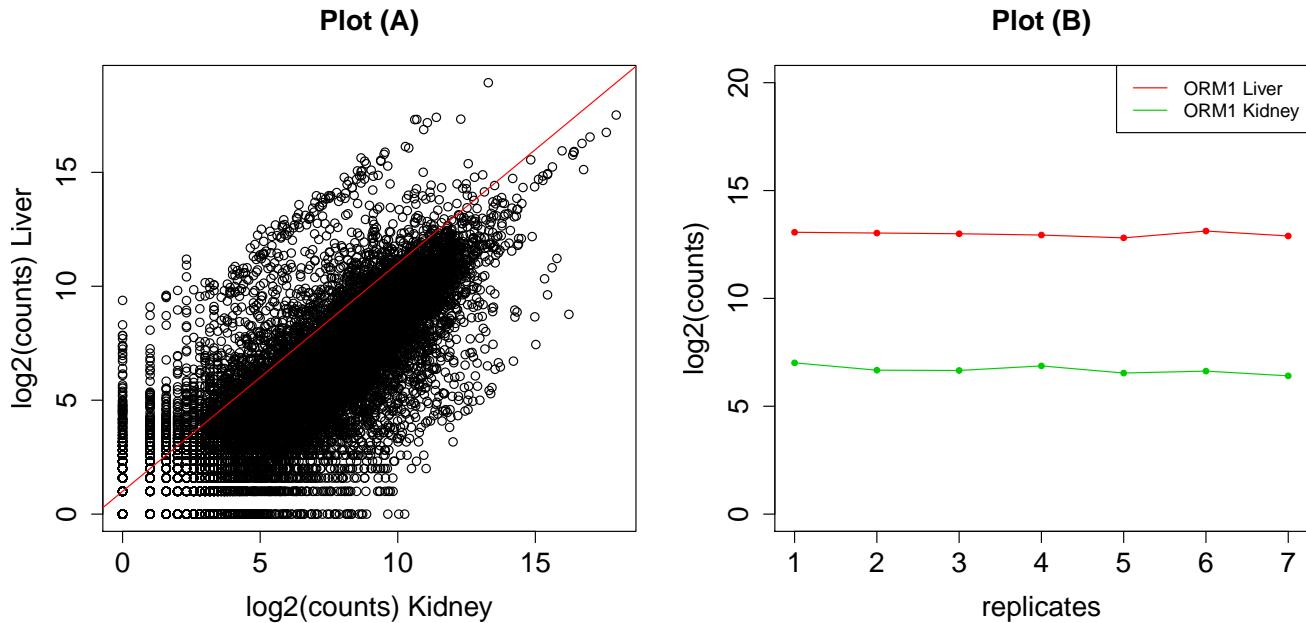
```

# Para guardar los gráficos en un archivo pdf
> pdf("fig_plots.pdf", width = 6*2, height = 6)
# Para indicar que la figura incluye dos gráficos situados
# horizontalmente en una fila y dos columnas
> par(mfrow = c(1,2))
## Plot (A)

# Con el comando "grep" seleccionamos las columnas que nos interesan
> kidney <- marioni[,grep("Kidney", colnames(mariioni))]
> liver <- marioni[,grep("Liver", colnames(mariioni))]
# Con "rowSums" sumamos los valores de las 7 réplicas de cada tejido
> plot(log2(rowSums(kidney)), log2(rowSums(liver)), xlab = "log2(counts) Kidney",
+ ylab = "log2(counts) Liver", main = "Plot (A)")
> abline(a=1, b=1, col = 2) # para dibujar la recta y = x (en rojo)
## Plot (B)

# Extraemos los datos del gen cuyo perfil de expresión vamos a representar:
> orm1 <- subset(mariioni, ExternalID == "ORM1")
# Perfil de de expresión en Hígado:
> orm1_liver <- orm1[,grep("Liver", colnames(orm1))]
> plot(1:7, log2(orm1_liver), type = "o", col = 2, pch = 20,
+ xlab = "replicates", ylab = "log2(counts)", ylim=c(0,20), main = "Plot (B)")
# Perfil de de expresión en Riñón:
> orm1_kidney <- orm1[,grep("Kidney", colnames(orm1))]
> lines(1:7, log2(orm1_kidney), type = "o", col = 3, pch = 20)
> legend("topright", c("ORM1 Liver", "ORM1 Kidney"), col = c(2,3), lty = 1)
# para indicar a R que esto es todo lo que incluiremos en la figura
> dev.off()
null device
1

```



**Figura 3.1:** Ejemplos de cómo usar la función 'plot'.

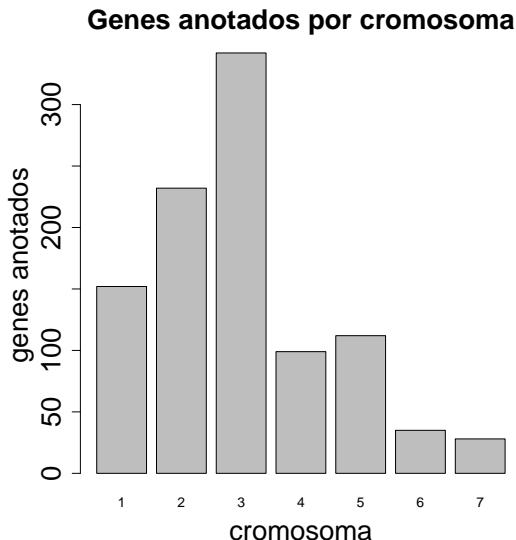
## Diagrama de barras

El *diagrama de barras* se utiliza para representar la frecuencia con la que aparecen los distintos valores de una variable cualitativa. En R, la función específica para generar diagramas de barras es ‘`barplot`’. Como ejemplo, supongamos que tenemos una lista (ficticia) de genes y el cromosoma al que pertenecen. Veamos cómo simular en R una lista de 1000 genes con estas características:

```
# Generamos aleatoriamente el cromosoma (del 1 al 7) al que pertenece cada gen.  
# En este caso, no todos los cromosomas tienen la misma probabilidad de aparecer.  
> anot.crom <- sample(1:7, 1000, prob = c(0.15,0.25,0.35,0.1,0.1,0.03,0.02),  
+ replace = TRUE)  
> names(anot.crom) <- paste("gen", 1:1000, sep = "_") # nombres de los genes  
> head(anot.crom) # lista de genes
```

Si estamos interesados en saber cuántos genes están anotados a cada cromosoma, podemos dibujar el diagrama de barras de la Figura 3.2 mediante el siguiente código:

```
> table(anot.crom) # tabla con número de genes por cromosoma  
anot.crom  
 1   2   3   4   5   6   7  
150 246 376 82 101 26 19  
> barplot(table(anot.crom), main = "Genes anotados por cromosoma",  
+ xlab = "cromosoma", ylab = "genes anotados")
```



**Figura 3.2:** Diagrama de barras ('`barplot`').

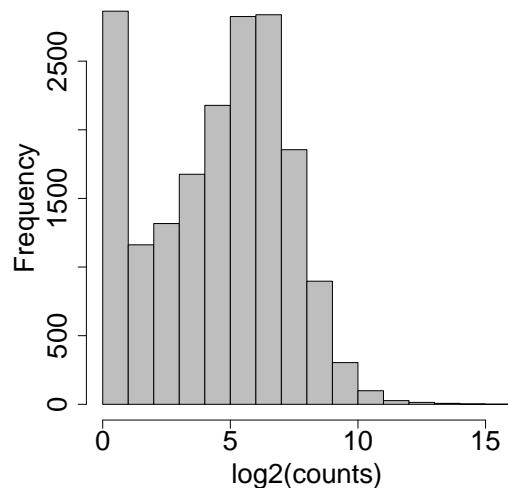
## Histograma

El *histograma* nos permite representar la distribución de frecuencias de nuestros datos. Se utiliza para variables aleatorias cuantitativas y cuando se dispone de al menos 50 observaciones. La forma más sencilla de construir un histograma es dividir el rango de los datos en intervalos de igual amplitud. Se cuenta el número de datos que “caen” en cada intervalo (frecuencia absoluta). La *frecuencia absoluta* de un intervalo determinado sería la altura de la barra del histograma correspondiente a dicho intervalo. Utilizando los datos de Marioni, veamos cómo construir en R un histograma para el logaritmo en base 2 de los datos de expresión de la primera réplica de riñón (ver Figura 3.3):

```
> hist(log2(marioni[, "R1L1Kidney"]), main = "Histograma R1L1Kidney",
+ col = "grey", xlab = "log2(counts)")
```

En primer lugar, indicamos a la función ‘`hist`’ los datos que queremos representar en el histograma. En nuestro caso, ‘`marioni[, "KidneyR1"]`’ a los que le aplicamos ‘`log2`’. El argumento ‘`main`’ nos permitirá indicarle el título del gráfico, ‘`col`’ es el argumento que controla el color de fondo de las barras del histograma (si no especificamos el valor de este parámetro, por defecto, las barras serán blancas) y ‘`xlab`’ sirve para añadir una etiqueta al eje X (‘`ylab`’ sería la correspondiente al eje Y). La función ‘`hist`’ dispone de más argumentos que se pueden ajustar para cambiar el formato del gráfico. Aunque no lo hemos utilizado, otro argumento útil es ‘`breaks`’, que sirve para especificar el número de intervalos o barras entre los que se distribuirán los datos o también en qué intervalos exactamente queremos que se distribuyan. Cuando no se especifica un valor concreto para un argumento, la función utiliza los valores programados por defecto, que se pueden consultar en la ayuda de dicha función.

**Histograma datos Marioni (R1L1Kidney)**



**Figura 3.3:** Histograma (‘`hist`’).

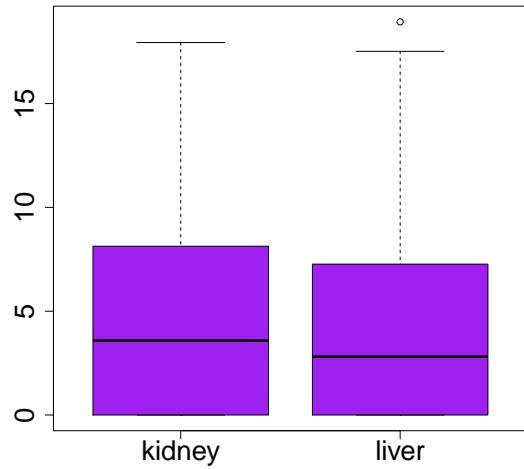
Debe tenerse en cuenta que, al representar ‘`log2(counts)`’ en el histograma, se están excluyendo aquellos genes con valor de expresión 0, puesto que el logaritmo devuelve el valor ‘`-Inf`’ y el histograma no los recoge. Por tanto, estamos representando aquellos genes con un nivel de expresión positivo. Así pues, por ejemplo, observamos en el histograma de la Figura 3.3 que hay casi 3000 genes con valores de expresión entre  $2^5$  y  $2^6$  (que correspondería al intervalo ‘`[5, 6]`’ para ‘`log2(counts)`’).

### Gráfico de cajas y bigotes

Para representar la distribución de una variable numérica, otra alternativa al histograma es el *gráfico de cajas y bigotes* (“*Box-Whisker plot*”). Este gráfico, además, nos permite comparar simultáneamente varias distribuciones. En R, se pueden crear gráficos de cajas y bigotes utilizando la función ‘`boxplot`’. El siguiente ejemplo de código muestra cómo comparar las distribuciones de los datos de expresión en riñón e hígado con los datos de Marioni.

```
\begin{lstlisting}
> boxplot(list("kidney"=log2(rowSums(kidney)+1), "liver"=log2(rowSums(liver)+1)),
+ col = "purple")
```

En este caso, hemos representado los datos en escala logarítmica para facilitar la visualización. Además, hemos sumado el valor 1 a la expresión para evitar el valor ‘`-Inf`’ al calcular ‘`log2(0)`’.



**Figura 3.4:** Gráfico de cajas y bigotes (‘boxplot’)

El resultado se muestra en la Figura 3.4, en la que vemos un gráfico de cajas y bigotes para cada tejido. Fijémonos en el correspondiente a “liver” para la interpretación de este tipo de representaciones gráficas. La caja representa los datos entre el primer y el tercer cuartil ( $P_{25}$  y  $P_{75}$ ), siendo la línea central, la mediana de los datos. Por tanto, el 50 % de los valores está comprendido entre los dos extremos de la caja (en este caso, entre 0 y 7 aproximadamente). Antes de ver qué significan los bigotes, necesitamos comprobar si existen datos anómalos. En un diagrama de cajas y bigotes, normalmente, se considera que un valor es anómalo (extremo) si está alejado de la caja más de 1,5 veces la longitud de la caja. Dicho de otro modo, el valor es anómalo si cae fuera del intervalo:  $[P_{25} - 1,5(P_{75} - P_{25}); P_{75} + 1,5(P_{75} - P_{25})]$ . Los datos anómalos se representan con puntos en la figura. En “liver” podemos observar la existencia de dos datos anómalos. Para dibujar los bigotes, tenemos que eliminar los posibles datos anómalos y calcular el mínimo y el máximo de los valores restantes. Así pues, en “liver” vemos que el bigote inferior se solapa con la caja. Esto es porque el valor mínimo es 0, al igual que el primer cuartil, indicándonos que al menos el 25 % de los datos tienen valor 0.

### 3.2.3. Ejercicios

- Utilizando los datos de Marioni y la función `apply` de R, calcula la mediana de los datos de expresión de cada gen en cada tejido.

*Solución:*

```
> head(marioni)
> median.kidney = apply(marioni[,grep("Kidney", colnames(marioni))], 1, median)
> median.liver = apply(marioni[,grep("Liver", colnames(marioni))], 1, median)
```

- Representa mediante un `barplot` la desviación típica de la expresión génica en cada réplica de riñón para los datos de Marioni. Pinta cada barra de un color diferente y sin borde.

*Solución:*

```

> sd.kidney = apply(marioni[,grep("Kidney", colnames(marioni))], 2, sd)
> barplot(sd.kidney, names.arg = paste("R", 1:5, sep = ""), col = 2:6,
+ border = FALSE, main = "Kidney")

```

3. Genera unos datos ficticios, cuya primera columna sean instantes de tiempo de 0' a 180' (a intervalos de 30'), y la segunda y tercera columna valores de expresión del gen X en las condiciones A y B (valores discretos generados aleatoriamente entre 0 y 20). Representa en un gráfico los perfiles de expresión del gen en ambas condiciones.

*Solución:*

```

> tiempo = seq(0,180,30)
> ficticios = data.frame("tiempo" = tiempo,
+ "A" = sample(0:20, length(tiempo), replace = TRUE),
+ "B" = sample(0:20, length(tiempo), replace = TRUE))
> plot(ficticios$tiempo, ficticios$A, type = "b", main = "Gen X",
+ xlab = "", ylab = "expresión", pch = 20, col = 2, lwd = 3,
+ ylim = range(ficticios[,-1]), xaxt = "n")
> lines(ficticios$tiempo, ficticios$B, type = "b", pch = 20, col = 4, lwd = 3)
> axis(side = 1, at = tiempo, labels = paste(tiempo, "'", sep = ""))
> legend("topleft", c("A", "B"), col = c(2,4), lwd = 3)

```

### 3.3. Probabilidad

#### 3.3.1. Definición y propiedades

La *probabilidad* es una medida numérica del grado de certidumbre de que ocurra un determinado suceso, y es un valor comprendido entre 0 y 1. Cuanto más cercano a 1 esté el valor de la probabilidad del suceso A, menor incertidumbre tendremos de que dicho suceso tenga lugar. Así pues,  $P(A) = 0$  indica que A es un *suceso imposible* y, si  $P(A) = 1$ , A es un *suceso seguro*.

Una *estimación* (frecuentista) de la probabilidad de un suceso es la proporción de veces que se observa dicho suceso a lo largo de muchas repeticiones de un experimento.

Sean A y B dos sucesos cualesquiera. Algunas de las propiedades más importantes de la probabilidad de un suceso son:

- Si  $\bar{A}$  es el suceso complementario (contrario) a A,  $P(\bar{A}) = 1 - P(A)$ .
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ , es decir, la probabilidad de que ocurra A o B es igual a la suma de la probabilidad de que suceda A más la probabilidad de que suceda B menos la probabilidad de que sucedan ambos simultáneamente.
- $P(A|B) = P(A \cap B)/P(B)$ , lo que significa que la probabilidad de que ocurra A sabiendo que B ha ocurrido, es igual a la probabilidad de que ambos ocurran dividida por la probabilidad de B.
- Teorema de Bayes:  $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Cuando  $P(A|B) = P(A)$  (o equivalentemente  $P(A \cap B) = P(A)P(B)$ ), se dice que los sucesos A y B son independientes, ya que la probabilidad de que suceda uno de ellos no afecta a la probabilidad de que suceda el otro.

Se suele utilizar el término inglés “*odds*” para designar el cociente entre la probabilidad favorable a un suceso y la de su suceso contrario. Por ejemplo, una probabilidad de 0.75 corresponde a un *odds* de

75:25 ó de 3 (ó de 3 contra 1). Esta es la fórmula que relaciona *odds* y probabilidades:

$$odds = \frac{probabilidad}{1 - probabilidad} \quad (3.4)$$

Una variable aleatoria lleva asociada una determinada *función* o *distribución de probabilidad*. Para las variables discretas, la función de probabilidad indica cuál es la probabilidad de que la variable tome un valor concreto. En cambio, para las variables continuas, se define la función de densidad de probabilidad, que nos permite obtener la probabilidad de que la variable tome valores dentro de un intervalo determinado.

Aunque hay tantas distribuciones de probabilidad como variables aleatorias podamos definir, hay algunas distribuciones que se repiten con frecuencia en la naturaleza. Estas distribuciones han sido estudiadas en profundidad, modelizadas matemáticamente y muchos de los métodos estadísticos que veremos se basan en ellas. A continuación, presentamos las distribuciones de probabilidad más utilizadas en biología, tanto para variables discretas como para variables continuas, así como otras distribuciones que se derivan de ellas.

### 3.3.2. Distribuciones discretas

#### Distribución binomial

Supongamos un experimento con dos posibles resultados: “éxito” o “fracaso”. Sea  $p$  la probabilidad de éxito y, por tanto,  $q = 1 - p$  la de fracaso. Repetimos este experimento  $n$  veces y las distintas repeticiones son independientes entre sí. Sea  $X$  la variable aleatoria definida como el número de éxitos que obtenemos en las  $n$  repeticiones del experimento.  $X$  puede tomar valores enteros entre 0 y  $n$ . Una variable aleatoria así definida sigue una *distribución binomial* con parámetros  $n$  y  $p$ . Conocidos estos parámetros, su función de probabilidad se puede calcular de la siguiente forma:

$$f(x) = P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}$$

Por ejemplo, si la probabilidad de que un determinado gen presente una mutación es del 1% y se ha realizado un estudio con 2000 individuos seleccionados al azar, el número de individuos que presentarán una mutación en dicho gen es una variable aleatoria con distribución binomial. En este caso,  $p$  será la probabilidad de que el gen presente la mutación:  $p = 0,01$  y cada experimento consiste en observar a un individuo y determinar si su gen presenta la mutación, por lo que el número de experimentos realizados será  $n = 2000$ . Podemos suponer que los individuos son independientes entre sí. Para calcular la probabilidad de que al menos 15 individuos de los 2000 seleccionados presenten una mutación del gen en cuestión, definimos  $X =$  número de mutaciones observadas en los 2000 individuos. Así,  $X \sim B(n = 2000, p = 0,01)$ . La probabilidad que nos interesa calcular se puede escribir como  $P(X \geq 15)$ . Calcular dicha probabilidad requeriría sumar las siguientes probabilidades:  $P(X = 15) + P(X = 16) + P(X = 17) + \dots + P(X = 2000)$ . Una alternativa sería utilizar la función de distribución:  $P(X \geq 15) = 1 - P(X < 15) = 1 - P(X \leq 14)$ . Veamos cómo calcular probabilidades en R para una distribución binomial:

```
> # P(X=15)
> dbinom(15, size=2000, prob=0.01)
[1] 0.05151763
> # P(X<=14)
> pbinom(14, size=2000, prob=0.01)
[1] 0.1036992
```

```

> # P(X>=15) = 1-P(X<=14)
> 1-pbinom(14, size=2000, prob=0.01)
[1] 0.8963008

```

En una distribución binomial de parámetros  $n$  y  $p$ , la media es  $\mu = np$  y la varianza  $\sigma^2 = npq$ .

## Distribución de Poisson

Cuando, en una distribución binomial, el parámetro  $n$  se hace cada vez mayor ( $n \rightarrow \infty$ ) y  $p$  se hace cada vez menor ( $p \rightarrow 0$ ), siendo el producto  $np$  constante, se obtiene la *distribución de Poisson* de parámetro  $\lambda = np$ :

$$f(x) = P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!} \quad (3.5)$$

En general, una variable aleatoria que mida el número de veces que un suceso ocurre en una determinada unidad (de tiempo, espacio, etc.) seguirá una distribución de Poisson de parámetro  $\lambda$ , siempre y cuando el número de ocurrencias del suceso no dependa del tamaño o número de unidades. Un ejemplo de una variable Poisson sería el número de lecturas que mapean a un determinado gen en un experimento de RNA-seq [13].

En la distribución de Poisson, tanto su media como su varianza son iguales a  $\lambda$ . Una propiedad importante de esta distribución es que la suma de variables Poisson independientes es una variable Poisson con media igual a la suma de las respectivas medias de las variables.

Las funciones en R para trabajar con la distribución de Poisson son ‘dpois’, ‘ppois’ , ‘qpois’ y ‘rpois’, que se utilizan para obtener la función de probabilidad, la función de distribución, los cuantiles de la distribución y para generar valores aleatorios de una distribución Poisson, respectivamente.

## Distribución binomial negativa

Supongamos que tenemos un experimento con dos posibles resultados (éxito y fracaso), siendo  $p$  la probabilidad de éxito. Repetimos el experimento hasta conseguir (por vez primera) un número de éxitos  $k$  determinado. Una *variable binomial negativa* es aquella que cuenta el número de veces que se tiene que repetir el experimento hasta conseguir los  $k$  éxitos. Cuando  $k = 1$ , esta distribución también se conoce con el nombre de *distribución geométrica*. La función de probabilidad de una variable  $X \sim BN(k, p)$  viene dada por:

$$f(x) = P(X = x) = \binom{x-1}{k-1} p^k (1-p)^{x-k} \quad (3.6)$$

La distribución binomial negativa también es útil para modelizar conteos tipo Poisson cuando no podemos asumir que la media y la varianza son iguales. Por ejemplo, se ha comprobado que cuando tenemos experimentos de RNA-seq con réplicas biológicas, la distribución del número de lecturas que mapean a un determinado gen a lo largo de las réplicas se ajusta más a una distribución binomial negativa que a una Poisson [13].

En R, los cálculos relativos a esta distribución se pueden hacer con las funciones ‘dnbinom’, ‘pnbinom’, ‘qnbinom’ y ‘rnbinom’.

### 3.3.3. Distribuciones continuas

#### Distribución normal

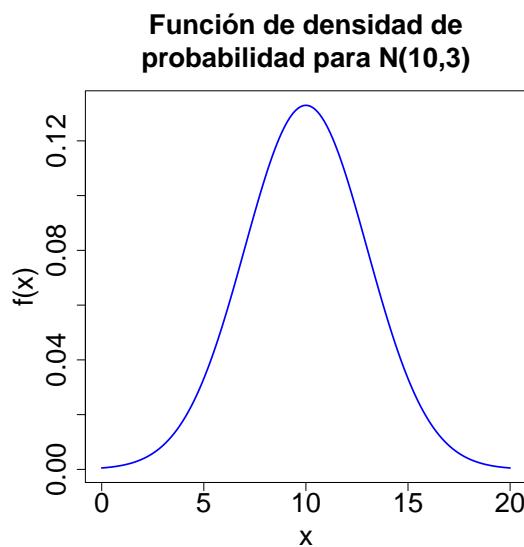
En general, observaciones repetidas del mismo fenómeno varían debido al error experimental. En concreto, varían alrededor de un valor central siguiendo una distribución aproximadamente simétrica, de forma que pequeñas desviaciones ocurren con mayor frecuencia que desviaciones grandes. La *distribución normal o gaussiana* es una importante distribución teórica que modeliza esta situación y que representa un papel fundamental en la estadística. Los parámetros que la definen son la media y la desviación típica, siendo la media el eje de simetría de la distribución. La función de densidad de probabilidad de una variable normal  $X \sim N(\mu, \sigma)$  es:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad (3.7)$$

La Figura 3.5 muestra un ejemplo de distribución normal. Los comandos de R utilizados para representar esta gráfica son los siguientes:

```
\begin{lstlisting}
> plot(seq(0,20,0.1),
+       dnorm(seq(0,20,0.1), mean = 10, sd = 3),
+       type = "l", lwd = 2, col = "blue",
+       main = "Función de densidad de probabilidad para N(10,3)",
+       xlab = "x", ylab = "f(x)"
```

En primer lugar, se han generado los valores  $x$  para los que se va a dibujar la curva de densidad. Para ello, se ha utilizado la función ‘seq’, que, en este caso, genera valores secuencialmente desde 0 hasta 20 a intervalos de 0.1. Con ‘dnorm’, se obtiene el valor de la función de densidad para dichos valores. Los argumentos de la función ‘plot’ que se han utilizado son ‘type = ‘‘l’’’ para indicar que queremos dibujar una línea (sin puntos), ‘lwd = 2’ para indicar que el grosor de la línea debe ser 2. El resto de argumentos ya han sido explicados anteriormente (ver Subsección 3.2.2).



**Figura 3.5:** Ejemplo de distribución normal ('dnorm').

Cuando la media es 0 y la desviación típica es 1, la distribución normal se denomina *distribución normal tipificada*. Dada una variable aleatoria  $X$  que sigue una distribución normal,  $X \sim N(\mu, \sigma)$ , podemos obtener una nueva variable normal tipificada  $Z$  de la siguiente forma:  $Z = \frac{X - \mu}{\sigma}$ .

Una propiedad de la distribución normal es que tanto la suma como la resta de variables normales independientes sigue siendo una variable normal.

El *Teorema Central del Límite* juega un papel clave en que la probabilidad normal sea tan común y aparezca en numerosas ocasiones. Este teorema afirma que bajo ciertas condiciones, una suma de variables aleatorias, todas ellas con la misma distribución de probabilidad (aunque esta sea desconocida), tiene una distribución que se aproxima a la normal conforme aumenta el número de sumandos.

Una de las consecuencias del Teorema Central del Límite es que, por ejemplo, una variable con distribución binomial  $B(n, p)$  se puede considerar una variable con distribución normal cuando  $n \rightarrow \infty$ . Lo mismo sucede con la distribución de Poisson, para valores grandes de  $\lambda$  ( $\lambda \rightarrow \infty$ ), se aproxima a una distribución normal.

Otra consecuencia importante del Teorema Central del Límite es que, dada una variable  $X$  de distribución desconocida con media  $\mu$  y desviación típica  $\sigma$  y una muestra de tamaño  $n$  suficientemente grande, la correspondiente media muestral seguirá una distribución normal:  $\bar{x} \sim N(\mu, \frac{\sigma}{\sqrt{n}})$ .

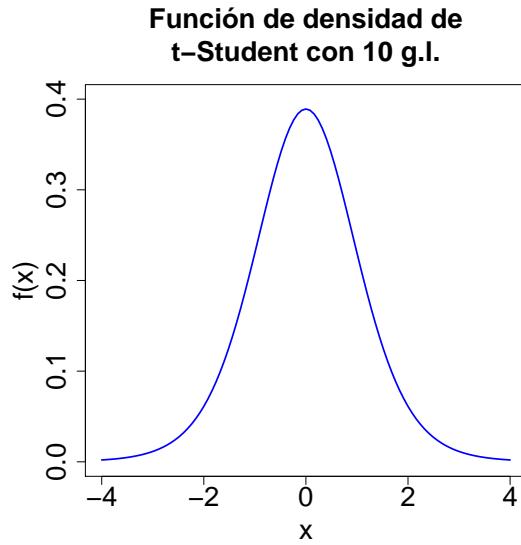
**Z-score** El *Z-score* de una observación  $x$  se define como:  $Z = \frac{x - \mu}{\sigma}$  y es el número de desviaciones típicas que ese dato dista de la media. Esta conversión de los datos se denomina normalización o estandarización. La  $Z$  proviene del hecho de que una variable normal tipificada o estandarizada (con media 0 y desviación típica 1) se denota a menudo con  $Z$  y se calcula de la misma forma. Así, si  $X \sim N(\mu, \sigma)$ , entonces  $Z \sim N(0, 1)$ . Sin embargo, los *Z-scores* también pueden definirse en ausencia de normalidad y son útiles para hacer comparables valores de variables con distinto rango y/o distribución.

En bioinformática, los programas tipo BLAST utilizan los *Z-scores* para determinar si dos secuencias son homólogas o bien su similitud se debe al azar. Así, por ejemplo, el valor  $x$  sería el valor de alineamiento entre ambas secuencias. Para obtener el *Z-score*, permutan una o ambas secuencias un gran número de veces, calculando cada vez el valor de alineamiento. Se calcula la media y desviación típica de estos valores de alineamiento y se utilizan para obtener el *Z-score*. Cuanto mayor sea éste, mayor será la probabilidad de que las secuencias sean homólogas ya que más alejado estará su valor de alineamiento de los valores esperados por azar. Así, los *E-values* indican el número de casos en los que, por azar, se obtendría un mejor valor de alineamiento que el obtenido para el par de secuencias comparadas. Por tanto, cuando más extremo sea el *Z-score*, menor será el *E-value*. Para más detalle consultar Significación estadística y *E-value* en el Capítulo 8.

## Distribuciones derivadas de la normal

Gracias al Teorema Central del Límite, la distribución normal cobra una gran importancia a la hora de modelizar el comportamiento de variables muy diversas. Existen muchos métodos estadísticos que se basan, por tanto, en la distribución normal o en distribuciones derivadas a partir de ella, bien al variables normales o al realizar operaciones con ellas. A continuación presentamos algunas de estas distribuciones.

**Distribución “*t de Student*”** Cuando la desviación típica poblacional es desconocida y queremos tipificar nuestra variable  $X$ , podemos utilizar la desviación típica muestral  $s$ . En ese caso, la variable definida como  $t = \frac{X - \mu}{\frac{s}{\sqrt{n}}}$  no sigue una distribución normal, sino una distribución de probabilidad conocida como “*t de Student*”. Esta distribución también es simétrica, tiene media 0 y el único parámetro que determina su forma son los grados de libertad. En la Figura 3.6 se representa una distribución *t* de Student con 10 grados de libertad.



**Figura 3.6:** Ejemplo de distribución ‘*t de Student*’ (‘dt’).

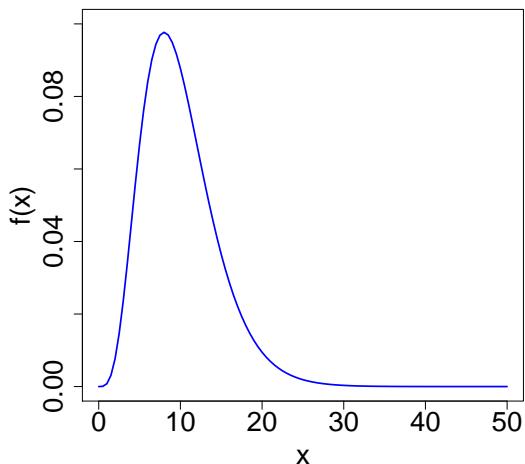
En concreto, si queremos calcular probabilidades para la media muestral  $\bar{x}$ , pero desconocemos la desviación típica poblacional  $\sigma$ , podemos hacerlo con una distribución *t* de Student. Si el tamaño muestral es  $n$ , los grados de libertad de la distribución serán  $n - 1$ :  $\frac{\bar{x} - \mu}{s/\sqrt{n}} \sim t_{n-1}$ .

**Distribución  $\chi^2$**  La *distribución  $\chi^2$*  surge a partir de la necesidad de obtener la distribución de probabilidad de la varianza muestral  $s^2$ , aunque también tiene otros usos. Una variable  $\chi^2$  se define como la suma de  $k$  variables normales, tipificadas y elevadas al cuadrado. El número de variables independientes sumadas constituye los grados de libertad de la variable  $\chi^2$ , que es el parámetro que la define. Obviamente, una variable  $\chi^2$  siempre toma valores positivos y no es una distribución simétrica. Véase en la Figura 3.7 un ejemplo de esta distribución.

La relación entre la varianza muestral y la distribución  $\chi^2$ , para una muestra de tamaño  $n$ , es la siguiente:  $\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$ .

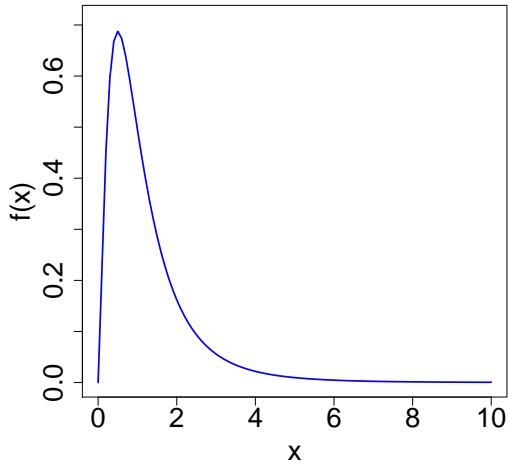
**Distribución F** Dadas dos variables  $\chi^2$  independientes con  $n_1$  y  $n_2$  grados de libertad respectivamente, se define la *distribución F* el cociente de dichas variables divididas por sus respectivos grados de libertad. Así pues, dadas dos muestras independientes de tamaños  $n_1$  y  $n_2$  es fácil observar que  $\frac{s_1^2/\sigma_1^2}{s_2^2/\sigma_2^2} \sim F_{n_1-1, n_2-1}$ . Una variable con distribución F toma siempre valores positivos y da lugar a curvas no simétricas (ver ejemplo en la Figura 3.8).

### Función de densidad de chi-cuadrado con 10 g.l.



**Figura 3.7:** Ejemplo de distribución  $\chi^2$  ('dchisq').

### Función de densidad de F con 5 y 10 g.l.



**Figura 3.8:** Ejemplo de distribución F ('df').

### Distribución de Gumbel (o de valores extremos)

La *distribución de Gumbel* se utiliza para modelizar la distribución del máximo o el mínimo (es decir, los valores extremos) de una variable y viene caracterizada por dos parámetros: la moda  $\mu$  y  $\beta$ , que es un parámetro de escala. Su función de distribución viene dada por:

$$F(x) = e^{-e^{-(x-\mu)/\beta}} \quad (3.8)$$

La media de la distribución es igual a  $\mu + \gamma\beta$ , donde  $\gamma$  es la *constante de Euler-Mascheroni* ( $\gamma \approx 0,5772156649015328606$ ).

Un ejemplo de uso de la distribución de Gumbel en bioinformática es la distribución que siguen los *scores* de los alineamientos locales entre dos secuencias al azar cuando se aplica un algoritmo Smith-Waterman.

### 3.3.4. Ejercicios

1. Utilizando la función *dt*, escribe los comandos necesarios para obtener la gráfica de la Figura 3.6.

*Solución:*

```
> plot(seq(-5,5,0.1), dt(seq(-5,5,0.1), df = 10), type = "l", lwd = 2,
+ main = "Función de densidad de t-Student con 10 g.l.",
+ xlab = "x", ylab = "f(x)", col = "blue")
```

2. Calcula  $P(X > 2)$ , sabiendo que  $X \sim t_5$ .

*Solución:*

```
> 1 - pt(q = 2, df = 5)
[1] 0.05096974
```

3. Calcula  $P(s^2 > 5)$ , sabiendo que la varianza poblacional  $\sigma^2$  es igual a 3 y el tamaño de la muestra es  $n = 30$ .

*Solución:*

Sabemos que  $(n - 1)s^2/\sigma^2 \sim \chi_{n-1}^2$ . Por tanto,  $P(s^2 > 5) = P((n - 1)s^2/\sigma^2 > (30 - 1)5/3) = P(\chi_{29}^2 > 48.33)$ . En R:

```
> 1 - pchisq(q = 48.33, df = 29)
[1] 0.01358054
```

4. Obtén el percentil 95 de una distribución *F* con 12 y 20 grados de libertad.

*Solución:*

```
> qf(p = 0.95, df1 = 12, df2 = 20)
[1] 2.277581
```

5. Ilustrar gráficamente que una distribución de Poisson con parámetro  $\lambda = 20$  se aproxima a una distribución normal.

*Solución:*

```
> x1 = 0:40 # valores discretos para la variable Poisson
> x2 = seq(0,40,0.01) # valores continuos para la variable normal
> plot(x1, dpois(x1, lambda = 20), type = "h", col = 2, lwd = 5,
+ xlab = "", ylab = "")
> lines(x2, dnorm(x2, mean = 20, sd = sqrt(20)), col = 4, lwd = 2)
> legend("topright", c("Poisson", "Normal"), col = c(2,4), lwd = 3, bty = "n")
```

6. Genera aleatoriamente 1000 valores de una distribución normal tipificada y comprueba que el histograma obtenido a partir de ellos tiene forma de campana de Gauss.

*Solución:*

```
> valores = rnorm(1000, mean = 0, sd = 1)
> hist(valores, main = "Normal", xlab = "", col = "grey")
```

## 3.4. Inferencia estadística

Cuando queremos analizar una o más características (variables aleatorias) de una población pero no es posible recoger información de todos los individuos u objetos de dicha población, nos tenemos que limitar al estudio de un subconjunto de ella (muestra). En ese caso, se pueden estimar los valores de los parámetros de la población (por ejemplo su media o su desviación típica) mediante los estadísticos muestrales. Esto se conoce como *estimación puntual*.

Sin embargo, resulta más interesante no sólo aportar un estimación puntual de un parámetro poblacional desconocido, sino medir la probabilidad del error que se puede estar cometiendo al extender la información contenida en la muestra a toda la población objeto del estudio. Aquí entra en juego la *inferencia estadística*, tanto la clásica como la bayesiana. Veremos en esta sección cómo la inferencia estadística clásica o frecuentista resuelve el problema de la estimación por intervalos de confianza o los contrastes de hipótesis.

Además, los métodos de inferencia pueden clasificarse en dos grandes grupos: pruebas paramétricas y pruebas no paramétricas. Las *pruebas paramétricas* son aquellas que parten de la suposición de que la población objeto del estudio sigue una determinada distribución de probabilidad. En el caso de las *pruebas no paramétricas*, no se requieren suposiciones sobre la distribución de probabilidad de la variable estudiada. En igualdad de condiciones y suponiendo que se cumple la suposición paramétrica, las pruebas paramétricas son más potentes que las no paramétricas, esto es, la probabilidad de error es menor.

### 3.4.1. Estadística paramétrica

#### Intervalos de confianza

Se pueden construir intervalos de confianza para distintos parámetros de una población. Tomaremos como ejemplo para explicar el concepto de intervalo de confianza, el de la media poblacional.

Un intervalo de confianza para la media poblacional (que tiene un valor fijo pero desconocido) es un intervalo que contiene, con una probabilidad muy elevada, a este parámetro. Dicha probabilidad es el *nivel de confianza*. Así pues, el intervalo de confianza obtenido dependerá de los valores de los estadísticos muestrales, es decir, de la media y desviación típica muestral en nuestro caso. Por tanto, para cada muestra seleccionada a partir de una misma población, el intervalo de confianza calculado puede variar.

Si la variable aleatoria  $X$  estudiada sigue una distribución normal (por tanto, estamos ante un método de estadística paramétrica) y su desviación típica es desconocida, el intervalo de confianza para la media poblacional se puede calcular de la siguiente forma:  $[\bar{x} - t_{n-1}(\alpha/2) \frac{s}{\sqrt{n}}; \bar{x} + t_{n-1}(\alpha/2) \frac{s}{\sqrt{n}}]$ , donde  $n$  es el tamaño muestral y  $t_{n-1}(\alpha/2)$  es el valor crítico de una distribución  $t$  de Student para  $\alpha/2$ , es decir, es un valor tal que  $P(t_{n-1} > t_{n-1}(\alpha/2)) = \alpha/2$ , siendo  $1 - \alpha$  el nivel de confianza del intervalo. Si, por ejemplo,  $\alpha = 0,05$ , la probabilidad de que un intervalo construido a partir de una muestra aleatoria seleccionada de la población contenga al verdadero valor de la media poblacional será de 0,95.

Por lo tanto, al calcular un intervalo de confianza, estamos proporcionando un rango de valores con una probabilidad elevada de contener al parámetro poblacional que pretendemos estimar, en lugar de ofrecer únicamente una estimación puntual.

## Contrastes de hipótesis

En algunos estudios, no es tan importante estimar el valor de un parámetro o parámetros de la población, sino determinar el *grado de confianza* en la veracidad de una hipótesis sobre dicho parámetro. Veamos un ejemplo:

Un problema muy común en bioinformática es la comparación de dos poblaciones (dos tratamientos, un grupo control y un grupo de enfermos, dos tejidos, etc.). Supongamos que queremos evaluar si el nivel medio de expresión de un gen cambia entre un grupo control (personas sanas) y un grupo de enfermos (personas diabéticas) de forma estadísticamente significativa o si, por el contrario, las diferencias observadas en la expresión entre ambos grupos se deben al azar. Así pues, la variable aleatoria  $X$  estudiada será la expresión del gen y nuestro objetivo es determinar si la media de  $X$  es la misma para las dos poblaciones comparadas:  $\mu_{sanos} = \mu_{enfermos}$ ?

Al resolver un *contraste de hipótesis*, pretendemos evaluar cuán creíble es una afirmación sobre la población a partir de los valores observados en la muestra. En un contraste de hipótesis, como su nombre indica, se contrastan dos hipótesis: la *hipótesis nula*  $H_0$  y la *hipótesis alternativa*  $H_1$ . La mayoría de contrastes de hipótesis que hacen referencia a un único parámetro  $\theta$  de la población se pueden clasificar en *bilaterales* (si  $H_1 : \theta \neq \theta_0$ ) o *unilaterales* (si  $H_1 : \theta < \theta_0$  o bien  $H_1 : \theta > \theta_0$ ).

Sigamos con el ejemplo anterior y supongamos que disponemos de una muestra de cada población: 10 pacientes diabéticos y 10 sanos (grupo control). Supongamos también que se ha medido la expresión del gen utilizando la técnica de *microarrays* y los datos se han procesado y normalizado convenientemente, por lo que podemos asumir que la expresión del gen sigue una distribución normal. En este caso, tiene sentido plantear un contraste de hipótesis bilateral, siendo  $H_0 : \mu_{sanos} = \mu_{enfermos}$ . La hipótesis nula engloba la afirmación que supondremos cierta mientras no se pueda demostrar lo contrario. Para el ejemplo que nos ocupa, supondremos que el gen no ha cambiado de expresión entre ambos grupos, salvo que haya mucha evidencia en los datos de lo contrario.

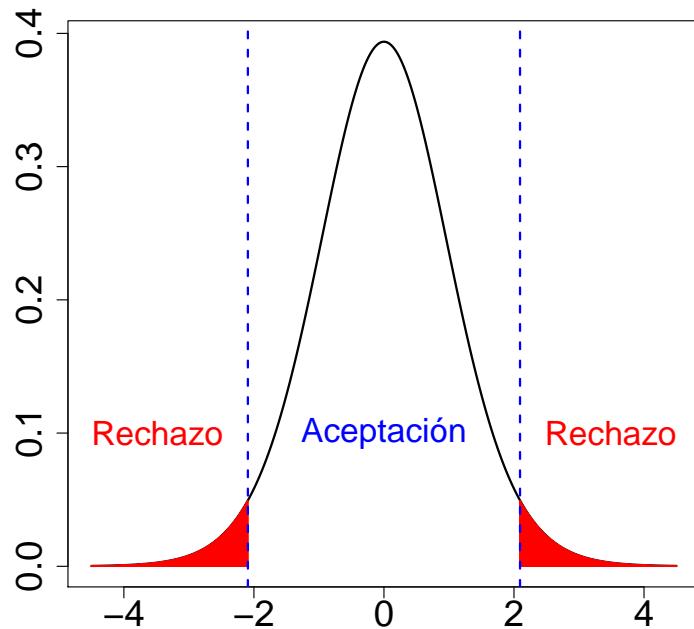
Cuando se resuelve un contraste de hipótesis (lo que significa decidir si rechazamos  $H_0$  o si la admitimos como cierta porque no podemos demostrar lo contrario), se pueden cometer dos tipos de error: rechazar  $H_0$  cuando es cierta (“*falso positivo*”) o aceptarla cuando es falsa (“*falso negativo*”).

- La *probabilidad de error de tipo I o riesgo de primera especie* se define como:  
$$\alpha = P(\text{rechazar } H_0 \mid H_0 \text{ es cierta})$$
- La *probabilidad de tipo II o riesgo de segunda especie* sería:  
$$\beta = P(\text{aceptar } H_0 \mid H_0 \text{ es falsa})$$

A  $1 - \beta$  se le denomina *potencia* del test estadístico y representa la capacidad que tiene el test de detectar los verdaderos positivos.

El objetivo es, pues, establecer una regla de decisión que permita resolver el contraste de hipótesis minimizando los errores. Se considera más grave el error de tipo I, por lo que las reglas de decisión se basan generalmente en establecer un límite para este tipo de error (*nivel de significación*), al tiempo que tratan de minimizar el error de tipo II (es decir, aumentar la potencia del test). Para ello, se define un estadístico que resuma la información de la muestra y nos ayude a discriminar entre los casos en los que se cumple la hipótesis nula y los que no. Si queremos realizar un contraste de hipótesis sobre la media de una población con varianza desconocida ( $H_0 : \mu = \mu_0$ ) y tenemos una muestra de tamaño  $n$ , elegiríamos el estadístico de contraste  $t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$ , que sigue una distribución  $t$  de Student con  $n - 1$  grados de libertad cuando la hipótesis nula es cierta. Por tanto, cuanto mayor sea el valor de  $t$ , más distinta será la media muestral del valor  $\mu_0$ . Si esto sucede, podría ser que la hipótesis nula fuera cierta

pero que se obtuvo por azar una media muestral muy diferente a la poblacional. Pero también podría ser debido a que realmente la hipótesis nula no es cierta y esto es, de hecho, lo que vamos a creer. Pero, ¿cuánto debe valer el estadístico  $t$  para que decidamos rechazar  $H_0$ ? La región de aceptación y la de rechazo vendrán determinadas por el nivel de significación  $\alpha$ , según la Figura 3.9 (para un caso en el que el tamaño de la muestra sea  $n = 20$ ). Así, los límites de la región de aceptación vendrán marcados por el valor de la  $t$  de Student que deje una probabilidad de  $\alpha$  en las colas ( $\alpha/2$  en cada una de ellas). Cuando el valor del estadístico  $t$  obtenido a partir de la muestra caiga en la región de rechazo, rechazaremos la hipótesis nula, y no la podremos rechazar en caso contrario.



**Figura 3.9:** La curva representa la distribución  $t$  de Student con  $n - 1$  grados de libertad. El nivel de significación  $\alpha$  se divide entre las dos colas de la distribución (áreas en rojo). El valor  $t$  que determina que la probabilidad de las colas sea  $\alpha$ , delimitando las regiones de rechazo y la región de aceptación, viene indicado con las líneas azules.

Se define el *p-valor* como el nivel de significación mínimo que nos llevaría a rechazar la hipótesis nula para nuestros datos. En el ejemplo anterior, se calcularía el p-valor como  $P(|t_{n-1}| > |t|)$ . Por tanto, cuando el p-valor sea menor que el nivel de significación fijado, rechazaremos la hipótesis nula.

Aprovechando el ejemplo de la expresión media de un gen en un grupo de individuos sanos y enfermos, veamos cómo resolver el contraste de hipótesis para la igualdad de medias de dos poblaciones normales mediante R:

```
# valores de expresión del gen en cada grupo
> sanos <- c(2.0, 0.4, 1.3, 0.3, 2.1, 0.9, 0.4, 1.1, 2.7, 1.7)
> enfermos <- c(4.4, 5.5, 5.2, 5.0, 4.8, 5.0, 3.6, 5.6, 5.1, 4.9)
# t-test
> t.test(x = enfermos, y = sanos, alternative = "two.sided",
+         mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95)
```

Welch Two Sample t-test

```
data: enfermos and sanos
t = 11.436, df = 16.076, p-value = 3.91e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.949214 4.290786
sample estimates:
mean of x mean of y
 4.91      1.29
```

Dado que se trata de un contraste bilateral, hemos elegido ‘alternative = "two.sided"’. La opción ‘mu = 0’ significa que estamos contrastando la hipótesis nula de que la diferencia entre las medias de ambos grupos es 0. Hemos indicado ‘paired = FALSE’ porque se trata de muestras independientes y no datos apareados. Serían datos apareados si, por ejemplo, hubiéramos medido la expresión del gen en el mismo individuo antes y después de un tratamiento, ya que por cada individuo tendríamos dos valores a comparar que estarían “emparejados” por pertenecer al mismo individuo. En el caso de muestras independientes, el estadístico utilizado para el test de la  $t$  difiere si las varianzas poblacionales son iguales. Se podría hacer un test  $F$  (‘var.test’ en R) para comparar las varianzas poblacionales de sanos y enfermos y, si no podemos rechazar que sean iguales, cambiar la opción ‘var.equal = FALSE’ por ‘var.equal = TRUE’. Por último, al indicar ‘conf.level = 0.95’ estamos pidiendo también el intervalo de confianza para la diferencia de medias  $\mu_{enfermos} - \mu_{sanos}$  con un nivel de confianza del 95 %.

Dado que el p-valor asociado a este contraste es igual a  $3.91e - 09$ , podemos rechazar la hipótesis nula para un nivel de significación del 5 % (e incluso mucho menor, mientras no rebasemos el p-valor obtenido). Así, la expresión media significativamente diferente entre ambos grupos. El intervalo de confianza obtenido para dicha diferencia de medias sería [2,949214; 4,290786].

## Tests paramétricos en R

Función en R	Contraste de hipótesis	Descripción
t.test	$H_0 : \mu = \mu_0; H_0 : \mu_1 = \mu_2$	Test para la media de una población normal o para comparar medias de dos poblaciones normales, tanto para datos apareados o muestras independientes (con y sin igualdad de varianzas).
var.test	$H_0 : \sigma_1^2 = \sigma_2^2$	Comparación de varianzas de dos poblaciones normales (F-test).
bartlett.test	$H_0 : \sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2$	Comparación de varianzas en poblaciones normales.
cor.test	$H_0 : \rho = 0$	Contrasta si existe correlación lineal (Pearson) entre dos variables.
poisson.test	$H_0 : \lambda = \lambda_0; H_0 : \lambda_1 = \lambda_2$	Contrastes sobre el parámetro $\lambda$ de la distribución de Poisson en una o dos poblaciones.
binom.test	$H_0 : \pi = \pi_0$	Contraste sobre una proporción binomial.
prop.test	$H_0 : \pi_1 = \pi_2$	Contraste sobre dos proporciones binomiales.

**Tabla 3.1:** Resumen de algunas funciones en R para resolver contrastes de hipótesis mediante métodos paramétricos.

### 3.4.2. Estadística no paramétrica

Como se ha indicado anteriormente, los test paramétricos son más potentes que los no paramétricos siempre y cuando se cumplan los requisitos para aplicarlos. Cuando esto no sucede, es más correcto aplicar un test no paramétrico. Además, algunos contrastes de hipótesis se resuelven necesariamente mediante un test no paramétrico. Este es el caso, por ejemplo, de los tests de bondad de ajuste (Kolmogorov-Smirnov o  $\chi^2$ , entre otros), que sirven para determinar a partir de una muestra, si los datos provienen de una distribución de probabilidad determinada. El test  $\chi^2$  también se utiliza para determinar si dos variables son independientes a partir de una tabla de contingencia.

En los *tests no paramétricos* también se genera como resultado un p-valor que nos permitirá decidir si aceptamos o no la hipótesis nula contrastada en cada caso. La Tabla 3.2 resume algunos de los tests más utilizados para comparar los valores de los parámetros de posición a partir de una, dos o varias muestras.

Función en R	Contraste de hipótesis	Descripción
wilcox.test	$H_0 : Me_1 = Me_2$	Compara las medianas de dos poblaciones, tanto para datos apareados como para muestras independientes.
chisq.test	$H_0 : X \text{ e } Y \text{ independientes}$	Independencia entre dos variables a partir de tablas de contingencia.
chisq.test	$H_0 : X \text{ sigue distribución } f$	Test de bondad de ajuste.
fisher.test	$H_0 : X \text{ e } Y \text{ independientes}$	Independencia entre dos variables a partir de tablas de contingencia.
ks.test	$H_0 : X \text{ sigue distribución } f$	Test de bondad de ajuste de Kolmogorov-Smirnov.

**Tabla 3.2:** Resumen de algunas funciones en R para resolver contrastes de hipótesis mediante métodos no paramétricos.

### 3.4.3. Remuestreo

Los métodos descritos anteriormente para resolver un contraste de hipótesis o calcular intervalos de confianza asumen una distribución conocida para los estadísticos muestrales utilizados (en muchos casos la distribución normal o alguna de sus derivadas). Sin embargo, no siempre se cumple el requisito de normalidad de los datos o, en ocasiones, no es posible estimar la distribución de probabilidad del estadístico muestral utilizado. Como hemos dicho, una solución en estos casos sería la estadística no paramétrica, que también incluye los llamados métodos de remuestreo.

Los *métodos de remuestreo* se basan en “reutilizar” los datos en nuestra muestra para generar la distribución de probabilidad de un estadístico, en lugar de hacerlo a partir de suposiciones paramétricas. Aunque tienen un coste computacional mayor, ofrecen también mayor flexibilidad. Algunos de los métodos de remuestreo más extendidos son el “*bootstrap*”, el “*jackknife*” y los métodos de permutación.

#### *Bootstrap*

Este método consiste en estimar la distribución de un estadístico mediante la obtención de sucesivas muestras con reemplazamiento a partir de la muestra original. Normalmente, estas nuevas muestras tienen el mismo tamaño muestral que la original y, al ser un muestreo con reemplazamiento, la misma observación puede repetirse, por lo que se consiguen muestras diferentes entre sí. En cada nueva

muestra generada, se calcula el valor del estadístico cuya distribución se desea estimar. Todos los valores obtenidos a partir de los remuestreos conformarán la distribución de dicho estadístico. Se puede también aplicar el método *bootstrap* para resolver contrastes de hipótesis.

Se recomienda utilizar el procedimiento *bootstrap* cuando la distribución teórica del estadístico de contraste es desconocida o complicada de calcular o cuando el tamaño muestral es insuficiente.

La simplicidad de este método es una de sus ventajas, así como la posibilidad de aplicarlo a cualquier parámetro de la distribución, incluyendo cuantiles, proporciones, coeficientes de correlación, etc. Sin embargo, tiene tendencia a generar estimaciones optimistas.

### ***Jackknife***

El método “*jackknife*” se utiliza en inferencia para estimar el sesgo y la variabilidad de un estadístico. La idea subyacente es calcular el valor del estadístico sucesivamente con todas las observaciones de la muestras dejando fuera una (o  $k$ ) de ellas cada vez.

Una de las diferencias entre el método *jackknife* y el *bootstrap* es que este último genera resultados diferentes cada vez que se ejecuta sobre los mismos datos, mientras que con “*jackknife*” los resultados son siempre los mismos. Sin embargo, el método *bootstrap* permite no sólo estimar la variabilidad del estadístico en cuestión, sino también su distribución de probabilidad. Por otra parte, aunque el método *jackknife* suele estimar mejor la varianza del estadístico, no es recomendable usarlo para estadísticos tales como la mediana o los cuantiles de una distribución. Así pues, el método a elegir dependerá del problema a resolver.

### **Tests de permutación**

En un *test de permutación*, la distribución del estadístico de contraste bajo la hipótesis nula se obtiene calculando todos los posibles valores de dicho estadístico para cada “reorganización” de las etiquetas de cada observación, es decir, se permutando las condiciones experimentales asociadas a las distintas observaciones muestrales. Por cada permutación, obtendremos un valor para el estadístico. Con un número elevado de permutaciones, podremos estimar la distribución del estadístico a partir de todos los valores calculados. El p-valor se obtendrá dividiendo el número de valores calculados que “se alejan” más de la hipótesis nula que el valor del estadístico obtenido para la muestra sin permutar entre el número total de permutaciones realizadas.

Un requisito importante que se debe cumplir para aplicar un test de permutación es que las observaciones sean intercambiables bajo la hipótesis nula. Si no se cumple, es más recomendable utilizar como alternativa un método *bootstrap*.

#### **3.4.4. Corrección por tests múltiples**

El problema de los *test múltiples* aparece cuando contrastamos simultáneamente varias hipótesis nulas. Este problema cobra especial relevancia en el campo de la bioinformática, ya que es habitual realizar contrastes de hipótesis para miles de genes (en análisis de expresión diferencial), miles de términos GO (en análisis de enriquecimiento funcional), miles de posibles mutaciones, SNPs, etc.

Recordemos que la probabilidad de error de tipo I es la probabilidad de rechazar  $H_0$  cuando es cierta (falsos positivos). Cuanto mayor sea el número de contrastes realizados, mayor será la probabilidad

de cometer error de tipo I en alguno de ellos. Existen varias formas de generalizar el concepto de probabilidad de error de tipo I para el caso de los tests múltiples. A continuación se definen dos muy comunes:

- *Family-Wise Error Rate (FWER)*: Es la probabilidad de cometer error de tipo I en uno o más de los tests realizados. Si los tests son independientes entre sí, se calcula como  $1 - (1 - \alpha)^m$ , donde  $\alpha$  es el nivel de significación fijado en cada test y  $m$  el número de tests realizados.
- *False Discovery Rate (FDR)*: Es la proporción esperada de errores de tipo I entre todas las hipótesis nulas rechazadas ( $R$ ). Si  $V$  es el número de hipótesis nulas rechazadas sin error (“verdaderos positivos”),  $FDR = E(Q)$ , donde  $Q = V/R$  si  $R > 0$  y  $Q = 0$  si  $R = 0$ .

Así pues, para reducir el error de tipo I “generalizado” (sea cual sea la definición utilizada), una solución sería ajustar el nivel de significación  $\alpha$  fijado para cada test o, equivalentemente, corregir los p-valores obtenidos en los contrastes realizados (p-valores nominales). De este modo, se obtienen los p-valores ajustados, que serán los que utilicemos para resolver finalmente cada contraste. Ajustar los p-valores es la opción más extendida y existen muchos métodos para hacer dicha corrección. Veamos algunos ejemplos.

Si optamos por minimizar la *FWER*, el método más sencillo es la *corrección de Bonferroni*, que consiste en multiplicar los p-valores nominales por el número de tests a realizar ( $m$ ). Este tipo de corrección es muy conservadora y suele dar lugar a un mayor número de falsos negativos que otras correcciones. Algunas modificaciones menos conservadoras de este procedimiento son las propuestas por Sidak o Holm, entre otras.

Si queremos minimizar la *FDR*, podemos aplicar la *corrección de Benjamini & Hochberg* que consiste en ordenar los  $m$  p-valores obtenidos de menor a mayor y multiplicar el p-valor que ocupa la posición  $i$  por  $m/i$ . Esta corrección permite reducir el número de falsos positivos sin generar tantos falsos negativos como la corrección de Bonferroni, por ello suele ser bastante utilizada. Tanto Bonferroni como Benjamini & Hochberg requieren que los tests sean independientes. La corrección de Benjamini & Yekuteli, por ejemplo, es una modificación del procedimiento de Benjamini & Hochberg para tests dependientes.

Existen otros procedimientos más sofisticados para la corrección por test múltiples que se basan en técnicas de remuestreo como los propuestos por Westfall y Young [20], por Efron y Tusher (librería de R ‘SAM’) o por Dudoit y Van Der Laan [5].

### 3.4.5. Enfoque bayesiano

El teorema de Bayes (ver Subsección 3.3.1) nos permite calcular cómo cambia la probabilidad de un suceso  $A$  en base al conocimiento previo que tengamos ( $B$ ). La *estadística bayesiana* se basa en esta idea para estimar la distribución de probabilidad de un parámetro poblacional  $\theta$ , que ahora no se considera un valor fijo (como en estadística clásica), sino una variable aleatoria. La adaptación del teorema de Bayes a este contexto quedaría de la siguiente forma:

$$P(\theta|datos) = \frac{P(datos|\theta)P(\theta)}{P(data)} \quad (3.9)$$

$P(\theta)$  es la probabilidad *a priori*, mientras que  $P(\theta|datos)$  es la probabilidad *a posteriori*.  $P(datos|\theta)$  es la verosimilitud de los datos para un valor determinado del parámetro  $\theta$  y a  $P(data)$  se le denomina *constante de proporcionalidad* en el contexto de la estadística bayesiana.

Una de las ventajas de la estadística bayesiana sobre la estadística clásica es que permite incorporar información previa sobre el problema y no solamente los datos observados. Y quizás la parte más controvertida de la estadística bayesiana es precisamente la incorporación de este conocimiento previo a través de la distribución de probabilidad *a priori* y cómo puede afectar esto al proceso inferencial. Si no tenemos información previa o no queremos que esta afecte en modo alguno a los resultados, se pueden seleccionar distribuciones *a priori* “no informativas” o “poco informativas”, que asignarán iguales o similares probabilidades *a priori* a todos los posibles valores del parámetro.

Una vez obtenida la distribución de probabilidad *a posteriori* del parámetro estudiado (que intuitivamente representa la distribución *a priori* modificada por la información aportada por los datos), la inferencia bayesiana se basará en dicha distribución para obtener las conclusiones. Por ejemplo, una estimación para el parámetro se calcularía como la media o la mediana de la distribución *a posteriori*.

En estadística bayesiana los contrastes de hipótesis no se resuelven del mismo modo que en estadística clásica. En lugar de rechazar o no la hipótesis nula a partir del p-valor, se calcula una probabilidad asignada a cada hipótesis. Por ejemplo:

$$P(H_1|datos) = \frac{P(datos|H_1)P(H_1)}{P(datos)} \quad (3.10)$$

A partir de estas probabilidades se puede obtener el *odds-ratio a posteriori* como:

$$\frac{P(H_0|datos)}{P(H_1|datos)} = \frac{P(datos|H_0)P(H_0)}{P(datos|H_1)P(H_1)} \quad (3.11)$$

que es el producto del odds-ratio *a priori* ( $\frac{P(H_0)}{P(H_1)}$ ) por un término conocido como factor de Bayes ( $\frac{P(datos|H_0)}{P(datos|H_1)}$ ).

### 3.4.6. Ejercicios

- Realiza un test de comparación de varianzas para comprobar si las varianzas son iguales a partir de los datos del ejemplo de la Sección 3.4.1.

*Solución:*

```
> var.test(sanos, enfermos, ratio = 1, alternative = "two.sided")
F test to compare two variances

data: sanos and enfermos
F = 2.058, num df = 9, denom df = 9, p-value = 0.2973
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.5111746 8.2854386
sample estimates:
ratio of variances
2.057986
```

Según el resultado del test, no podemos rechazar la hipótesis nula de igualdad de varianzas para un nivel de significación del 5 %.

- Resuelve el contraste de hipótesis planteado en el ejemplo de la Sección 3.4.1 mediante un método no paramétrico (sin remuestreo).

*Solución:*

En lugar de hacer un test de comparación de medias, plantearemos un test de comparación de medianas. En R:

```
> wilcox.test(sanos, enfermos, alternative = "two.sided", paired = FALSE)

Wilcoxon rank sum test with continuity correction

data: sanos and enfermos
W = 0, p-value = 0.0001806
alternative hypothesis: true location shift is not equal to 0
```

El p-valor nos indica que podemos rechazar la hipótesis nula para un nivel de significación del 5%.

3. Resuelve el contraste de hipótesis planteado en el ejemplo de la Figura 3.9 mediante el método *bootstrap* y mediante un test de permutación a partir de la función *sample* de R. Utiliza el estadístico  $t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$  y  $R = 1000$  permutaciones o *bootstraps*.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

*Solución:*

```
# Definimos el estadístico t
> miT = function(x,y) { (mean(x)-mean(y)) / sqrt((var(x)+var(y))/10) }
> Tcalc = miT(sanos, enfermos)
> miR = 1000
> # Bootstrap
> miNULL = sapply(1:miR, function (x) {
+ mipermu = sample(c(sanos, enfermos), replace = TRUE)
+ miT(mipermu[1:10],mipermu[11:20])
+ })
> miPval = length(which(abs(miNULL) > abs(Tcalc))) / miR
> miPval
[1] 0
> # Test de permutación
> miNULL = sapply(1:miR, function (x) {
+ mipermu = sample(c(sanos, enfermos), replace = FALSE)
+ miT(mipermu[1:10],mipermu[11:20])
+ })
> miPval = length(which(abs(miNULL) > abs(Tcalc))) / miR
> miPval
[1] 0
```

Según el método *bootstrap* no podemos rechazar la igualdad de medias para un nivel de significación del 5%, mientras que en el test de permutación sí la rechazaríamos.

4. Utilizando una transformación logarítmica de los datos de Marioni (y suponiendo normalidad de los datos transformados y varianzas distintas), contrasta la igualdad de medias de expresión entre los dos tejidos para cada gen, haciendo la corrección por test múltiples mediante el método de Bonferroni.

*Solución:*

```
# Quitamos los genes no expresados en ninguna condición
# y hacemos la transformación logarítmica
> marioni.log = marioni[,-1]
> rownames(marioni.log) = marioni[,1]
> marioni.log = marioni.log[which(rowSums(marioni.log) > 0),]
> marioni.log = log2(1 + marioni.log)
```

```

# Hacemos un t.test para cada gen
# El t.test da error si los valores son constantes dentro de una condición
# por lo que definimos como NA los p-valores en estos casos
> mispval = NULL
> for(i in 1:nrow(marioni.log)) {
+ if (sd(as.numeric(marioni.log[i,1:5])) == 0 |
+ sd(as.numeric(marioni.log[i,6:10])) == 0 ) {
+ mispval = c(mispval, NA)
+ } else {
+ mispval = c(mispval, t.test(marioni.log[i,1:5], marioni.log[i,6:10],
+ alternative = "two.sided", paired = FALSE,
+ var.equal = FALSE)$"p.value")
+ }
+ }
> names(mispval) = rownames(marioni.log)
# Quitamos los valores NA antes de ajustar por tests múltiples
> mispval = na.omit(mispval)
# Corrección por tests múltiples
> misadjpval = p.adjust(mispval, method = "bonferroni")
# Genes diferencialmente expresados entre tejidos para alfa=0.05
> misDEG = names(which(misadjpval < 0.05))
> length(misDEG)
[1] 2128

```

## 3.5. Modelos lineales

En el apartado anterior, nos hemos centrado principalmente en la parte de la inferencia estadística dedicada a comparar dos poblaciones (por ejemplo, cuando se compara la expresión media de un gen entre la población de individuos sanos y la de enfermos). En este apartado, comenzaremos por describir un método paramétrico (el *modelo ANOVA*) que permite la comparación de más de dos poblaciones.

A continuación, estudiaremos los *modelos de regresión*, que son una generalización de los modelos ANOVA y que servirán para evaluar la dependencia de una variable con respecto a otras variables explicativas así como para definir la ecuación de la función que las relaciona.

Por último, veremos los *modelos lineales generalizados* que se utilizan en los casos específicos en los que no se cumple la hipótesis de normalidad necesaria para aplicar los modelos ANOVA o los modelos de regresión.

Todos estos modelos se tratan de forma muy básica en este libro. Remitimos al lector a otros textos más específicos en esta materia como [7, 8, 15], para ampliar la información.

### 3.5.1. Modelos ANOVA

El objetivo del *análisis de la varianza* (ANOVA) es evaluar la contribución de una o más variables categóricas (factores) en la variabilidad total de la variable respuesta. Dicha variable respuesta debe ser una variable aleatoria continua con distribución normal.

Supongamos que nuestra variable respuesta es la expresión de un gen (que ha sido estimada mediante *microarrays*) y que tenemos varias observaciones de esta variable en diferentes individuos que padecen una determinada enfermedad. Supongamos que los diferentes individuos los podemos clasificar según el tratamiento que hayan recibido, por ejemplo, un grupo control y dos tratamientos diferentes A y B. Así pues, el factor tratamiento tiene 3 niveles. ANOVA descompone la variabilidad de la expresión

génica en la parte debida al factor tratamiento y en otra parte debida a otras causas (o simplemente a la propia variación aleatoria de los datos). Esta descomposición se utiliza para definir el estadístico de contraste ( $F$ -ratio), siendo la hipótesis nula a contrastar la igualdad de medias entre los grupos de tratamiento  $H_0 : \mu_{control} = \mu_A = \mu_B$ . Esta hipótesis equivale a decir que el factor tratamiento no influye en la expresión media del gen, ya que se obtiene la misma media para los tres tratamientos estudiados. Si se rechaza esta hipótesis en base al p-valor obtenido en el contraste, la conclusión será que la expresión media del gen no es la misma para los tres grupos, es decir, que depende del factor tratamiento.

La ecuación del modelo lineal correspondiente al modelo ANOVA con un factor se puede escribir como:  $y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$ , donde  $y_{ij}$  sería la expresión génica del individuo  $j$  dentro del grupo de tratamiento  $i$ ,  $\mu$  la media global de todas las observaciones,  $\alpha_i$  el efecto de pertenecer al grupo  $i$  y  $\varepsilon_{ij}$  el error residual del individuo  $j$  del grupo  $i$ .

El modelo ANOVA permite estudiar el efecto de más de un factor, así como los efectos de las interacciones entre factores. Siguiendo con el ejemplo, supongamos que queremos evaluar la influencia del sexo del individuo en la expresión génica. Por tanto, ahora tendremos dos factores “sexo” (con dos niveles) y “tratamiento” (con tres niveles). Así mismo, podemos estudiar también la interacción entre sexo y tratamiento: ¿las diferencias de expresión media entre tratamientos son las mismas para ambos sexos? Si esto es así, concluiremos que no existe interacción entre estos dos factores.

Veamos cómo hacer con R un ANOVA siguiendo este ejemplo:

```
# datos de expresión para 24 individuos
> expre = c(0.5, 2.1, 5.0, 0.3, 2.2, 4.7,
+ 0.4, 2.0, 5.2, 0.2, 2.2, 4.9,
+ 0.1, 4.7, 1.9, 0.8, 4.6, 1.9,
+ 0.3, 4.9, 1.8, 0.5, 4.7, 2.0)
# data frame con la expresión
# y el grupo al que pertenece cada individuo:
# sexo: 0=H, 1=M
# trat: 1=control, 2=A, 3=B
> misdatos = data.frame(''sexo'' = as.factor(rep(0:1, each = 12)),
+ "trat" = as.factor(rep(1:3, 8)),
+ "expr" = expre)
# ANOVA
> manova = aov(expr ~ trat*sexo, data = misdatos)
```

La expresión ‘expr ~ trat\*sexo’ indica que estamos estudiando la variable “expresión” en función del “tratamiento” y del “sexo”, incluyendo la interacción entre ambos (de ahí el símbolo multiplicativo ‘\*’). Si no queremos incluir interacciones, basta con escribir ‘expr ~ trat+sexo’. Veamos los resultados y cómo interpretarlos:

```
> summary(manova)
      Df Sum Sq Mean Sq F value Pr(>F)
trat        2   49.21   24.604 816.346 <2e-16 ***
sexo        1     0.09    0.094   3.111 0.0948 .
trat:sexo   2   32.04   16.021 531.581 <2e-16 ***
Residuals  18    0.54    0.030
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1
```

En ANOVA se realiza un contraste de hipótesis por cada factor e interacción, y el estadístico de contraste (“F value”) sigue una distribución  $F$  con grados de libertad los del factor a estudiar y los residuales. Según los p-valores obtenidos (‘Pr (>F)’), podemos concluir que para un nivel de significación del 5% el factor tratamiento y la interacción tiene un efecto estadísticamente significativo sobre la expresión media del gen.

Para ver dónde están las diferencias entre tratamientos, se puede realizar una prueba *post-hoc*, como por ejemplo la *prueba de Tukey*:

```

> mitukey = TukeyHSD(mianova)
> mitukey
  Tukey multiple comparisons of means
  95 % family-wise confidence level

Fit: aov(formula = expr ~ trat * sexo, data = misdatos)

$trat
      diff       lwr      upr   p adj
2-1 3.037500e+00 2.8159651 3.2590349 0
3-1 3.037500e+00 2.8159651 3.2590349 0
3-2 4.440892e-16 -0.2215349 0.2215349 1

$sexo
      diff       lwr      upr   p adj
1-0 -0.125 -0.2739011 0.0239011 0.0947505

$`trat:sexo`
      diff       lwr      upr   p adj
2:0-1:0 1.775 1.3848719 2.1651281 0.0000000
3:0-1:0 4.600 4.2098719 4.9901281 0.0000000
1:1-1:0 0.075 -0.3151281 0.4651281 0.9888090
2:1-1:0 4.375 3.9848719 4.7651281 0.0000000
3:1-1:0 1.550 1.1598719 1.9401281 0.0000000
3:0-2:0 2.825 2.4348719 3.2151281 0.0000000
1:1-2:0 -1.700 -2.0901281 -1.3098719 0.0000000
2:1-2:0 2.600 2.2098719 2.9901281 0.0000000
3:1-2:0 -0.225 -0.6151281 0.1651281 0.4709202
1:1-3:0 -4.525 -4.9151281 -4.1348719 0.0000000
2:1-3:0 -0.225 -0.6151281 0.1651281 0.4709202
3:1-3:0 -3.050 -3.4401281 -2.6598719 0.0000000
2:1-1:1 4.300 3.9098719 4.6901281 0.0000000
3:1-1:1 1.475 1.0848719 1.8651281 0.0000000
3:1-2:1 -2.825 -3.2151281 -2.4348719 0.0000000

```

Si nos fijamos en el factor tratamiento, vemos los p-valores resultantes de comparar los tratamientos dos a dos. Estos p-valores ajustados (ya que se están contrastando simultáneamente varias hipótesis) nos indican que hay diferencias estadísticamente significativas entre el grupo control y los tratamientos A y B, pero que no las hay entre ambos tratamientos (veremos después por qué). En cuanto al sexo, era de esperar que no hubiera diferencias estadísticamente significativas entre sexos, ya que dicho factor no había resultado ser significativo en el ANOVA. Para interpretar mejor dónde están las diferencias y qué significado tiene la interacción en este caso, vamos a centrarnos en la Figura 3.10, generada de la siguiente forma:

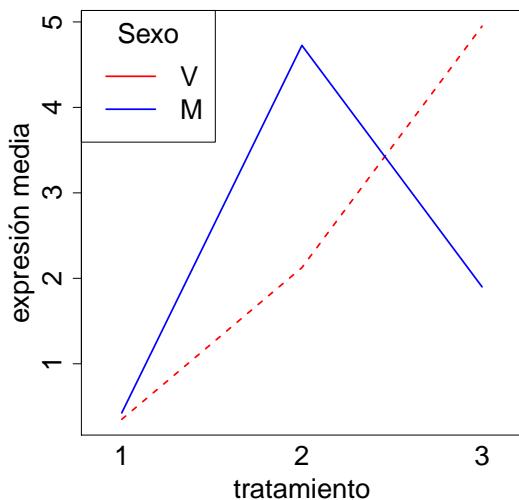
```

> interaction.plot(misdatos$strat, misdatos$sexo, misdatos$expr,
+ trace.label = "sexo", col = c(2,4), xlab = ``tratamiento'', 
+ ylab = "expresión media", lwd = 2, legend=F)
> legend('topleft', c('V', 'M'), title='Sexo', lwd=2, col = c(2,4))

```

El gráfico permite apreciar que no hay diferencias entre sexos en el grupo control, mientras que se obtiene una mayor expresión media con el tratamiento A en mujeres que en hombres y sucede exactamente lo contrario con el tratamiento B. Aunque este es un gráfico meramente descriptivo, la prueba de Tukey nos permite contrastar la significación estadística de estas afirmaciones.

Aunque no entraremos en más detalle sobre el ANOVA en este texto, debemos recalcar la importancia de validar el modelo a través de los residuos obtenidos, para comprobar a partir de ellos los requisitos



**Figura 3.10:** Gráfico de interacción entre los factores sexo y tratamiento. La línea roja corresponde a hombres y la azul a mujeres. El tratamiento 1 es el grupo control, mientras que los códigos 2 y 3 corresponden a los tratamientos A y B, respectivamente.

de normalidad, independencia y homocedasticidad (igualdad de varianzas) necesarios para aplicar el modelo ANOVA. El lector puede encontrar más información en la bibliografía recomendada.

La alternativa no paramétrica al modelo ANOVA con un factor es el *test de Kruskal-Wallis* (en R, la función ‘kruskal.test’), que no requiere que la variable siga una distribución normal.

### 3.5.2. Regresión lineal

Cuando las variables explicativas en nuestro modelo no son únicamente categóricas sino continuas o bien nos interesa conocer exactamente la ecuación que relaciona la variable respuesta con las variables explicativas, utilizamos los *modelos de regresión lineal*.

Al igual que en ANOVA, en los modelos de regresión lineal se deben cumplir las hipótesis de normalidad, independencia y homocedasticidad. Así pues, una parte importante del análisis es validar el modelo obtenido, es decir, verificar dichas hipótesis a partir de los residuos generados.

#### Regresión lineal simple

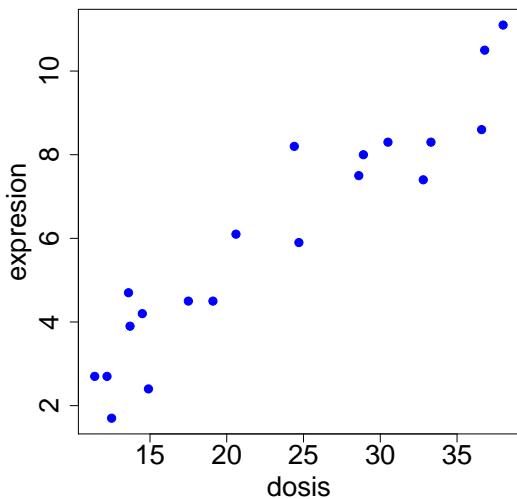
En *regresión lineal simple* sólo introducimos en el modelo una variable explicativa, por lo que la ecuación que relaciona la variable respuesta con dicha variable explicativa es una recta (la recta de regresión). Siguiendo con el ejemplo de la expresión génica, imaginemos que ahora nuestro objetivo es predecir el valor de la expresión en función de la dosis suministrada de cierto medicamento. Veamos gráficamente si tiene sentido ajustar una recta de regresión a estos datos, mediante el gráfico de dispersión de la Figura 3.11 que se ha generado con la siguiente instrucción:

```
> dosis = c(38.0, 20.6, 17.5, 30.5, 36.6, 12.5,
+ 12.2, 11.4, 24.7, 24.4, 13.7, 32.8,
+ 14.9, 33.3, 36.8, 14.5, 13.6, 28.6,
```

```

+ 28.9, 19.1)
> expre = c(11.1, 6.1, 4.5, 8.3, 8.6, 1.7,
+ 2.7, 2.7, 5.9, 8.2, 3.9, 7.4,
+ 2.4, 8.3, 10.5, 4.2, 4.7, 7.5,
+ 8.0, 4.5)
> plot(dosis, expre, ylab = "expresion", pch = 20, col = 4, cex = 2)

```



**Figura 3.11:** Gráfico de dispersión para las variables “dosis” y “expresión génica”.

En la Figura 3.11 se aprecia una tendencia lineal en la relación entre la dosis y la expresión del gen. Para cuantificar numéricamente el grado de relación lineal entre dos variables, se utiliza el *coeficiente de correlación lineal de Pearson*, que no depende de las magnitudes ni unidades de medida y siempre toma valores entre -1 y 1. Valores cercanos a 1 indican una fuerte relación lineal creciente, mientras que valores cercanos a -1, indican que existe también una fuerte relación lineal pero decreciente. Un valor cercano a 0 significa que no hay relación lineal entre ambas variables, bien porque no hay ningún tipo de relación entre ellas, bien porque hay otro tipo de relación distinta de la lineal. En R:

```

> cor(dosis, expre, method = "pearson")
[1] 0.9448318

```

Como ya anticipaba el gráfico de dispersión (Figura 3.11), el coeficiente de correlación es bastante elevado (cercano a 1). Cuando los datos no son continuos pero sí están medidos en escala ordinal, se suelen utilizar otras medidas de correlación como el coeficiente de Kendall o el de Spearman.

Dado el fuerte grado de relación lineal observada entre las variables, tiene sentido ajustar un modelo de regresión lineal simple:  $Y = \beta_0 + \beta_1 X + \varepsilon$ . En nuestro caso,  $Y$  = expresión génica y  $X$  = dosis del medicamento. El término  $\varepsilon$  es el error y representa la variación aleatoria (no debida a las variables explicativas incluidas en el modelo).

La *recta de regresión* se calcula de forma que se minimicen las desviaciones entre los valores observados y los valores predichos mediante la recta. En concreto, los coeficientes de la recta serán aquellos que minimicen la expresión:  $\sum(y_i - (\beta_0 + \beta_1 x_i))^2$  (de ahí que también se le denomine recta de mínimos cuadrados). Se puede estimar la recta de regresión en R utilizando las siguientes instrucciones:

```

> rectaregre = lm(expre ~ dosis)
> summary.lm(rectaregre)

```

```

Call:
lm(formula = expre ~ dosis)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.37221 -0.56534 -0.00988  0.59352  1.81421 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.40845   0.56762  -0.72   0.481    
dosis        0.27845   0.02275  12.24 3.68e-10 ***  
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'. ' 0.1 ' ' 1

Residual standard error: 0.9254 on 18 degrees of freedom
Multiple R-squared: 0.8927, Adjusted R-squared: 0.8867 
F-statistic: 149.8 on 1 and 18 DF,  p-value: 3.677e-10

```

En primer lugar, vemos que la variable “dosis” tiene un p-valor asociado de  $3,68e-10$ . Este p-valor se obtiene al contrastar la hipótesis nula  $H_0 : \beta_1 = 0$ . Se realiza un contraste de hipótesis por cada una de las variables explicativas introducidas en el modelo. Rechazar  $H_0$  implica que el coeficiente es significativamente distinto de 0, por lo que tiene sentido incluir esta variable en el modelo. Vemos también que el coeficiente de determinación  $R^2$  (‘Multiple R-squared’) es 0,8927, es decir, nuestro modelo explica un 89,27% de la variabilidad de la expresión génica. Si nos fijamos en la columna ‘Estimate’, podemos ver que el modelo estimado sería  $E(Y) = -0,40845 + 0,27845X$ . A partir de este modelo, podemos predecir el valor medio de la expresión del gen para cada dosis de medicamento. Las predicciones o estimaciones serán válidas dentro del rango de las variables explicativas para el que hayamos estimado la ecuación y aumentará el error conforme nos alejemos de dicho rango.

## Regresión lineal múltiple

El modelo de *regresión lineal múltiple* permite incluir más de una variable en el modelo, incluso variables de tipo categórico. Estas últimas se introducen de forma especial. Para incluir una variable categórica con  $k$  posibles valores, se definen  $k - 1$  variables 0-1 (llamadas *ficticias* o *dummy*). Tomando como referencia el primer valor, las  $k - 1$  variables ficticias se definen como  $Z_i = 1$ , si el dato pertenece al grupo  $i$  y 0 en caso contrario, con  $i = 2, \dots, k$ . Si todas las variables  $Z_i$  valen 0, es porque el dato observado está en el grupo 1 (tomado como referencia).

Siguiendo con el ejemplo anterior, supongamos que también queremos introducir la variable continua “edad” en el modelo, obteniendo así un modelo de regresión lineal múltiple. Veamos cómo hacerlo en R:

```

> edad = c(49, 35, 22, 46, 37, 23, 21, 26, 32, 44,
+ 24, 44, 26, 39, 47, 38, 39, 46, 40, 31)
> regrel = lm(expre ~ dosis+edad)
> summary.lm(regrel)

```

```

Call:
lm(formula = expre ~ dosis + edad)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.43248 -0.31087 -0.07677  0.57753  1.04916 

```

```

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.14222   0.68312  -3.136  0.00602 **
dosis        0.20020   0.02938   6.815 3.01e-06 ***
edad         0.10019   0.02962   3.382  0.00354 **
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 0.7362 on 17 degrees of freedom
Multiple R-squared: 0.9359, Adjusted R-squared: 0.9283
F-statistic: 124 on 2 and 17 DF,  p-value: 7.247e-11

```

El p-valor del modelo (asociado al test  $F$ ) es  $7.247e-11$ , que indica que alguna de las variables explicativas introducidas tienen un coeficiente significativamente distinto de 0. Si nos fijamos en los p-valores asociados a cada variable, vemos que ambas contribuyen a explicar la variabilidad de la expresión génica. Para comparar modelos con distinto número de variables, estudiaremos el coeficiente de determinación  $R^2$  ajustado ('Adjusted R-squared') que, en este caso, es superior al del modelo de regresión lineal simple.

Veamos cómo introducir la variable categórica "sexo" en el modelo.

```

> sexo = as.factor(rep(c("H", "M"), 10))
> regre2 = lm(expre ~ dosis+edad+sexo)
> summary.lm(regre2)

Call:
lm(formula = expre ~ dosis + edad + sexo)

Residuals:
    Min      1Q  Median      3Q     Max
-1.29644 -0.34019  0.04329  0.30752  1.19309

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.15782   0.64422  -3.350  0.00407 **
dosis        0.18272   0.02942   6.212 1.25e-05 ***
edad         0.12045   0.03020   3.989  0.00106 **
sexoM       -0.59349   0.33607  -1.766  0.09647 .
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 0.6942 on 16 degrees of freedom
Multiple R-squared: 0.9463, Adjusted R-squared: 0.9363
F-statistic: 94.04 on 3 and 16 DF,  p-value: 2.243e-10

```

Como podemos observar, si introducimos la variable categórica como un factor, R ya se encarga de definir las variables ficticias. En este caso, ha tomado como referencia a los hombres ("H") y ha definido la variable 'sexoM' que tomará el valor 1 para las mujeres. Aunque el p-valor asociado a esta variable se puede considerar "dudoso", ya que es 0,09647, optaremos por incluirla en el modelo ya que el  $R^2$  ajustado ha aumentado respecto al modelo anterior.

Finalmente, podemos ajustar un modelo incluyendo las interacciones entre todas las variables.

```

> regre3 = lm(expre ~ dosis*sexo*edad)
> summary.lm(regre3)

Call:
lm(formula = expre ~ dosis * sexo * edad)

```

```

Residuals:
    Min      1Q   Median      3Q     Max 
-1.29067 -0.27464  0.03207  0.34393  1.01045 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.307223  2.482711  0.124   0.9036    
dosis        0.076265  0.125565  0.607   0.5549    
sexoM       -8.405418  4.485650 -1.874   0.0855 .  
edad         0.038719  0.077077  0.502   0.6245    
dosis:sexoM 0.385618  0.250915  1.537   0.1503    
dosis:edad   0.003202  0.003115  1.028   0.3243    
sexoM:edad   0.243901  0.132858  1.836   0.0913 .  
dosis:sexoM:edad -0.011042  0.006237 -1.770   0.1020  
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 0.7009 on 12 degrees of freedom
Multiple R-squared: 0.959, Adjusted R-squared: 0.935 
F-statistic: 40.06 on 7 and 12 DF, p-value: 2.176e-07

```

Se obtienen resultados un tanto extraños, ya que aunque el p-valor del modelo es significativo, no lo son los asociados a las variables explicativas. Además el  $R^2$  ajustado ha disminuido con respecto al modelo anterior.

Así pues, consideramos que el mejor modelo de todos los estudiados es el que incluye las tres variables sin interacciones:  $E(Y) = -2,15782 + 0,18272dosis + 0,12045edad - 0,59349(sexo = M)$

Otras variables explicativas que se podrían introducir en el modelo, son términos cuadráticos, cúbicos o de cualquier orden, e incluso otras funciones de las variables explicativas como su raíz cuadrada, su logaritmo, etc. Por ejemplo,  $Y = \beta_0 + \beta_1X + \beta_2X^2 + \varepsilon$  sería otro posible modelo de regresión lineal múltiple.

Sea cual sea el modelo, el error debe cumplir que  $\varepsilon \sim N(0, \sigma^2)$ . Comprobar este requisito forma parte de la fase de validación del modelo.

### 3.5.3. Modelos lineales generalizados

Cuando, en un modelo de regresión lineal simple, el error no se distribuye normalmente ni se encuentra una transformación adecuada de la variable respuesta para que se satisfaga este requisito, se puede optar por los *modelos lineales generalizados* (GLM), que permiten que el error siga otro tipo de distribución distinta de la normal.

Siguiendo con el ejemplo anterior sobre la expresión génica, supongamos que hemos cuantificado la expresión utilizando la técnica de RNA-seq. El valor de expresión de un gen sería, pues, el número de lecturas que mapean a ese gen y, por tanto, un valor discreto. Una distribución de probabilidad que se podría ajustar a la distribución de los valores de expresión del gen podría ser la distribución de Poisson. Para obtener un modelo que relacione la expresión génica con la dosis de un medicamento u otras variables explicativas, necesitaríamos entonces aplicar un modelo lineal generalizado (en este caso, la regresión de Poisson).

La distribución de Poisson tiene la característica de que su media y su varianza son iguales. Dado que en datos de expresión medidos por RNA-seq con réplicas biológicas se ha observado que no necesaria-

mente se cumple que la media y la varianza sean iguales, se suelen utilizar otras distribuciones distintas a la de Poisson que tengan en cuenta esta “sobre-dispersión”, como por ejemplo la distribución binomial negativa. Este tipo de modelos son utilizados en distintos paquetes de R para calcular expresión diferencial como ‘edgeR’ [18] o ‘DESeq’ [1].

En R, se pueden obtener modelos lineales generalizados mediante la función ‘`glm`’, indicando con el parámetro ‘`family`’ la distribución de probabilidad que se quiere utilizar.

### 3.5.4. Ejercicios

- Resuelve el contraste de hipótesis planteado en el ejemplo de la Sección 3.4.1, utilizando un modelo ANOVA. ¿Se obtienen las mismas conclusiones que con la función ‘`t.test`’?

*Solución:*

```
# Preparamos los datos
> expre = c(sanos, enfermos)
> grupo = as.factor(rep(c("sanos", "enfermos"), each = 10))
# ANOVA
> manova = aov(expre ~ grupo)
> summary(manova)
   Df Sum Sq Mean Sq F value    Pr(>F)
grupo      1  65.52   65.52  130.8 1.09e-09 ***
Residuals  18   9.02    0.50
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1
```

Al igual que con ‘`t.test`’, llegamos a la conclusión de que para un nivel de significación del 5 % se rechaza la igualdad de medias entre ambos grupos.

- A partir del ejemplo de la Subsección 3.5.1, evalúa la significación estadística del efecto del tratamiento (sin tener en cuenta el sexo) utilizando el test de Kruskal-Wallis (función ‘`kruskal.test`’ en R).

*Solución:*

```
> kruskal.test(expr ~ trat, data = misdatos)

Kruskal-Wallis rank sum test

data: expr by trat
Kruskal-Wallis chi-squared = 15.4472, df = 2, p-value = 0.0004423
```

El p-valor nos indica que las medias de expresión para los tres tratamientos son significativamente diferentes para  $\alpha = 0.05$ .

- A partir del ejemplo de la Sección 3.5.2, estima un modelo cuadrático que relacione la dosis del medicamento con la expresión génica. ¿Es mejor este modelo que el modelo de regresión lineal simple?

*Solución:*

```
> dosis2 = dosis^2
> cuadratico = lm(expre ~ dosis + dosis2)
> summary.lm(cuadratico)

Call:
lm(formula = expre ~ dosis + dosis2)
```

```

Residuals:
    Min      1Q  Median      3Q     Max 
-1.34261 -0.60466  0.02275  0.60341  1.66162 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.324860   1.882141  -0.704   0.4910    
dosis        0.366790   0.174167   2.106   0.0504 .  
dosis2       -0.001825  0.003566  -0.512   0.6154    
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1

Residual standard error: 0.945 on 17 degrees of freedom
Multiple R-squared:  0.8943, Adjusted R-squared:  0.8819 
F-statistic: 71.94 on 2 and 17 DF,  p-value: 5.051e-09

> anova.lm(cuadratico)
Analysis of Variance Table

Response: expre
          Df  Sum Sq Mean Sq F value    Pr(>F)    
dosis      1 128.253 128.253 143.6239 1.027e-09 ***
dosis2     1  0.234   0.234   0.2619   0.6154    
Residuals 17 15.181   0.893                        
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05
'.' 0.1 ' ' 1

```

El coeficiente del término cuadrático no se puede considerar significativamente distinto de 0. Además el  $R^2$  ajustado es ligeramente inferior al obtenido en el modelo de regresión lineal simple. Por tanto, no es mejor el modelo cuadrático.

### 3.6. Métodos multivariantes

En un análisis bioinformático es común procesar miles de variables, por ejemplo, la expresión de todos los genes del genoma, o la concentración de miles de metabolitos. Este tipo de problemas se engloban dentro de la estadística multivariante. En general, los *métodos multivariantes* son aquellos que analizan simultáneamente medidas múltiples de cada uno de los individuos u objetos que forman parte de la investigación.

En los apartados anteriores, hemos visto cómo abordar estos problemas multivariantes de forma univariante: haciendo un contraste de hipótesis por cada variable o ajustando un modelo para cada una de ellas. Posteriormente, se aplica un método de corrección por tests múltiples para ajustar los p-valores obtenidos. Pero este enfoque univariante no suele tener en cuenta la posible relación de unas variables con otras, sino que las trata de forma independiente. Por eso son interesantes en este contexto los métodos multivariantes, que resumen la información de todas las variables y muestras estudiadas de forma eficiente y permiten destacar aquellas variables más informativas o clasificar las variables o individuos en grupos.

Los métodos multivariantes pueden catalogarse en métodos de “*aprendizaje supervisado*” o de “*aprendizaje no supervisado*”. Los métodos que abordan el problema de clasificar a los individuos en grupos o clases desconocidos de antemano entran dentro de los métodos de aprendizaje no supervisado. Si se

sabe a qué grupo pertenece cada individuo pero se pretende ver qué variables han determinado esta clasificación para poder hacer predicciones en nuevos individuos, se aplican los métodos de aprendizaje supervisado.

No podemos abordar todos los métodos multivariantes existentes en este texto, pero la siguiente tabla enumera algunos de los más comunes, así como algunas de las funciones en R para llevar a cabo los análisis correspondientes y la librería que las contiene. Algunas de las funciones están incluidas en los paquetes básicos de R (como ‘stats’), aunque pueden existir otros paquetes con implementaciones diferentes de las mismas o incluso paquetes en Bioconductor<sup>5</sup> específicos para el análisis de datos bioinformáticos:

Método	Clasificación	Función y librería R
Análisis de componentes principales (PCA)	No supervisado	prcomp {stats}
Escalado multidimensional	No supervisado	cmdscale {stats}
		isomds {MASS}
Análisis de correspondencias	No supervisado	ca {ca}
<i>Clustering</i>	No supervisado	kmeans {stats}
		pamk {fpc}
		hclust {stats}
		pvclust {pvclust}
Análisis factorial	No supervisado	factanal {stats}
Análisis de componentes independientes (ICA)	No supervisado	ica {e1071}
Random forest	No supervisado	randomForest {randomForest}
Análisis discriminante	Supervisado	lda, qda {MASS}
Mixturas	Supervisado	mix {mixdist}
		Mclust {mclust}
Vecinos más cercanos	Supervisado	knn {class}
Redes neuronales	Supervisado	nnet {nnet}
		neuralnet {neuralnet}
Regresión en mínimos cuadrados parciales (PLSR)	Supervisado	plsr {pls}
<i>Support Vector Machines</i> (SVM)	Supervisado	svm {e1071}

**Tabla 3.3:** Clasificación de los métodos multivariantes más comunes.

A continuación, se exponen con mayor detalle algunos de los métodos multivariantes más utilizados, como son el análisis de conglomerados (*clusters*) y el análisis de componentes principales (PCA).

### 3.6.1. Clustering

El *análisis de conglomerados* o *clustering* es una técnica de clasificación no supervisada cuya idea es agrupar los elementos en  $k$  grupos basándose en la similitud existente entre ellos. Se pueden agrupar tanto las variables como las observaciones. Algunos de los algoritmos de *clustering* más extendidos son el método de las *K-medias* (que fija previamente el número de grupos  $k$  en los que hay que clasificar a los elementos) o los *métodos jerárquicos* (que no fijan de antemano el número de grupos).

Independientemente del algoritmo de *clustering* utilizado, es importante determinar cómo se va a medir la similitud entre los individuos o, equivalente, definir la distancia entre ellos: distancia euclídea, correlación, etc. Esta medida de similitud dependerá del tipo de variables que estemos estudiando (cuantitativas, ordinales, categóricas, ...).

<sup>5</sup>Bioconductor. <http://www.bioconductor.org>

En R, la función ‘`kmeans`’ corresponde al algoritmo de las  $K$ -medias y ‘`hclust`’ al *clustering jerárquico*.

### 3.6.2. Análisis de Componentes Principales

El objetivo del *Análisis de Componentes Principales* (PCA) es resumir la información contenida en  $p$  variables utilizando un número menor de variables y generando la menor pérdida de información posible. De ahí que el PCA esté clasificado como un método de reducción de la dimensión.

Las nuevas variables calculadas se denominan *componentes principales* y son combinaciones lineales de las variables originales. Por tanto, lo que se hace en realidad es un cambio de base, una rotación de los ejes de forma que con un número reducido de componentes principales se explique un alto porcentaje de la variabilidad de los datos.

Sea  $X$  nuestra matriz de datos, con  $p$  columnas (variables) y  $n$  observaciones (filas). En principio, el número de componentes principales es igual al mínimo entre el número de variables originales (columnas de  $X$ ) y el número de observaciones (filas de  $X$ ). Las componentes principales están ordenadas de mayor a menor porcentaje de variabilidad explicada, siendo la primera componente la que mayor variabilidad recoge. Así, las últimas componentes principales se pueden descartar porque aportan muy poca información y, de esta forma, se reduce la dimensión original.

Las componentes principales se obtienen a partir de la descomposición de la matriz de varianzas y covarianzas de  $X$  en vectores y valores propios y están incorreladas entre sí. Si tenemos  $p$  variables  $y_1, y_2, \dots, y_p$  y  $n$  observaciones, cada componente principal  $k$  se define como una combinación lineal de las  $p$  variables originales. Para la observación  $i$ :  $z_{ik} = c_1y_{i1} + c_2y_{i2} + \dots + c_py_{ip}$ .  $z_{ik}$  es el *score* de la observación  $i$  para la *componente principal*  $k$ . Cada coeficiente  $c_j$  es el *loading* de la variable  $j$  en la componente principal  $k$ , y vienen dados por los *vectores propios*. Los *valores propios*  $\lambda_i$  son la varianza explicada por cada componente principal. Por tanto, para obtener la proporción de variabilidad explicada por la componente  $k$ , calcularemos  $\lambda_k / \sum \lambda_i$ .

Veamos un ejemplo muy sencillo de cómo realizar un PCA en R utilizando los datos de Marioni (ver Subsección 3.1.3). En primer lugar, leemos los datos y los preparamos para utilizar la función `prcomp`. Para ello, pasamos los nombres de los genes (que están en la primera columna) a los nombres de las filas y eliminamos esta primera columna. Después, quitaremos aquellos genes con 0 *counts* en todas las muestras, ya que no van a aportar ninguna información. Por último, transponemos la matriz, ya que nuestras variables son los genes (que deben estar en las columnas) y nuestras observaciones son las muestras (que pondremos en las filas).

```
# Mostrar qué tipo de datos hay en cada columna
> colnames(marioni)
[1] "EnsemblGeneID" "Chr"          "GeneStart"      "GeneEnd"       "Status"        "
[5] "ExternalID"      "R1L1Kidney"    "R1L2Liver"     "R1L3Kidney"    "R1L4Liver"     "R1L6Liver"     "
[9] "R1L7Kidney"      "R1L8Liver"     "R2L1Liver"     "R2L2Kidney"    "R2L3Liver"     "R2L4Kidney"    "
[13] "R2L6Kidney"      "R2L7Liver"     "R2L8Kidney"
# Leer los identificadores de los genes (la primera columna de datos)
# y asignar a cada fila de datos el nombre del gen que le corresponde
> rownames(marioni) = marioni[,1]
# Eliminar las filas que contienen datos diferentes de counts
> marioni = marioni[,-1:-6]
# Eliminar las filas con 0 counts
> marioni = marioni[which(rowSums(marioni) > 0),]
```

```
# Trasposición de la matriz de datos
> marioni = t(marioni)
```

En el análisis de componentes principales es importante el escalado de los datos. En general, es recomendable *centrar los datos* (es decir, restar a cada variable su media) y, cuando las variables están medidas en escalas muy diferentes, puede ser recomendable también escalar (dividir cada variable por su desviación típica). Ambas opciones están incluidas en la función ‘`prcomp`’. En este caso, hemos optado por centrar pero no escalar, ya que las variables son la expresión de los genes, y se suponen medidas en la misma escala.

```
> mypca = prcomp(marioni, center = TRUE, scale. = FALSE)
> summary(mypca)
Importance of components:

PC1      PC2      PC3      PC4      PC5      PC6
PC7      PC8
Standard deviation   5.262e+04 5.064e+03 2.924e+03 1.865e+03 495.24780 446.72315
411.23741 375.37072
Proportion of Variance 9.861e-01 9.130e-03 3.040e-03 1.240e-03 0.00009 0.00007
0.00006 0.00005
Cumulative Proportion 9.861e-01 9.952e-01 9.983e-01 9.995e-01 0.99961 0.99968
0.99974 0.99979

PC9      PC10     PC11     PC12     PC13     PC14
Standard deviation 355.12792 351.68619 341.29907 328.41629 318.59240 2.201e-10
Proportion of Variance 0.00004 0.00004 0.00004 0.00004 0.00004 0.000e+00
Cumulative Proportion 0.99984 0.99988 0.99993 0.99996 1.00000 1.000e+00
```

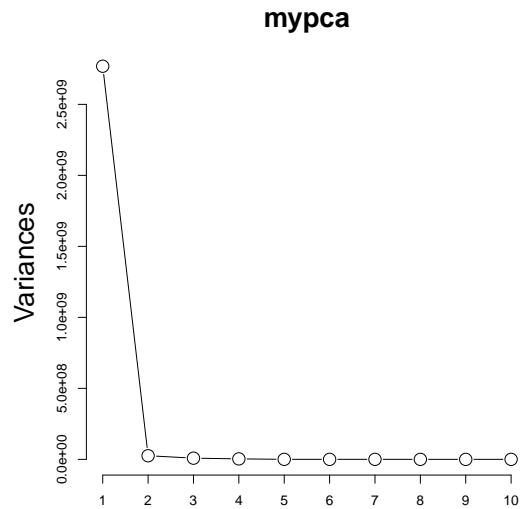
Como podemos observar, la primera componente explica más del 98 % de la variabilidad de los datos. Es lógico obtener este resultado en este ejemplo, ya que la principal fuente de variabilidad en nuestros datos es el tejido del que proceden las muestras y sólo hay dos tejidos. El siguiente paso sería decidir el número de componentes principales que queremos retener. Aunque en este sencillo ejemplo es obvio que con una componente es suficiente, existen diversos criterios para llevar a cabo esta selección. En este texto, presentaremos el criterio del “*Scree plot*”. En este gráfico (ver Figura 3.12), se representa el número de componente frente a su varianza. Se trata de buscar un “codo” en el gráfico, es decir, un número de componente a partir del cual las varianzas sean iguales o similares.

```
> screeplot(mypca, type = "lines")
```

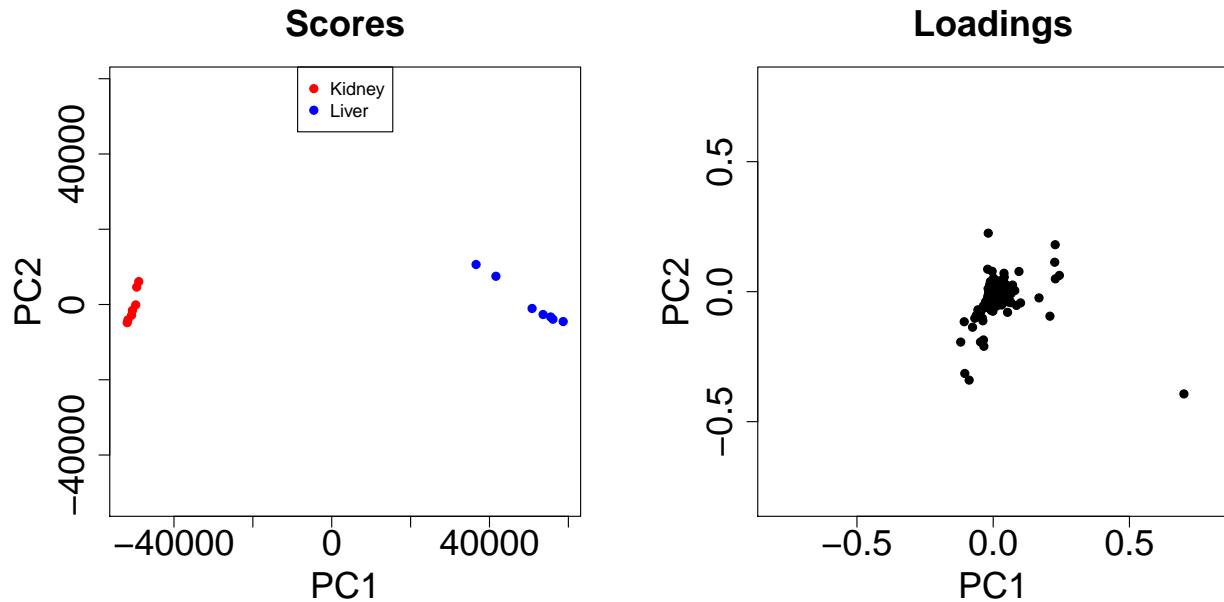
Veamos cómo representar el *gráfico de PCA* correspondiente a las dos primeras componentes principales (Figura 3.13). Es importante que los dos ejes tengan el mismo rango, de lo contrario la interpretación del gráfico podría ser confusa. Se han representado tanto los *scores* de las muestras como los *loadings* de los genes:

```
> par(mfrow = c(1,2))
> # Scores plot
> mylim = range(mypca$x)
> plot(mypca$x[, "PC1"], mypca$x[, "PC2"], xlim = mylim, ylim = mylim,
+       xlab = paste("PC1"), ylab = "PC2", col = "white", main = "Scores")
> points(mypca$x[grep("Kidney", rownames(mypca$x)), "PC1"],
+          mypca$x[grep("Kidney", rownames(mypca$x)), "PC2"], col = 2)
> points(mypca$x[grep("Liver", rownames(mypca$x)), "PC1"],
+          mypca$x[grep("Liver", rownames(mypca$x)), "PC2"], col = 4)
> legend("top", c("Kidney", "Liver"), pch = 19, col = c(2,4))
> # Loadings plot
> mylim2 = range(mypca$rotation)
> plot(mypca$rotation[, "PC1"], mypca$rotation[, "PC2"], xlim = mylim2, ylim = mylim2,
+       xlab = paste("PC1"), ylab = "PC2", main = "Loadings")
```

En el gráfico para *scores*, se observa que la primera componente separa claramente las muestras de riñón y de hígado. La segunda componente explica parte de la variabilidad entre réplicas, aunque en este



**Figura 3.12:** Scree plot.



**Figura 3.13:** PCA sobre los datos de Marioni

caso la agrupación de las réplicas de cada condición es muy buena. El gráfico de *loadings* muestra que hay algunos genes (especialmente el que corresponde al punto de la derecha) con un comportamiento un tanto extremo y que podrían considerarse “*outliers*”. Es recomendable que esto no pase. En estos casos se puede optar por eliminar dichas variables y repetir el análisis, o bien considerar el escalado de los datos o aplicar algún tipo de transformación para “suavizar” el comportamiento extremo de estas variables (como por ejemplo, realizar el PCA sobre la transformación logarítmica de los datos).

### 3.7. Bibliografía

- [1] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010.
- [2] G. Box, S. Hunter, and W. Hunter. *Estadística para Investigadores. Diseño, innovación y descubrimiento (2a edición)*. Reverté, 2008.
- [3] M. Crawley. *The R book*. Wiley, 2007.
- [4] M. DeGroot. *Probabilidad y Estadística*. Adisson-Wesley Iberoamericana, 1988.
- [5] S. Dudoit and M. Van Der Laan. *Multiple testing procedures with applications to genomics*. Springer, 2008.
- [6] O. Dunn and V. Clark. *Basic statistics: a primer for the biomedical sciences*. Wiley, 2009.
- [7] J. Faraway. *Linear models with R*. Chapman & Hall/CRC, 2004.
- [8] J. Faraway. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. Chapman & Hall/CRC, 2005.
- [9] J. Hair, R. Anderson, R. Tatham, and W. Black. *Multivariate analysis*. Englewood: Prentice Hall International, 2005.
- [10] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.
- [11] S. Krawetz and D. Womble. *Introduction to bioinformatics: a theoretical and practical approach*, volume 1. Humana Pr Inc, 2003.
- [12] J. Marioni, C. Mason, S. Mane, M. Stephens, and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome research*, 18(9):1509–1517, 2008.
- [13] A. Oshlack, M. Robinson, M. Young, et al. From RNA-seq reads to differential expression results. *Genome Biology*, 11(12):220, 2010.
- [14] D. Peña. *Análisis de datos multivariantes*. McGraw-Hill/Interamericana Madrid, 2002.
- [15] D. Peña. *Regresión y diseño de experimentos*. Alianza Editorial, 2002.
- [16] J. Pevsner. *Bioinformatics and Functional Genomics*. Wiley-Blackwell, 2009.
- [17] G. Quinn and M. Keough. *Experimental design and data analysis for biologists*. Cambridge Univ Pr, 2002.
- [18] M. Robinson, D. McCarthy, and G. Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.
- [19] R. Romero and L. Zúnica Ramajo. *Métodos estadísticos en ingeniería*. Universidad Politécnica de Valencia, 2008.
- [20] P. Westfall and S. Young. *Resampling-based multiple testing: Examples and methods for p-value adjustment*. Wiley-Interscience, 1993.



# Capítulo 4

# Fundamentos de programación

*Eduardo Andrés León y José María Fernández*

## 4.1. Introducción

La Bioinformática puede ser un término muy amplio y difícil de resumir. Podríamos decir que es el uso del *hardware* y *software* de un ordenador para analizar e interpretar datos biológicos. Ya seamos matemáticos, físicos, informáticos, biólogos o químicos, serán necesarios unos conocimientos mínimos tanto de programación, como de biología. Pero, antes de nada respondamos dos preguntas básicas en bioinformática. *¿Qué es un programa?* Un programa es un conjunto de instrucciones en un particular lenguaje de programación que puede ser interpretado por un ordenador. *¿Qué es un lenguaje de programación?* Se podría definir como el conjunto de instrucciones necesarias para escribir un programa. A través de esta sintaxis podremos comunicarnos con nuestro ordenador para que realice determinadas tareas por nosotros. Muchos de los lenguajes de programación son parecidos al lenguaje humano, pero están definidos de forma mucho más estricta. En este capítulo hablaremos más extensamente del lenguaje de programación Perl. Pero primero veamos los tipos de lenguaje de programación existentes.

## 4.2. Tipos de lenguajes de programación

### 4.2.1. Abstracción

Existen dos tipos de lenguajes claramente diferenciados: los lenguajes de bajo y alto nivel. El ordenador sólo entiende un lenguaje conocido como *código binario* o *código máquina*, consistente en ceros y unos. Es decir, sólo utiliza 0 y 1 para codificar cualquier acción. Los lenguajes más próximos a la arquitectura binaria de *hardware* se denominan *lenguajes de bajo nivel* y los que son más sencillos y se encuentran más cercanos al lenguaje humano se denominan *lenguajes de alto nivel*.

#### Lenguajes de bajo nivel

Son lenguajes totalmente dependientes de la máquina, es decir, que los programas que se realizan con este tipo de lenguajes no se pueden utilizar en otras máquinas al estar prácticamente diseñados a medida del *hardware*, aprovechan al máximo las características del mismo.

Dentro de este grupo se encuentran el *lenguaje máquina*, este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de ceros y unos (código binario) para formar las órdenes interpretables por el *hardware* de la computadora. Este lenguaje es mucho más rápido que los lenguajes de alto nivel. La desventaja es que son bastantes difíciles de manejar y usar, además de tener códigos fuente enormes donde encontrar un fallo es casi imposible. El *lenguaje ensamblador* es un derivado del lenguaje máquina y está formado por abreviaturas de letras y números llamadas mnemotécnicos. Con la aparición de este lenguaje se crearon los *programas traductores* para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Como ventaja con respecto al código máquina es que los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria. Las desventajas de este lenguaje siguen siendo prácticamente las mismas, añadiendo la dificultad de tener que aprender un nuevo lenguaje difícil de probar y mantener.

### **Lenguajes de alto nivel**

Son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Están dirigidos a solucionar problemas mediante el uso de *estructuras dinámicas de datos*, algo muy utilizado en todos los lenguajes de programación. Son estructuras que pueden cambiar de tamaño durante la ejecución del programa. Nos permiten crear estructuras de datos que se adapten a las necesidades reales de un programa.

Se trata de *lenguajes independientes de la arquitectura* del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, puede migrar de una máquina a otra sin ningún tipo de problema.

Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno de la máquina para la que están diseñados. Tan sólo necesitan un traductor que entienda tanto el código fuente como las características de la máquina.

#### **4.2.2. Ejecución**

Se denominan *lenguajes compilados* cuando un *compilador* (programa traductor) traduce el *código fuente* del programa en *código máquina* (código objeto). Otro programa, el *enlazador*, unirá los ficheros de código objeto del programa principal con los de las librerías para producir el programa ejecutable. Ejemplos: C, Java.

En contraposición, en los *lenguajes interpretados* un programa *intérprete* ejecuta las instrucciones del programa de manera directa. Ejemplo: Lisp, Perl.

También hay *lenguajes mixtos*, como Java, que primero pasan por una fase de compilación en la que el código fuente se transforma en “*bytecode*”, y éste puede ser ejecutado luego (interpretado) en ordenadores con distintas arquitecturas (procesadores) que tengan todos instalados la misma “*máquina virtual*” Java.

#### **4.2.3. Paradigma de programación.**

El *paradigma de programación* es el estilo de programación empleado. Algunos lenguajes soportan varios paradigmas, y otros sólo uno. Se puede decir que históricamente han ido apareciendo para facilitar la tarea de programar según el tipo de problema a abordar, o para facilitar el mantenimiento del *software*,

o por otra cuestión similar, por lo que todos corresponden a lenguajes de alto nivel (o nivel medio), estando los lenguajes ensambladores “atados” a la arquitectura de su procesador correspondiente.

Los lenguajes clásicos de *programación procedural* dividen el problema en partes más pequeñas, que serán realizadas por subprogramas (*subrutinas, funciones o procedimientos*), que se llaman unas a otras para ser ejecutadas. Ejemplos: C, Pascal.

Otro tipo es la *programación orientada a objetos* (POO), donde se crea un sistema de *clases y objetos* siguiendo el ejemplo del mundo real, en el que unos objetos realizan acciones y se comunican con otros objetos. Ejemplos: C++, Java. La programación orientada a objetos es una forma de programar que trata de encontrar una solución a varios problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- *Clase*: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La *instanciación* es la lectura de estas definiciones y la creación de un objeto a partir de ellas.
- *Herencia* (por ejemplo, herencia de la clase C a la clase D): es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables públicas declaradas en C. Los componentes registrados como “privados” (*private*) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de OOP.
- *Objeto*: entidad provista de un conjunto de propiedades o *atributos* (datos) y de comportamiento o funcionalidad (*métodos*) los mismos que consecuentemente reaccionan a eventos. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una *instancia* a una clase.
- *Método*: algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un “mensaje”. Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un “evento” con un nuevo mensaje para otro objeto del sistema.
- *Modularidad*: propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas *módulos*), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.

Otros lenguajes de *programación funcional* realizan tareas evaluando funciones, como en Matemáticas, de manera recursiva. Ejemplo: Lisp.

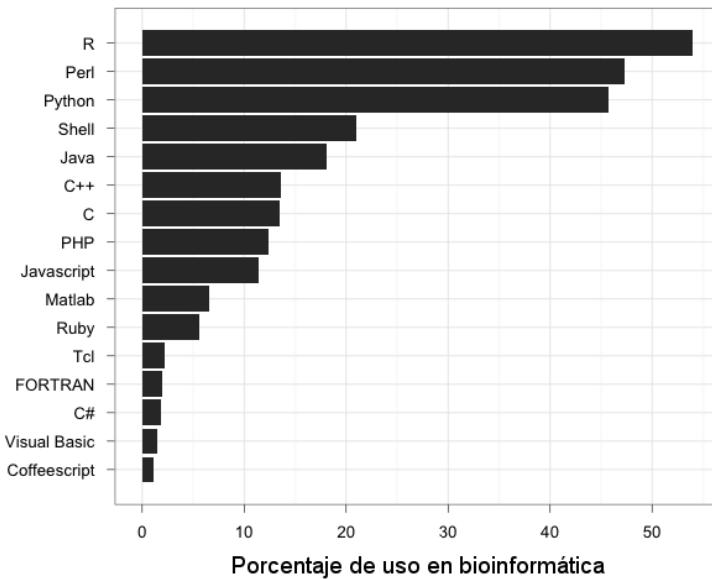
La *programación lógica* consiste en expresar una tarea empleando lógica formal matemática. Ejemplo: Prolog.

Otros paradigmas de programación son: programación genérica, programación reflexiva, programación orientada a procesos, etc.

#### 4.2.4. ¿Que lenguaje de programación usar?

La elección del lenguaje de programación nunca debería ser aleatoria o basada únicamente en preferencias personales. Ya sea más o menos fácil el aprender un determinado lenguaje de programación,

lo ideal es usar el más óptimo dependiendo de la tarea que vayamos a realizar. El *rendimiento* a la hora de ejecutar un determinado programa, puede ser uno de los criterios a tener en cuenta. El tiempo empleado en leer o “*parsear*” un fichero (que podríamos definir como obtener información a partir de datos, los cuales aparecen en un determinado formato no enriquecido), por ejemplo en formato FASTA, o en realizar un alineamiento de secuencias, depende en gran medida del lenguaje de programación a usar [1]. Lenguajes como C o C++, son conocidos por su alto rendimiento en cuanto a uso de procesador como a uso de memoria RAM, sin embargo son lenguajes de alto nivel que *a priori* sin ser informáticos, pueden resultar complicados de dominar. En la mayoría de situaciones deberemos buscar un compromiso entre la dificultad y el tiempo a emplear en escribir el programa y el rendimiento del mismo. No tiene sentido aprender un nuevo lenguaje para ahorrar una hora de cálculo. Si tenemos en cuenta el estudio sobre los lenguajes más usados en el campo de la bioinformática en el año 2012, mostrado en la Figura 4.1, podemos comprobar como R (lenguaje de programación para el análisis estadístico) junto a Perl y Python, son los más usados.



**Figura 4.1:** Porcentaje de uso de lenguajes de programación en 2012.

Un buen bioinformático, debería ser capaz de programar en varios lenguajes y usar uno u otro en función de sus necesidades y los recursos computacionales a su disposición, así como los algoritmos a usar y el tipo de datos de partida. En el presente texto escogeremos Perl como modelo de lenguaje de programación de alto nivel para explicar su funcionamiento a lo largo del capítulo. Veremos los tipos de datos con los que trabaja, esencialmente escalares, *arrays* y *hashes*. A su vez veremos estructuras de control, como bucles y sentencias condicionales y expresiones regulares entre otros.

### 4.3. Consola de Linux, Mac OS X y Windows. Operaciones básicas

Mirando al pasado de la informática, la forma de interactuar con los ordenadores de la época era con una *consola física* (Figura 4.2). Dicha consola física era un teclado unido a una pantalla, que estaba conectado por puerto serie al ordenador. Los contenidos que era capaz de mostrar esa consola eran únicamente textuales y monocromáticos, sin ninguna capacidad gráfica. Toda interacción con el

ordenador era a través de esa consola, tecleando órdenes y leyendo la salida proporcionada por esas órdenes.



**Figura 4.2:** Terminal DEC VT100, año 1978.

Esta forma de trabajo, aunque en principio parece arcaica, sigue vigente, debido a que es mucho más sencillo realizar el mismo conjunto de operaciones sobre muchos ficheros desde la consola que desde el entorno gráfico. Dicho conjunto de operaciones se pueden automatizar mediante la preparación de un lote de órdenes (o *script*), que es un guión de operaciones que dicta qué operaciones se realizan sobre los ficheros, y en qué orden. Actualmente, casi todos los sistemas operativos y entornos gráficos proporcionan programas de *consola virtual* que permiten seguir con esta forma de trabajo.

#### 4.3.1. Tipos de Shell

Una consola de texto (ya sea física o virtual) no sirve de nada sin un *intérprete de órdenes* para la línea de comandos (o *shell*). Una *shell* es un programa que entiende una serie de órdenes textuales comunes, como por ejemplo cambiar de directorio, mostrar los contenidos de un directorio, copiar, mover y borrar ficheros y directorios, y ejecutar programas.

En los sistemas Windows, la *shell* usada se llama ‘cmd’, mientras que para los sistemas derivados de UNIX, como por ejemplo Linux o Mac OS X, las *shells* que se usan principalmente son ‘bash’ y ‘tcsh’. Todas estas *shells* disponen de su propio *lenguaje de script*, que permite no sólo ejecutar una serie de órdenes una detrás de otra, sino también operaciones más complejas como aplicar las mismas operaciones a cada uno de los ficheros de un directorio, y manipular variables de entorno. Las variables de entorno rigen, entre otras cosas, dónde buscar los programas a ser ejecutados en la *shell*. La Tabla 4.1 muestra algunas de las operaciones básicas que se pueden realizar en una *shell*.

Operación básica	cmd	bash	tcsh
Listado de variables de entorno	SET		env, printenv
Valor de una variable de entorno	ECHO %VARIABLE%		echo \$VARIABLE
Establecer una variable de entorno	SET VARIABLE="valor"	export VARIABLE="valor"	setenv VARIABLE "valor"
Listado del directorio actual	DIR /W		ls

Contenidos de un directorio	<code>DIR /W directorio</code>	<code>ls directorio</code>
Contenido detallado	<code>DIR directorio</code>	<code>ls -l directorio</code>
Directorio actual	<code>ECHO %CD%</code>	<code>pwd</code>
Cambiar de directorio	<code>XS directorio</code>	<code>cd directorio</code>
Crear un directorio	<code>MKDIR directorio</code>	<code>mkdir directorio</code>
Copiar un fichero	<code>COPY fichero destino</code>	<code>cp fichero destino</code>
Copiar un directorio	<code>XCOPY /T /E directorio destino</code>	<code>cp -r directorio destino</code>
Borrar un fichero	<code>DEL fichero</code>	<code>rm fichero</code>
Borrar un directorio	<code>RD /S directorio</code>	<code>rm -r directorio</code>
Renombrar un fichero o directorio	<code>REN nombreViejo nombreNuevo</code>	<code>mv nombreViejo nombreNuevo</code>
Mover un fichero o directorio	<code>MOVE nombreViejo destino</code>	<code>mv nombreViejo destino</code>

**Tabla 4.1:** Operaciones básicas que se pueden ejecutar en una *shell*.

## 4.4. Perl y BioPerl

### 4.4.1. Introducción a Perl

Perl es el acrónimo de “lenguaje práctico para la extracción y el informe” (*Practical Extraction and Report Language*). Perl es un lenguaje de programación que toma características del lenguaje C, del lenguaje interpretado *bourne shell* (‘sh’), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación. Estructuralmente, Perl está basado en un estilo de bloques como los de C o AWK y fue ampliamente adoptado por su rendimiento en el procesado de texto.<sup>1</sup>

Fue diseñado por Larry Wall en 1987 y hasta 1991 la única documentación de Perl era una simple (y cada vez más larga) página de manual Unix. En 1991 se publicó *Programming Perl* [2] y se convirtió en la referencia de facto del lenguaje. Al mismo tiempo, el número de versión de Perl saltó a 4, no por marcar un gran cambio en el lenguaje, sino por identificar a la versión que estaba documentada en el libro. En este punto, Larry Wall abandonó Perl 4 para comenzar a trabajar en Perl 5, cuyo desarrollo continuó hasta 1994. Perl 5 fue publicado el 17 de octubre de 1994. Fue casi una completa reescritura del intérprete y añadió muchas nuevas características al lenguaje, incluyendo objetos, referencias, paquetes y módulos. A destacar, los módulos proveen de un mecanismo para extender el lenguaje sin modificar el intérprete. Esto permitió estabilizar su núcleo principal, además de permitir a los programadores de Perl añadirle nuevas características. El 26 de octubre de 1995, se creó el *Comprehensive Perl Archive Network*<sup>2</sup> (CPAN). CPAN es una colección de sitios web que almacenan y distribuyen fuentes en Perl, binarios, documentación, *scripts* y módulos. En 2012, Perl 5 es la versión más popular y continúa siendo mantenido. La nueva versión 6 de Perl 6 comenzó ser desarrollada en 2000 y todavía no ha sido terminada.

Perl está implementado como un *intérprete de comandos*, escrito en C, junto con una gran colección de módulos, escritos en Perl y C. El intérprete tiene una arquitectura orientada a objetos. Todos los elementos del lenguaje Perl —escalares, listas, hashes, referencias a código, manejadores de archivo— están representados en el intérprete como estructuras C. Las operaciones sobre estas estructuras están definidas como una numerosa colección de macros, *typedef* y funciones; estos constituyen la API C de

<sup>1</sup>Perl en Wikipedia. <http://es.wikipedia.org/wiki/Perl>

<sup>2</sup>Comprehensive Perl Archive Network. <http://www.cpan.org>

Perl. La API Perl puede ser desconcertante para el no iniciado, pero sus puntos de entrada siguen un esquema de nombres coherente, que ayuda a los que quieran utilizarla. La ejecución de un programa Perl se puede dividir en dos fases: tiempo de compilación y tiempo de ejecución. En *tiempo de compilación* el intérprete parsea el texto del programa en un árbol sintáctico. En *tiempo de ejecución*, ejecuta el programa siguiendo el árbol. El texto es parseado sólo una vez y el árbol sintáctico es optimizado antes de ser ejecutado, para que la fase de ejecución sea relativamente eficiente. Perl está diseñado para realizar tareas sencillas que con una *shell* sería complicado de llevar a cabo y facilita su escritura, comparada con otros lenguajes, como C.

Perl es software libre y está licenciado bajo la *GNU General Public License* y la *Artistic License*. Existen distribuciones disponibles para la mayoría de sistemas operativos. Está especialmente extendido en Unix y en sistemas similares, pero ha sido portado a las plataformas más modernas (y otras más obsoletas). Puede ser compilado desde el código fuente en todos los Unix, compatibles POSIX o cualquier otra plataforma Unix compatible. Sin embargo, esto no es normalmente necesario, porque Perl está incluido por defecto en la instalación de los sistemas operativos más populares a excepción de Windows.

#### 4.4.2. Instalación

##### Instalación Linux

Perl está instalado por defecto en las distribuciones más populares de GNU/Linux incluyendo Gentoo, Slackware, Mandriva, Debian, RedHat, SUSE y Ubuntu. En el caso de que quisieramos instalarlo, tendríamos dos opciones:

1. Usar *binarios precompilados*. Un binario precompilado no es más que el intérprete de perl, así como las librerías y los módulos necesarios para que funcione. Su instalación implica descargar el archivo, descomprimirlo y ejecutarlo desde la propia página de Perl<sup>3</sup> o la versión ofrecida por la empresa ActiveState<sup>4</sup>.
2. *Compilar perl desde el código fuente*. El código fuente, no es más que el código escrito directamente por el autor, para que nosotros creemos nuestros propios binarios. Las instrucciones a ejecutar en consola para instalar la versión 5.14 son las siguientes :

```
$ wget http://www.cpan.org/src/5.0/perl-5.14.2.tar.gz
$ tar -xzf perl-5.14.2.tar.gz
$ cd perl-5.14.2
$ ./Configure -des -Dprefix=$HOME/localperl
$ make
$ make test
$ make install
```

##### Instalación en MacOS

En el caso de los sistemas operativos de Apple, Perl viene incluido por defecto y no es necesario hacer nada especial. Simplemente abrir la consola y empezar a programar. En el caso de necesitar una instalación propia, ésta instalación, ya sea a través de binarios o del código fuente, es igual a la explicada anteriormente para los ordenadores que usan Linux. Después de todo, ambos proceden del sistema operativo UNIX.

---

<sup>3</sup>Página oficial de Perl. <http://www.perl.org>

<sup>4</sup>ActiveState Perl. <http://www.activestate.com>

## Instalación en Windows

Los usuarios de Windows normalmente instalan una distribución binaria de Perl. Compilar Perl desde el código fuente bajo Windows es posible, pero la mayoría de las instalaciones no disponen del necesario compilador de C. La opción más sencilla es descargar un fichero ejecutable desde la página web de ActiveState<sup>4</sup> que nos instalará todo lo necesario para que Perl funcione desde el interfaz de comandos propio de Windows. El sistema de emulación Cygwin<sup>5</sup> también proporciona otra forma de ejecutar Perl bajo Windows. Cygwin proporciona un entorno parecido al Unix en Windows que incluye ‘gcc’, por lo que compilar Perl desde el código es una opción accesible para los usuarios que prefieren esta opción.

### 4.4.3. Programando en Perl

A lo largo de esta sección, iremos profundizando en los detalles más relevantes de Perl. Empezando por el programa más sencillo. En Perl, el programa canónico ‘Hola mundo’ es:

```
#!/usr/bin/perl
print "Hola mundo\n"; # Imprime "Hola mundo"
```

La primera línea contiene el “shebang” (par de caracteres que identifica el texto que sigue), que le indica al sistema operativo dónde encontrar el intérprete de Perl. La segunda imprime el string “Hola mundo” y un carácter de nueva línea. Todos los comandos de Perl terminan en punto y coma (‘;’). Se pueden insertar comentarios en el código con el símbolo almohadilla (‘#’), todo lo que siga a dicho símbolo no será leído al ejecutar el código. El *shebang* es la forma normal para invocar al intérprete en los sistemas Unix. Los sistemas Windows pueden seguir utilizándolo o pueden asociar la extensión de archivo ‘.pl’ con el intérprete Perl. Algunos editores de texto también usan la línea *shebang* como una pista sobre el modo de trabajo en que deben operar. Si el programa es ejecutado por Perl y no invocado por el *shell*, la línea que empieza por el *shebang* es parseada para interpretar las opciones. En otro caso, es ignorada.

#### Tipos de datos en perl.

Perl tiene tres tipos de datos: escalares, listas y hashes:

- Un *escalar* es un solo valor; puede ser un número, un *string* (cadena de caracteres) o una referencia.
- Una *lista* es una colección ordenada de escalares (una variable que almacena una lista se llama *array*).
- Un *hash*, o memoria asociativa, es un mapeo de *strings* a escalares; los strings se llaman *claves* y los escalares *valores*.

Todas las variables están precedidas por un *sigilo*, que identifica el tipo de dato que es accedido (no el tipo de dato de la misma variable). Se puede usar el mismo nombre para variables de diferentes tipos sin que tengan conflictos. Los números se escriben de la forma usual, los *strings* (texto) están rodeados entre comillas, ya sean dobles o simples. Ver ejemplos de tipos de datos en el ??.

```
$variable;    # un escalar
$array;      # un array
%hash;        # un hash
$n           = 12;
```

---

<sup>5</sup>Cygwin project. <http://www.cygwin.com>

```
$nombre = "coche";
$color = 'rojo';
```

Una característica de Perl es que reconoce el tipo de dato de una variable. Perl convertirá los *strings* en números y viceversa dependiendo del contexto en que sean usados. En el siguiente ejemplo (??) los *strings* '\$n' y '\$m' son tratados como números cuando son argumentos del operador suma. Este código imprime el número '5', desechando cualquier información no numérica de la operación y dejando los valores de las variables intactos.

```
$n      = "3 casas";
$m      = "20 motos";
print $n + $m; # Resultado '5'
```

Una lista se define listando sus elementos, separados por comas y rodeados por paréntesis donde así sea requerido por la precedencia de los operadores:

```
@numeros = (32, 45, 16, 5);
```

Un *hash* puede ser inicializado desde una lista de pares clave/valor:

```
%color = (
    coche => 'rojo',
    moto  => 'azul',
);
```

Los elementos individuales de una lista son accedidos utilizando un índice numérico, dentro de corchetes. Valores individuales en un *hash* son accedidos utilizando la correspondiente clave, dentro de llaves. El símbolo '\$' identifica que el elemento accedido es un escalar.

```
$numeros[2];
$color{'moto'};
```

Múltiples elementos pueden ser accedidos usando en su lugar el sigilo '@' (identificando el resultado como una lista).

```
@numeros[2, 3, 1] # tres elementos de @puntuaciones
@color{'coche', 'moto'} # dos valores de %color
```

El número de elementos en un *array* puede ser obtenido mediante la función 'scalar' o con la ayuda del sigilo '\$#' éste último da el índice del último elemento dentro del array, que será una unidad menor que el número de elementos puesto que el índice del primer elemento del array es '0'.

```
scalar @amigos; # número de elementos en @amigos
$#amigos;       # índice del último elemento
```

Hay unas pocas funciones que operan sobre *hashes* enteros:

```
@nombres_de_clientes = keys  %direcciones;   # guarda todas las claves
@direcciones_de_email = values %direcciones;   # guarda todos los valores
```

## Estructuras de control

Perl tiene varias clases de estructuras de control, estructuras de control orientado al bloque o *bucles*, similar a los de los lenguajes de programación C y Java. Y *estructuras condicionales*. Las condiciones están rodeadas por paréntesis y los bloques subordinados por llaves:

```
while ( condición ) {...}
while ( condición ) {...} continue {...}
for ( expresión inicial; expresión condicional; expresión incremental ) {...}
foreach var ( lista ) {...}
foreach var ( lista ) {...} continue {...}
```

```

if ( condición ) {...}
if ( condición ) {...} else {...}
if ( condición ) {...} elsif ( condición ) {...} else {...}

```

Las palabras clave de control de flujo ‘next’, ‘last’, ‘return’ y ‘redo’ son expresiones, por lo que pueden ser usadas con los operadores cortocircuito. Perl también tiene dos construcciones implícitas para bucles:

```

resultados = grep {...} lista
resultados = map {...} lista

```

‘grep’ devuelve todos los elementos de lista en que el bloque subordinado evalúa a verdadero. ‘map’ evalúa el bloque subordinado por cada elemento de lista y devuelve una lista de los valores resultantes. Estas construcciones permiten un estilo simple de programación funcional. No hay declaración ‘switch’ (salto multi-camino) en Perl 5. La documentación Perl describe media docena de formas de conseguir el mismo efecto usando otras estructuras de control. Existe sin embargo un módulo Switch, que proporciona la funcionalidad modelada para el próximo Perl 6. Perl incluye una declaración ‘goto’ etiquetada, pero es usada raramente. Las situaciones donde en otros lenguajes se utiliza ‘goto’ no ocurren tan a menudo en Perl debido a sus amplias opciones de control de flujo. Existe también una declaración ‘goto &sub’ que realiza una llamada final. Termina la subrutina actual e inmediatamente llama a la subrutina especificada. Esto se usa en situaciones donde una nueva subrutina puede realizar una gestión de la pila más eficiente que el propio Perl (porque típicamente no se requiere ningún cambio en la pila actual), y en una recursión muy profunda este tipo de llamadas puede tener un sustancial impacto positivo en el funcionamiento porque evita la sobrecarga de la gestión contexto/pila en el momento de retornar.

## Subrutas

Las subrutas se definen con la palabra clave ‘sub’ e invocadas simplemente nombrándolas. Si la subrutina en cuestión no ha sido todavía declarada, es necesario, para el proceso de análisis sintáctico, poner los paréntesis.

```

foo();           # paréntesis necesarios aquí...
sub foo {...}
foo;            # pero no aquí

```

Una lista de argumentos pueden ser indicados después del nombre de la subrutina. Los argumentos pueden ser escalares, listas o *hashes*.

```
foo($x, @y, %z);
```

Los parámetros de una subrutina no necesitan ser declarados, ni en número ni en tipo, de hecho, pueden variar en cada llamada. Los *arrays* son expandidos a sus elementos, los *hashes* a una lista de pares clave/valor y todo el conjunto es pasado a la subrutina como una indiferenciada lista de escalares. Cualesquier de los argumentos pasados están disponibles para la subrutina en el *array* especial ‘@\_’. Los elementos de ‘@\_’ son asociados a los argumentos actuales; cambiando un elemento de ‘@\_’ cambia el argumento correspondiente. Los elementos de ‘@\_’ pueden ser accedidos con los subíndices de la forma normal:

```
$_[0], $_[1]
```

Sin embargo, el código resultante puede ser difícil de leer y los parámetros tener una semántica de pase por referencia, que puede resultar algo no deseable. Un modismo común es asignar ‘@\_’ a una lista de variables con nombres:

```
my ($x, $y, $z) = @_;
```

Esto afecta tanto a la mnemónica de los nombres de los parámetros como a la semántica de los valores pasados por valor. La palabra clave ‘my’ indica que las siguientes variables están léxicamente embebidas en el bloque que las contienen. Otro modismo es sacar los parámetros de ‘@\_’. Esto es muy común cuando la subrutina toma un sólo argumento:

```
my $x = shift; # Si no se dice nada, nos referimos a @_
```

Las subrutinas pueden devolver valores:

```
return 42, $x, @y, %z;
```

Si la subrutina no sale vía declaración ‘return’, entonces devuelve la última expresión evaluada en el cuerpo de la subrutina. *Arrays* y *hashes* en el valor de retorno son expandidos a una lista de escalares, igual que si fueran argumentos de una función. La expresión devuelta es evaluada en el contexto de la llamada de la subrutina; esto puede sorprender al desprevenido.

```
sub lista {      (4, 5, 6)      }
sub array { @x = (4, 5, 6); @x }

$x = lista; # devuelve 6 - último elemento de la lista
$x = array; # devuelve 3 - número de elementos de la lista
@x = lista; # devuelve (4, 5, 6)
@x = array; # devuelve (4, 5, 6)
```

## Expresiones regulares con Perl

El lenguaje Perl incluye una sintaxis especializada para escribir *expresiones regulares* y el intérprete contiene un motor para emparejar *strings* con expresiones regulares. El motor de expresiones regulares usa un *algoritmo de vuelta atrás* (*backtracking*), extendiendo sus capacidades desde el simple emparejamiento de patrones simples con la captura y sustitución de *strings*. El motor de expresiones regulares se deriva de ‘regex’, escrito por Henry Spencer. La sintaxis de expresiones regulares fue originalmente tomada de las expresiones regulares de Unix 8. Sin embargo, se diferenció ya antes del primer lanzamiento de Perl y desde entonces ha ido incorporando muchas más características. Otros lenguajes y aplicaciones están adoptando las expresiones regulares de Perl denominadas PCRE en vez de las expresiones regulares POSIX, incluyendo PHP, Ruby, Java y el Servidor HTTP Apache. El operador ‘m//’ (empareja) permite comprobar un emparejamiento por medio de una expresión regular (para abreviar, el precedente ‘m’ puede ser omitido). En el caso más simple, una expresión como:

```
$x =~ m/abc/;
```

evalúa a verdadero si y sólo si el *string* ‘\$x’ empareja con la expresión regular ‘abc’. Partes de la expresión regular pueden ser incluidas entre paréntesis y las partes correspondientes de un *string* emparejado son capturadas. Los *strings* capturados son asignados de forma secuencial a las variables internas ‘\$1’, ‘\$2’, ‘\$3’,... y una lista de *strings* capturados se devuelve como valor del emparejamiento:

```
$x =~ m/a(.)c/; # captura el carácter entre 'a' y 'c' y lo guarda en $1
```

El operador ‘s///’ (sustitución) especifica una operación de búsqueda y reemplazo:

```
$x =~ s/abc/aBc/; # Convierte la 'b' en mayúscula
```

Las expresiones regulares en Perl pueden tomar unos modificadores. Son sufijos de una sola letra que modifican el significado de la expresión:

```
$x =~ m/abc/i;      # emparejamiento independientemente de si están en mayúscula o minúscula
$x =~ s/abc/aBc/g;  # búsqueda y reemplazo recursivo global (a lo largo de todo el string)
```

Las expresiones regulares pueden ser densas y crípticas. Esto es porque la sintaxis de las expresiones regulares es extremadamente compacta, generalmente usando caracteres sueltos o pares de caracteres que representan sus operaciones.

Un uso común de las expresiones regulares es el de especificar delimitadores de campos al operador ‘split’:

```
@palabras = split m/,/, $linea; # divide $línea en valores separados por comas.
```

El operador ‘split’ complementa la captura de *string*. La captura de *string* devuelve las partes que emparejan con una expresión regular y ‘split’ devuelve las partes que no emparejan.

## Ejemplos simples

**Ejemplo 1.** Cómo convertir un secuencia de DNA en RNA (Código 4.1). Como sabéis, el DNA está compuesto de 4 tipo de nucleótidos denominados, adenina (A), guanina (G), citosina (C) y timina (T) (Sección 7.3). En el caso del RNA la base equivalente a la timina (T) es el uracilo (U).

Código 4.1: Traduciendo DNA en RNA.

```
1 #!/usr/bin/perl
2 #Nuestra secuencia de DNA
3 $secuencia_DNA="AGCTTGACGTGACACATG";
4 #Como RNA es igual al DNA (a excepción del uracilo), copiamos el valor
5 $secuencia_RNA=$secuencia_DNA;
6 #Ahora cambiamos las timinas (T) por uracilos (U) a lo largo de todo el string
7 $secuencia_RNA =~ s/T/U/g;
8 #Imprimimos el resultado
9 print "La secuencia de DNA '$secuencia_DNA' transformada en RNA es '$secuencia_RNA'\n";
```

El resultado debe ser:

```
La secuencia de DNA 'AGCTTGACGTGACACATG' transformada en RNA es
'AGCUUGACGUGACACAUG'.
```

**Ejemplo 2.** Cómo convertir un secuencia de RNA en un fichero en DNA (Código 4.2). En este caso, vamos a hacer el caso contrario, es decir convertir una secuencia de RNA en DNA. La complicación viene por el hecho de leer la secuencia directamente desde un fichero en nuestro ordenador. Lo primero que necesitamos es un fichero con una secuencia de RNA. Por lo tanto usando un editor de texto, creamos un fichero llamado ‘secuenciaRNA.txt’, cuyo contenido sea el siguiente:

```
UUACGGCGGAGCUUUGACGUACGAUCGAUCG
```

Código 4.2: Traduciendo RNA de un fichero en DNA.

```
1 #!/usr/bin/perl
2 #Pedimos al usuario que introduzca la ruta al fichero con la secuencia
3 print "Introduce el fichero con tu secuencia: ";
4 #El parámetro introducido (la ruta al fichero), se asocia a la variable $fichero_RNA.
        STDIN = Standard Input
5 $fichero_RNA = <STDIN>;
6 #Al escribir la ruta, presionamos la tecla Intro, la cual no forma parte del nombre del
        fichero. Por eso se elimina con la función chomp.
7 chomp $fichero_RNA;
8 #Abrimos el fichero cuyo contenido será una secuencia. Si no se puede abrir, el programa
        muere dando un error.
9 open(RNAFILE, $fichero_RNA) || die "No puedo abrir el fichero ($fichero_RNA)\n";
10 #Asociamos a la variable RNA_secuencia, el contenido de fichero $fichero_RNA
```

```

11 $secuencia_RNA = <RNAFILE>;
12 #Como RNA es igual al DNA (a excepción del uracilo), copiamos el valor.
13 $secuencia_RNA=$secuencia_DNA;
14 #Ahora cambiamos las timinas por uracilos a lo largo de todo el string.
15 $secuencia_RNA =~ s/T/U/g;
16 #Imprimimos el resultado
17 print "La secuencia de DNA '$secuencia_DNA' transformada en RNA es '$secuencia_RNA'\n";

```

El resultado debe ser :

```

Introduce el fichero con tu secuencia: secuenciaRNA.txt
La secuencia de RNA 'UUACGGCGGAGCUUUGACGUACGAUCG' trasformada en DNA es
'TTACGGCGGAGCTTGACGCTACGATCGATCG'

```

**Ejemplo 3.** Cómo obtener la secuencia complementaria a una secuencia de DNA (Código 4.3). Lo primero que necesitamos es de nuevo un fichero con una secuencia de DNA. Por lo tanto usando un editor de texto, creamos un fichero llamado ‘secuenciaDNA.txt’, cuyo contenido sea el siguiente:

```
TGTGGTACGTACGACAAACGTCGTACGTA
```

Código 4.3: Secuencia reversa complementaria.

```

1#!/usr/bin/perl
2#Pedimos al usuario que introduzca la ruta al fichero con la secuencia
3print "Introduce el fichero con tu secuencia: ";
4#El parámetro introducido (la ruta al fichero), se asocia a la variable $fichero_DNA.
5$STDIN = Standard Input
6$fichero_DNA = <STDIN>;
7#Al escribir la ruta, presionamos la tecla Intro, la cual no forma parte del nombre del
8#fichero. Por eso se elimina con la función chomp.
9chomp $fichero_DNA;
10#Abrimos el fichero cuyo contenido será una secuencia. Si no se puede abrir, el programa
11#muere dando un error.
12open(DNAFILE, $fichero_DNA) || die "No puedo abrir el fichero ($fichero_DNA)\n";
13#Asociamos a la variable $DNA, el contenido de fichero $fichero_DNA
14$DNA = <DNAFILE>;
15#Creamos un array @nucleotidos, de forma que en cada posición del array, haya un nucle
16#ótido procedente de $DNA
17@nucleotidos=split(//,$DNA);
18#Definimos un hash para hacer las conversiones en la hebra contraria a la nuestra.
19my %hash_complementarias;
20$hash{a}="t";
21$hash{c}="g";
22$hash{g}="c";
23$hash{t}="a";
24$hash{u}="a";
25#Usamos un bucle para recorrer de atrás hacia adelante la secuencia. Así la obtenemos de
26#forma inversa
27print "La secuencia reversa complementaria de DNA '$DNA' es ''";
28for ($var=$#nucleotidos; $var>=0; $var--) {
29    #Cada posición del array @nucleotidos, por ejemplo $nucleotidos[5].
30    #Pero como lo que queremos es su complementario, el nucleótido resultado se obtiene
31    #usando nuestro hash de secuencias complementarias
32    print $hash{$nucleotidos[$var]};
33}
34print "'\n";

```

El resultado debe ser:

```

La secuencia reversa complementaria del DNA 'TGTGGTACGTACGACAAACGTCGTACGTA' es
'ATGCATGCTGCAAACAGCATGCATGGTGT'.

```

**Ejemplo 4.** Cómo obtener la longitud de una secuencia de nucleótidos (Código 4.4). En este último ejemplo haremos uso de una subrutina llamada ‘contador’ que tomará como valor de entrada una variable con la secuencia y retornará como salida su longitud. Usaremos la misma secuencia creada en el ejemplo anterior en el fichero llamado ‘secuenciaDNA.txt’.

Código 4.4: Longitud de una secuencia.

```
1 #!/usr/bin/perl
2 #Pedimos al usuario que introduzca la ruta al fichero con la secuencia
3 print "Introduce el fichero con tu secuencia: ";
4 #El parámetro introducido (la ruta al fichero), se asocia a la variable
# $fichero_secuencia. STDIN = Standard Input
5 $fichero_secuencia = <STDIN>;
6 #Al escribir la ruta, presionamos la tecla Intro, la cual no forma parte
# del nombre del fichero. Por eso se elimina con la función 'chomp'.
7 chomp $fichero_secuencia;
8 #Abrimos el fichero cuyo contenido será una secuencia. Si no se puede abrir, el programa
# muere dando un error.
9 open(SEQFILE, $fichero_secuencia) || die "No puedo abrir el fichero ($fichero_secuencia)
\n";
10 #Asociamos a la variable $secuencia, el contenido de fichero $fichero_secuencia
11 $secuencia = <SEQFILE>;
12 #Enviamos la secuencia a una subrutina que nos devolverá el número de elementos
13 $longitud_secuencia=contador($secuencia);
14 print "El tamaño de la secuencia '$secuencia' es: $longitud_secuencia\n";
15 #Subrutina para contar los elementos
16 sub contador(){
17     #Usamos el my, para que la variable $secuencia sólo exista dentro de la subrutina y no
# interfiera con la usada anteriormente
18     my $secuencia=shift;
19     #Separamos el string de la secuencia en cada uno de sus elementos
20     @split=split('//,$secuencia);
21     #devolvemos el número total de elementos
22     return(scalar(@split));
23 }
24 }
```

Al usar éste programa con el fichero usado anteriormente, el resultado es el siguiente:

```
Introduce el fichero con tu secuencia: secuenciaDNA.txt
El tamaño de la secuencia 'TGTGGTACGTACGACAAACGTCGTACGTA' es: 29
```

#### 4.4.4. BioPerl y la programación orientada a objetos

##### ¿Qué es BioPerl?

BioPerl es una colección de módulos de Perl que facilitan el desarrollo de *scripts* en Perl para aplicaciones de bioinformática [3]. Ha desempeñado un papel integral en el *Proyecto Genoma Humano*.<sup>6</sup> Se trata de un activo proyecto de software libre apoyado por la *Open Bioinformatics Foundation*. La primera versión estable fue lanzada el 11 de junio de 2002; la última estable (en términos de la API) es la 1.6.9, lanzada en abril de 2011. La versión 1.6.0 es considerada la más estable (en términos de errores) y la recomendada para el uso diario. También hay lanzamientos periódicos producidos por desarrolladores. Con el fin de aprovechar BioPerl, el usuario necesita una comprensión básica del lenguaje de programación Perl que incluya una comprensión de cómo Perl utiliza las referencias, módulos, objetos y métodos.

<sup>6</sup>BioPerl en Wikipedia. <http://es.wikipedia.org/wiki/BioPerl>

## Instalación de BioPerl

**Instalación en Linux/MacOs.** La instalación que recomendamos en los sistemas basados en Unix, es la basada en CPAN. Como se explicó anteriormente (Subsección 4.4.1), en CPAN podemos encontrar multitud de paquetes que nos facilitarán mucho a la hora de programar. Siguiendo las instrucciones de BioPerl [3], los pasos son los siguientes: Desde la consola ejecutamos:

```
$ perl -MCPAN -e shell  
cpan> force install C/CJ/CJFIELD/S/BioPerl-1.6.0.tar.gz
```

Aceptando las opciones por defecto, la instalación se realizará sin problemas.

**Instalación en Windows.** En el caso de Windows, la instalación es más compleja, ya que carecemos del compilador de C. La opción más recomendada es instalar la versión de Perl de ActiveState (Sección 4.4.2) y ejecutar el administrador de paquetes de Perl (*Perl Package Manager*) desde la consola:

```
C:> ppm-shell  
ppm>  
ppm> install PPM-Repositories  
ppm> repo add http://bioperl.org/DIST  
ppm> repo add uwinnipeg  
ppm> repo add trouchelle  
ppm> install BioPerl
```

### 4.4.5. Programando con BioPerl

En la introducción de este capítulo se explicó lo que es una clase, un objeto y una función (Subsección 4.2.3), ahora en este apartado lo veremos con ejemplos. Para ellos vamos a usar la clase secuencia ‘Bio::Seq’, para crear un objeto que contenga nuestra secuencia de interés. Una vez creado, usaremos varias funciones de BioPerl para procesar la secuencia.

```
$seq = Bio::Seq->new(  
    -seq => 'ATGGGGTGGTACCT',  
    -id  => 'mi_secuencia');
```

Podemos trabajar con el objeto secuencia usando métodos propios de BioPerl. En el siguiente ejemplo, reescribiremos el Código 4.3 usando métodos de BioPerl.

Código 4.5: Secuencia reversa complementaria con BioPerl.

```
1#!/usr/bin/perl  
2#Cargamos la clase Seq de BioPerl. Seq es la clase cuyos funciones nos permite trabajar  
3#con secuencias  
4use Bio::Seq;  
5#Creamos un objeto secuencia, con su secuencia y su nombre.  
6$seq = Bio::Seq->new(  
7    -seq => 'TGTGGTACGTACGACAAACGTCGTACGTA',  
8    -id  => 'mi_secuencia');  
9#usamos la funcion revcom para obtener la reversa complementaria  
10$reversa_complementaria=$seq->revcom();  
11#Una vez tenemos el objeto con la reversa complementaria, usamos la función seq  
12#para obtener una string con la secuencia.  
13$secuencia_revcomp= $reversa_complementaria->seq();  
14print "La secuencia reversa complementaria de DNA '" . $seq->seq() . "' es '  
     $secuencia_revcomp'\n";
```

El resultado, al igual que el código previo es:

La secuencia complementaria de DNA: 'TGTGGTACGTACGACAAACGTCGTACGTA' es  
'TACGTACGACGTTGTCGTACGTACCAACA'.

En el ejemplo anterior, nosotros mismos tuvimos que programar un bucle inverso para recorrer la secuencia de forma reversa y para obtener los nucleótidos complementarios creamos un *hash* con las equivalencias. En el caso de BioPerl, sólo tenemos que aplicar el método 'revcom' a nuestro objeto secuencia y ya lo tenemos.

#### 4.4.6. Ejercicios con BioPerl

**Ejemplo 1.** El primer ejemplo, nos servirá para crear la secuencia de proteína a partir de un secuencia de DNA (Código 4.6).

Código 4.6: Traduciendo DNA a proteína

```
1 #!/usr/bin/perl
2 use Bio::PrimarySeq;
3 #Creamos un objeto secuencia
4 my $seq = Bio::PrimarySeq->new(
5   -seq =>'AACCCCTAGCACTGCGCCGAAATATGGCATCCGTGGTATCCGACTCTGCTGCTGTTAAAAA',
6   -primary_id => 'Secuencia'
7 );
8 #Usamos el método 'translate', que retorna un objeto secuencia de aminoácidos
9 #y a su vez el método 'seq', para que directamente me imprima el resultado.
10 print $seq->translate()->seq() . "\n";
```

**Ejemplo 2.** El segundo ejemplo, lo complicaremos un poco más y va a constar de un análisis de enzimas de restricción sobre una secuencia (Código 4.7). De forma que veremos si existe alguna enzima capaz de cortar en nuestra secuencia de estudio. Obtendremos las enzimas y los fragmentos cortados.

Código 4.7: Análisis de enzimas de restricción.

```
1 #!/usr/bin/perl
2 use Bio::Restriction::EnzymeCollection;
3 use Bio::Restriction::Analysis;
4
5 # Creamos un objeto secuencia
6 my $seq = Bio::PrimarySeq->new(
7   -seq =>'AGCTTAATTCTTACCTAGCTCTGACTGCAACGGGCAATATGTCTC',
8   -primary_id => 'Secuencia'
9 );
10 #Creamos un objeto análisis de restricción cuyo parámetro es mi secuencia
11 my $ra = Bio::Restriction::Analysis->new(-seq=>$seq);
12 #obtenemos todas las enzimas de restricción capaces de cortar a nuestra secuencia
13 my $all_cutters = $ra->cutters;
14 #Por cada enzima que puede cortar, obtenemos su nombre y la secuencia donde corta
15 foreach my $enz ( $all_cutters->each_enzyme ) {
16   #con el método fragments() obtenemos los fragmentos cortados
17   my @fragments = $ra->fragments($enz);
18   #recorremos los fragmentos (si hay repetición de secuencia se obtendría más de un
19   #corte para la misma enzima, por eso nos devuelve un array)
20   foreach $frag (@fragments) {
21     #imprimimos el nombre (con el método 'name') y el fragmento
22     print $enz->name ."\t" . $frag ."\n";
23 }
```

#### **4.4.7. Resumen**

Como ya hemos comentado a lo largo del capítulo, Perl es un lenguaje maduro y ampliamente utilizado en bioinformática. Es fácil de aprender y su sintaxis es similar al lenguaje inglés hablado, lo que le hace especialmente útil para aquellos con poca experiencia en el mundo de la programación.

De Perl podemos destacar principalmente: la enorme utilidad de las expresiones regulares (PCRE) (Sección 4.4.3) y el repositorio CPAN (Subsección 4.4.1) que contiene infinidad de módulos y funciones que completan al lenguaje y nos ayudan a la hora de realizar casi cualquier tarea, incluyendo BioPerl (Subsección 4.4.4) para la realización de numerosas tareas bioinformáticas.



## 4.5. Bibliografía

- [1] M. Fourment and M. R. Gillings. *A comparison of common programming languages used in bioinformatics.*, volume 9. 2008.
- [2] R. L. Schwartz and L. Wall. *Programming Perl*. jan 1991.
- [3] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. R. Gilbert, I. Korf, H. Lapp, H. Lehväslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, and E. Birney. The bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618, oct 2002.



# Capítulo 5

## Minería de datos

*José María Fernández y Beatriz García Jiménez*

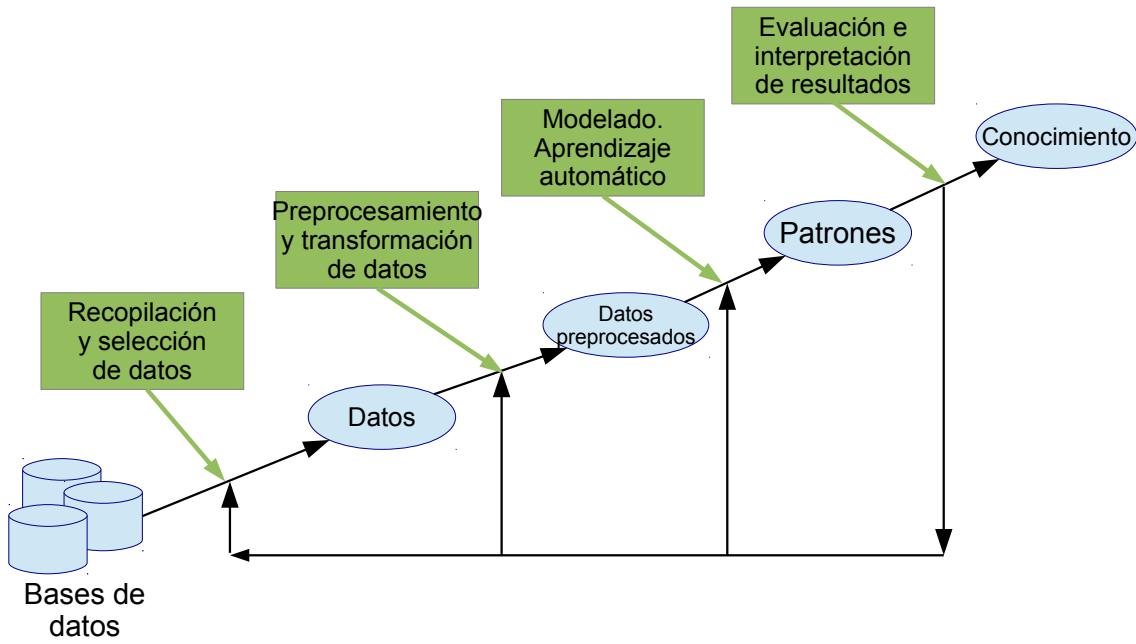
En el presente capítulo se explicarán los pasos del proceso de extracción del conocimiento de bases de datos, con las alternativas posibles en cada uno. Se verán ejemplos prácticos de cómo aplicarlos utilizando las herramientas disponibles, por ejemplo el entorno gráfico Weka. La descripción de las bases de los algoritmos concretos de aprendizaje automático (redes de neuronas, árboles de decisión, máquinas de vector de soporte vectorial, etc.) queda fuera del ámbito de este capítulo.

### 5.1. Introducción

La *inteligencia artificial* (IA) se define como “la ciencia e ingeniería dedicadas a la creación de máquinas o programas inteligentes”, según John McCarthy, quien acuñó el término en 1956. En otros términos, la IA es un área dentro de las Ciencias de la Computación que trata de resolver automáticamente problemas [46]. Está dividida en diversos campos, siendo los más destacables la representación del conocimiento, búsqueda, razonamiento y aprendizaje, planificación, agentes, procesamiento del lenguaje natural, visión artificial y robótica. La IA permite múltiples aplicaciones, que a grandes rasgos se agrupan en la predicción y el diagnóstico automático, la optimización de parámetros, la agrupación de muestras, etc.

Utilizando técnicas de IA se puede realizar *minería de datos*, que es una fase del proceso de descubrimiento de conocimiento en bases de datos (del inglés, *knowledge discovery in databases, KDD*) [13]. El proceso de KDD se divide en distintas fases, como muestra la Figura 5.1, que se deben realizar consecutivamente, aunque se pueda volver a alguna de ellas, a lo largo del proceso completo, dependiendo de los resultados obtenidos. Cada una de estas fases se irá detallando en una sección independiente en el resto de este capítulo, a partir de la Sección 5.3, tras presentar los retos de dicho proceso en bioinformática.

Como se puede observar en la Figura 5.1, la minería de datos se considera sólo una fase del KDD. A pesar de esta distinción entre KDD y minería de datos como “*todo y parte*”, respectivamente, en otros casos ambos términos se consideran sinónimos [42]. Independientemente de la denominación utilizada, el objetivo es el mismo: encontrar regularidades en los datos, que se puedan utilizar para realizar predicciones futuras que ayuden a la toma de decisiones. Para ello, se necesita llevar a cabo todas las fases descritas anteriormente, aunque la más conocida sea sólo la que aplica explícitamente algoritmos



**Figura 5.1:** Fases del proceso KDD.

de AA. En este capítulo se adopta la segunda visión: denominar a todo el proceso minería de datos, porque es el enfoque más extendido fuera de la comunidad de IA y AA, como es el caso de la biología computacional.

### 5.1.1. Aplicaciones de la minería de datos en bioinformática

La minería de datos se aplica ampliamente en biología [28, 34, 50]. Larrañaga et al. [28] muestran una perspectiva de todas las tareas biológicas en las que se aplican los algoritmos de *aprendizaje automático* (AA): en genómica, proteómica, metabolómica, evolución, biología de sistemas y minería de textos, poniendo de manifiesto que la aplicación de AA a biología molecular es un área bien establecida, desde hace tiempo y que continua vigente. Para un repaso bibliográfico exhaustivo, se pueden consultar las revisiones de aplicación a biología de AA (y otros sistemas de IA) [15, 34].

A continuación se presentan algunos ejemplos de uso de diferentes técnicas de AA [41] para distintos dominios de la biología molecular [50]:

- La definición de sitios de iniciación de la traducción en los genes de *E.coli* fue uno de los primeros usos de las redes de neuronas en bioinformática.
- La localización de genes en la secuencia de ADN es una de las aplicaciones más importantes del AA en este área, usando árboles de decisión o clasificadores bayesianos, entre otros; incluso combinando diferentes fuentes de información.
- Para el reconocimiento de regiones promotoras de la transcripción se han utilizado redes de neuronas, modelos de Markov, computación evolutiva y métodos del vecino más cercano.
- La detección de sitios de ensamblaje alternativo se ha realizado con modelos ocultos de Markov.
- Para la predicción de genes involucrados en enfermedades genéticas se han utilizado árboles de

decisión.

- En la predicción de los efectos fenotípicos de polimorfismos de nucleótidos aislados (del inglés, *single nucleotide polymorphisms*, SNP) no sinónimos se han comparado los resultados de máquinas de vector de soporte y bosques de árboles aleatorios, usando información estructural y evolutiva.
- A partir de datos de análisis de ADN, se usa el algoritmo C4.5 para extraer reglas que proporcionen conocimiento comprensible por el humano.
- Para la predicción de la estructura secundaria de proteínas se ha empleado el método del vecino más cercano.
- Para la predicción de estructura terciaria de las proteínas y clasificación en familias de proteínas se han utilizado redes bayesianas.
- Para el reconocimiento de patrones partiendo de datos de microarrays se encuentran múltiples técnicas de aprendizaje supervisado y de agrupamiento, consiguiendo principalmente subconjuntos de genes (perfiles de expresión génica) implicados en la diagnosis de cáncer.
- Para predecir si la respuesta reguladora de un gen es alta o baja se utiliza un conjunto de árboles de decisión.
- PSORT II realiza localización sub-cellular usando árboles de decisión inicialmente, y k-vecinos más cercanos en una versión posterior.
- ProtFun realiza anotación funcional con combinaciones de redes de neuronas.
- Se realiza modelado de redes de regulación génica, por ejemplo, con redes de neuronas y optimización por enjambre de partículas.
- Para la reconstrucción de árboles filogenéticos se suelen aplicar otros sistemas inteligentes, como es la computación evolutiva, por ejemplo los algoritmos genéticos o las colonias de hormigas.
- También hay que mencionar la presencia de sistemas de inteligencia computacional en software comercial, especialmente para el descubrimiento de fármacos.

Tras este breve repaso se puede concluir que el AA es una técnica de pasada, continua y futura aplicación para resolver problemas en múltiples dominios de la biología molecular.

### 5.1.2. Conceptos básicos

Antes de continuar, es necesario establecer la nomenclatura y conceptos clave que se utilizan en minería de datos, donde el principal objetivo es el reconocimiento automático de patrones en un conjunto de datos.

Así, un *conjunto de datos* es una lista de ejemplos, muestras o instancias; por ejemplo una lista de genes, de ORFs o de proteínas.

Cada *ejemplo* es el elemento individual utilizado para aprender, o sobre el que dar una predicción. Un ejemplo se representa por un conjunto de atributos o características asociadas al mismo. Los *atributos* pueden ser muy variados tanto en cantidad como en contenido, y entre ellos se suele encontrar el atributo clase. Por ejemplo, para una proteína se pueden incluir atributos básicos que se pueden extraer fácilmente de la secuencia, como la carga, la longitud, el punto isoeléctrico; otros ligeramente más complejos, como qué tipos de dominios tiene y cuántos; o incluso atributos más elaborados, como un

cálculo numérico sobre una red de interacción a la que pertenezca la proteína, o también la salida de otro predictor basado en AA.

Un *patrón* es la combinación de un conjunto de atributos, con un valor determinado para cada uno, que caracterizan la pertenencia a una clase.

Una *clase* se suele definir mediante un conjunto de etiquetas o valores, que pueden ser nominales, pero también numéricas. Si el objetivo es binario, los valores de la clase serán sólo dos: sí/no, o 1/0. Por ejemplo, para predecir si un gen está implicado en el desarrollo del cáncer de mama, los genes anotados con alguna relación con dicha anomalía tendrían la etiqueta “sí”, y para los que no existe anotación, tendrían la etiqueta “no”. Si la clasificación es entre más de dos categorías, los valores de la clase pueden ser múltiples. Por ejemplo, para predecir la localización celular de una proteína, el atributo clase podría tomar un valor para cada orgánulo, otro para el núcleo y otro para el citoplasma y otro para la membrana, o incluso utilizar los más de 3000 términos de la ontología de componente celular de Gene Ontology (ver Subsección 1.5.2). Incluso un ejemplo podría tener asociada más de un valor de la clase, por ejemplo, si la predicción es de función.

Cuando los ejemplos de un conjunto de datos tienen clases asociadas, dicho conjunto se suele dividir en un *conjunto de entrenamiento* y un *conjunto de prueba* (o evaluación o *test*). El primero se utiliza para construir el modelo, y el segundo para evaluar la capacidad predictiva del mismo, comparando la clase predicha con la clase original conocida. Por último, el conjunto de ejemplos sobre el que se usa el modelo una vez generado se denomina *conjunto de aplicación*, e incluye las instancias de las que no se conoce la clase. Por ejemplo, para un problema de anotación funcional, los conjuntos de entrenamiento y prueba los formarían los genes con anotación, en un determinado vocabulario, y el conjunto de aplicación, los genes sin anotación en ese vocabulario de funciones.

En los algoritmos más extendidos y usados de AA se sigue el *AA inductivo*. Éste asume que si un determinado patrón se cumple sobre una cantidad suficiente de ejemplos del conjunto de entrenamiento, también se verificará sobre nuevos ejemplos no utilizados para construir el modelo de predicción. En este principio se apoya su capacidad predictiva.

### 5.1.3. Herramienta: Weka

Weka (del inglés, *Waikato Environment for Knowledge Analysis*) significa “Entorno para Análisis del Conocimiento de la Universidad de Waikato” [24]. Weka se usa desde hace tiempo en aplicaciones bioinformáticas [16], como por ejemplo: [18, 26, 49].

Weka se puede instalar fácilmente descargando el software de su sitio web <sup>1</sup>, para distintos sistemas operativos, o ejecutarlo sin instalación previa, invocando el fichero ‘weka.jar’ por una consola de comandos:

```
$ java -jar weka.jar
```

aunque es recomendable aumentar la memoria que usa por defecto, sobre todo en bioinformática, donde la cantidad de datos empleados suele ser considerable. Por ejemplo, podemos aumentar la memoria RAM reservada para este programa a 1GB:

```
$ java -Xmx1024M -jar weka.jar
```

Weka permite realizar casi todas las fases del proceso de KDD presentado en la Figura 5.1, excepto la inicial de recopilar los datos. A continuación se presenta brevemente el funcionamiento básico de Weka, en su modo más frecuente, denominado ‘Explorer’.

---

<sup>1</sup>Waikato Environment for Knowledge Analysis. <http://www.cs.waikato.ac.nz/ml/weka>

Para un conocimiento más profundo de Weka, además del manual de ayuda incluido en la instalación de la aplicación, existen diversos tutoriales y mucha documentación adicional en la web para esta herramienta<sup>1</sup>. Pero la principal referencia es el libro escrito por los autores originales de la herramienta [57]. Por lo tanto, para cualquier duda concreta sobre el funcionamiento de Weka lo más adecuado es consultar el manual principal o el libro, dado que en este capítulo sólo se presentan las nociones básicas para poder empezar a trabajar con Weka.

En el modo ‘Explorer’, hay diferentes pestañas encargadas de las distintas fases del proceso de minería de datos. En la pestaña ‘Preproces’ se cargan los datos recopilados, y se pueden aplicar los preprocesamientos y transformaciones, como se explica en la Sección 5.4. Para cargar los datos, generalmente se hace desde un fichero en el formato ARFF (del inglés, *Attribute-Relation File Format*) propio de Weka, descrito en la Figura 5.2, o desde un fichero de texto, con los atributos separados por tabulaciones o comas, que Weka importa automáticamente al formato ARFF. No obstante, Weka también permite cargar los datos desde una URL o desde una base de datos a través de ODBC.

```
% Definición de atributos
% Comentarios precedidos de %
@relation proteins
@attribute id_protein string
@attribute length numeric
@attribute charge real
@attribute transmembrane_domain boolean
@attribute class {membrane, extracellular_matrix, organelle}

% Definición de datos
@data
ENST00000417324,203,0.082353,yes,membrane
ENST00000378486,1416,0.122175,no,extracellular_matrix
ENST00000377573,469,0.138593,no,extracellular_matrix
...
...
```

**Figura 5.2:** Ejemplo de datos en formato ARFF.

La tercera y cuarta fase del proceso de minería de datos se realiza con las pestañas ‘Classify’, ‘Associate’ y ‘Cluster’, donde se seleccionan y configuran los algoritmos de AA, se ejecutan, y se ven los resultados, en formato texto y gráfico, según se explica en la Sección 5.6. El modo ‘Explorer’ de Weka también permite realizar actividades adicionales del proceso de minería de datos aquí presentado, como es la visualización por pares de atributos, en la pestaña ‘Visualize’.

Las operaciones que se realizan en la interfaz de la herramienta Weka pulsando botones (en el modo ‘Explorer’), también se pueden realizar por línea de comandos, o de forma gráfica con diagramas de flujo (en el modo ‘Knowledge Flow’).

Debido a que Weka es un proyecto de código libre, permite que se modifique el código Java de sus algoritmos y filtros existentes, según las necesidades de cada uno. Incluso permite que el interesado se programe sus propios algoritmos en Java, siguiendo su estructura de clases, para que también pueda ser incluido como un método más de la aplicación completa, si se desea. Gracias a esta característica, la herramienta Weka ha ido ampliándose y perfeccionándose con la colaboración de múltiples interesados en el AA por todo el mundo.

Weka es útil para un principiante por la facilidad que proporciona su interfaz gráfica. Con el software R, presentado en el Capítulo 3, también se pueden realizar operaciones de minería de datos. Pero en este capítulo se presenta Weka como alternativa gráfica, aunque adicionalmente Weka permite también

el uso de la línea de comandos, por si se necesita incluir en procesamientos automáticos, como suele ser común en bioinformática.

También cabe mencionar la existencia del software RapidMiner [40] como una de las últimas alternativas para realizar operaciones de minería de datos. Permite importar los algoritmos tanto de Weka como de R. Su principal uso es en entornos reales y empresariales, aunque empieza a existir un uso creciente en educación e investigación.

## BioWeka

BioWeka [21] es una extensión de la herramienta Weka para facilitar el manejo de datos biológicos, como secuencias o datos de expresión génica, que no es posible en la herramienta Weka original. BioWeka surge por la necesidad de la bioinformática de aplicar muchos cambios de formato, y unir datos de salida de una interfaz con los de otra, antes de poder aplicar un algoritmo de AA. Se invierte mucho tiempo en programar para realizar estas transformaciones, por lo que BioWeka decide proporcionar un amplio rango de conversores y filtros frecuentemente utilizados en biología computacional, para facilitar o simplificar principalmente la fase de preprocesamiento.

La principal ventaja frente a Weka para los problemas biológicos es que BioWeka permite cargar secuencias en formato FASTA directamente (y algunos más de frecuente uso biológico), realizar operaciones que impliquen directamente secuencias, como extracción de características a partir de ellas, y la aplicación de clasificación basada en alineamientos. Para las secuencias de aminoácidos, además de calcular el número de veces o frecuencia de aparición de cada aminoácido en la secuencia, BioWeka permite extraer características de la secuencia, basadas en propiedades de los aminoácidos que la componen. Los valores estables de dichas propiedades para cada aminoácido se encuentran en la “Base de datos de índices de aminoácidos” [33], que contiene más de 500 propiedades, llamadas índices. Cada índice se compone de 20 valores numéricos, uno para cada aminoácido. Dicha base de datos está accesible de forma libre en un fichero de texto simple. Con esta matriz de propiedades por aminoácido, BioWeka calcula y promedia un valor global de dichas propiedades para la secuencia completa. De esta manera, se pueden extraer más de 500 atributos a partir de una secuencia, entre las que se encuentran el índice de hidrofobicidad, volumen de residuo, polaridad, peso molecular, carga positiva y negativa, índice de hélices alfa, punto isoeléctrico, etc. Por su parte, para las secuencias de ADN es más común utilizar conteo de nucleótidos, o frecuencia de codones, que también BioWeka es capaz de calcular automáticamente.

La instalación de BioWeka no es tan sencilla como la de Weka, por no estar tan extendido su uso como el de Weka. Se requiere tener instalada una versión de Weka, y descargar un fichero empaquetado de Java (.jar) de su sitio web<sup>2</sup>. Una vez que se siguen las instrucciones de instalación y configuración de la descarga de BioWeka, se podrían utilizar la siguiente instrucción para convertir un fichero de secuencias en formato FASTA al formato ARFF que usa Weka:

```
$ java bioweka.core.converters.sequence.FastaSequenceLoader -A PROTEIN  
-i secuencias.faa -o secuencias.arff
```

y a partir de ahí contabilizar el número de apariciones de cada aminoácido:

```
$ java bioweka.filters.sequence.extractors.SymbolCounter  
-i secuencias.arff -o secuencias_contadorAA.arff
```

u obtener las más de 500 propiedades que caracterizan la secuencia, a partir de los valores de sus aminoácidos en la base de datos de índices, con:

---

<sup>2</sup>BioWeka. <http://sourceforge.net/projects/bioweka>

```
$ java bioweka.filters.sequence.extractors.SymbolPropertyAnalyzer
-P bioweka.providers.AAindexProvider
-i secuencias.arff -o secuencias_propIndiceAA.arff
```

Cabe destacar que normalmente en un proceso completo de minería de datos, el uso de estas transformaciones de BioWeka debe combinarse con los filtros que proporciona originalmente Weka (ver Sección 5.4), por ejemplo, para borrar los atributos innecesarios de los más de 500 que nos proporciona la última operación, o normalizar alguno de ellos.

Para una revisión completa de todas las capacidades de BioWeka, se recomienda consultar el manual incluido en la descarga del software<sup>2</sup>, dado que aquí sólo se exponen algunos ejemplos de uso destacados.

## 5.2. Retos de la minería de datos en bioinformática

Existe una larga lista de retos para la inteligencia artificial y la minería de datos asociados a los datos y al contexto biológico. Este hecho implica que haya que gestionar con cuidado ciertos aspectos de la IA en bioinformática.

Las principales aspectos a tener en cuenta son:

- Manejo de un *gran número de datos*, que obligan a automatizar absolutamente todos los procesos y se requieren grandes recursos computacionales, en tiempo y memoria, impidiendo a veces incluso la aplicación de ciertos métodos estándar.
- *Ruido intrínseco* en los datos [4]. Las razones de dicho ruido, aparte de un error inicial experimental, pueden ser: un proceso erróneo de carga en las bases de datos públicas, una interpretación incorrecta de los experimentos, una revisión de diferentes elementos por distintos supervisores, o la unión de información cargada en las bases de datos por diversas personas. Este ruido repercute en la inexactitud y falta de completitud de los datos utilizados para el entrenamiento del AA. Tanto en la asignación de clases, que siempre incluye *falsos positivos* y *falsos negativos*, como en los atributos, al utilizar los valores conocidos hasta el momento, que en un futuro pueden modificarse (por ejemplo, corrigiendo errores o añadiendo anotaciones desconocidas). En cada momento, se anota lo que se conoce, lo cual no quiere decir que una proteína sin anotar en un vocabulario, en un futuro no lo esté. Por lo tanto, en un esquema ideal de clasificación, las predicciones positivas estarían repartidas entre los verdaderos y los falsos positivos; mientras que en problemas de clasificación en biología, al estar basados con frecuencia en conocimiento incompleto, las predicciones positivas se dividen con un porcentaje mínimo para verdaderos positivos, otro similar para falsos positivos, y la mayor parte para desconocidos [59]. Así, hay que tener presente que dicho ruido y falta de completitud afecta a la calidad de los resultados obtenidos en un proceso de AA, al depender en gran medida de la calidad de los datos de entrenamiento [29].
- *Multiplicidad*, por ejemplo, múltiples identificadores biológicos diferentes para un mismo elemento, lo cual exige un proceso de mapeo continuo [27], así como una posible fuente adicional de ruido.
- *Interrelación compleja* de todos los elementos biológicos. Se conoce que la mayoría de las anotaciones funcionales (utilizadas frecuentemente como datos de entrada de la predicción) proceden de transferencia por homología [45]. Por lo tanto, se deben analizar siempre cuidadosamente dichas interrelaciones, para no sesgar el proceso de aprendizaje al incluir instancias muy similares para entrenar y para evaluar a la vez. Por ejemplo, muchos genes y proteínas pueden ser miembros de una misma familia por similitud en secuencia o estructura, incluso en diferentes especies.

- *Redundancia* en las bases de datos. Los mismos genes y proteínas pueden estar en distintos repositorios, hasta con distintos valores en sus propiedades. Incluso las secuencias pueden ser diferentes, debido a un proceso de secuenciación distinto o a la transcripción alternativa. Las anotaciones deberían ser las mismas, pero en ocasiones pueden diferir también. Así, determinar el valor más adecuado es un problema añadido.
- *Clasificación multi-clase*. En biología, frecuentemente no se afrontan problemas en los que una clasificación binaria sea suficiente, sino que se necesita realizar una selección entre  $N$  posibles valores o funciones [19].
- *Clasificación multi-etiqueta*. Cada gen o proteína generalmente está involucrado en más de una función, requiriendo que se pueda asignar más de un valor de la clase a cada ejemplo [53]. La multi-funcionalidad es un reto importante y poco afrontado hasta hoy en la anotación funcional [32].
- *Clases des-balanceadas*. En bioinformática la cantidad de ejemplos positivos (con frecuencia, procedentes de fuentes experimentales) siempre es mucho menor que la de ejemplos negativos. En el dominio de anotación funcional, que también es multi-clase, el número de instancias para cada una de las  $N$  posibles clases es, además, muy diferente entre sí. Este des-balanceo de clases en predicción de función puede venir dado porque unas funciones son más comunes (transporte y enlace) en la célula que otras (funciones relacionadas con ácidos grasos y metabolismo de los fosfolípidos) y también porque las anotaciones están sesgadas y limitadas a tipos de proteínas sobre las que se han hecho más estudios y análisis [2].
- *Definición de clases imprecisa*, no fiable y a veces a distintos niveles (por ejemplo: en una jerarquía), o incluso con clase desconocida para algunos ejemplos.
- *Múltiples fuentes* de información a integrar en un solo esquema de representación del conocimiento. Este reto también implica decidir si tomar todos los datos disponibles o un subconjunto que cumpla unos criterios, si tomar los datos pre-procesados u originales, entre otras cuestiones.
- *Valores desconocidos* (del inglés, *missing values*). Pueden ser debidos a la pérdida de datos por un problema particular de manejo de datos [28]. Pero los valores desconocidos más relevantes, o más complicados de gestionar, surgen del hecho de la inexistencia de anotaciones de todos los tipos de información para todos los genes o proteínas, con una carga semántica asociada de carácter biológico, que desaconseja su gestión con los métodos estándar. La representación del conocimiento debe cubrir este tipo de casos.
- *Agrupamientos* en listas o grupos de elementos biológicos. Con frecuencia se utiliza como unidad el concepto de lista o grupo de genes/proteínas, en lugar de considerar un gen o producto genético independiente. Con lo que hay que asociar a cada grupo sus propiedades, intrínsecas y agregadas.
- *Falta de estandarización* de los métodos de predicción de anotación de función. Dificultad en cómo y con qué comparar, y qué medida de evaluación utilizar.

En conclusión, existen múltiples habilidades que las técnicas de minería de datos deben desarrollar y evaluar para la resolución de problemas biológicos.

### 5.3. Recopilación y selección de datos

El volumen de datos disponible en el área de ciencias de la vida (biología molecular, biología medioambiental, medicina personalizada) crece desde hace varios años a un ritmo superior al marcado por la

*ley de Moore* [1] (ver Subsección 1.2.2 y Figura 6.1). Ello es debido a la creación y abaratamiento de técnicas experimentales de altas prestaciones, como por ejemplo las técnicas de ultra-secuenciación [39].

Antes de la llegada de estas técnicas experimentales de altas prestaciones, un científico del área de ciencias de la vida formulaba hipótesis y sugería conjeturas sobre los mecanismos que regulan los procesos biológicos a distintos niveles, pero no le era posible disponer de suficiente información experimental que permitiera rebatir o corroborar dichas afirmaciones. Los datos experimentales disponibles eran tan escasos se podían recopilar, inspeccionar y seleccionar manualmente.

Actualmente, debido al volumen de información experimental y derivada disponible, es imposible realizar todas las tareas anteriormente descritas de forma manual. Por ello, la etapa de recopilación de datos es aún más crucial. La determinación de las fuentes de datos a usar, qué bases de datos biológicas a consultar, qué versiones de los datos emplear y la coherencia y completitud de todos los datos recopilados son factores a tener en cuenta.

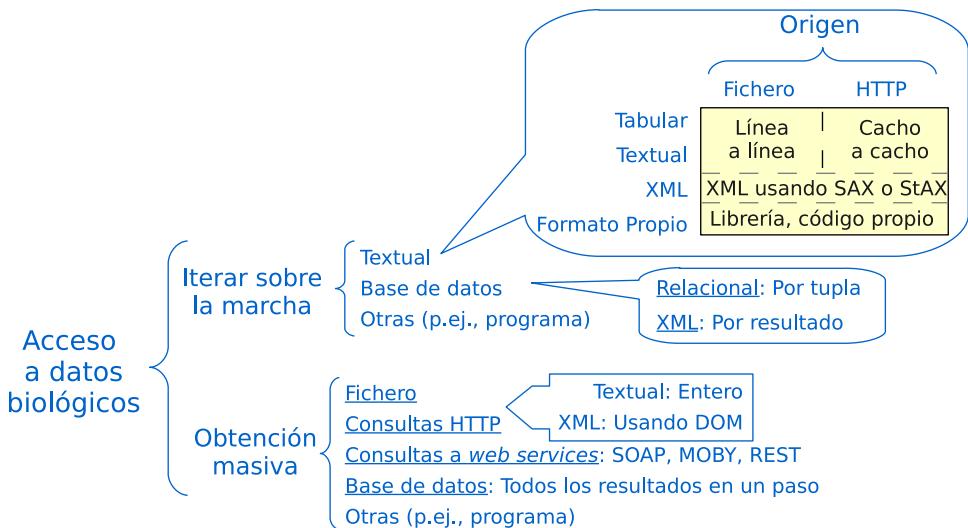
### 5.3.1. Repositorios de datos biológicos.

Los conceptos usados en las áreas de biología, biología molecular, bioinformática y muchas otras van adaptándose y remodelándose a medida que surgen nuevas evidencias que los corroboren, refinen o rebaten. Ello lleva a que el conocimiento recopilado vaya siendo moldeado con premisas cambiantes. Éste y otros factores han conducido a que se hayan y se estén creando multitud de representaciones, formatos de fichero y de bases de datos [17]. La consecuencia directa es que muchas de las propias representaciones, formatos de fichero y bases de datos van transformándose a medida que cambian los conceptos biológicos a ser representados, además de dejar de usarse algunas y de ser creadas otras nuevas [20].

Un *repositorio de datos* es un almacén de información cuyos contenidos comparten ciertas características semánticas, pero ello no asegura que esté diseñado para ser consultado y usado programáticamente. Existen repositorios de datos pensados para ser usados por especialistas, como por ejemplo el compendio de genes humanos y fenotipos genéticos OMIM [3], o PubMed, la base de datos de citas y resúmenes bibliográficos de literatura biomédica. Ambas albergan mucho conocimiento recopilado, pero buena parte del mismo no puede ser extraído de forma automática, al estar únicamente expresado en lenguaje natural. Un caso de repositorio de datos pensado tanto para las personas como para minería de datos es la base de datos UniProt [52], que está especializada en el almacenamiento de secuencias de proteínas, junto con sus anotaciones funcionales y su relación con contenidos de otras bases de datos biológicas. Casi todas esas anotaciones son sistemáticas y mantenidas, ya que provienen de ontologías de anotaciones biológicas usadas en este tipo de repositorios de datos. Una ontología de anotaciones que sirve de nexo de unión de muchos repositorios de datos es *Gene Ontology* [51], y como muchas ontologías biológicas, es de por sí un repositorio de datos.

Los repositorios de datos biológicos suelen contener tanto *información primaria* (p.ej. obtenida experimentalmente) como *información derivada* (basada en datos ya existentes). Ejemplos de repositorios primarios serían ENA [35] (European Nucleotide Archive), EGA (European Genome-phenome Archive), GEO [5] (Gene Expression Omnibus) o wwPDB [55] (Worldwide Protein Data Bank). La calidad de un repositorio de datos se mide por la calidad de la información que contiene. Esta calidad viene determinada por varios factores, como la coherencia interna de la información recopilada, la conectividad de ese conocimiento con otras entradas de datos en otras bases de datos, o la frecuencia con la que el conocimiento derivado que se encuentre almacenado en el repositorio se mantiene al día.

Para una más amplia descripción sobre la estructura, funcionamiento y principales bases de datos biológicas leer el Capítulo 1.



**Figura 5.3:** Estrategias de obtención y procesamiento de los datos biológicos.

Tal como se explica en la Figura 5.3, a la hora de planificar una sesión de minería de datos y recopilar la información necesaria, una vez determinados qué repositorios de datos a usar, y cómo vamos a interrelacionarlos, hay que plantear la manera más efectiva para recuperar y extraer los registros de conocimiento a integrar. Todos los repositorios de datos disponen de sus propias *interfaces web* enfocadas al usuario final, que proporcionan *herramientas de búsqueda* entre las entradas de ese repositorio, e intentan mostrar de forma gráfica el conocimiento asociado a las entradas a visualizar. Como estas interfaces web no siempre suelen ser las más apropiadas para la recuperación de muchas entradas de datos, muchos de estos repositorios de datos proporcionan tanto *interfaces programáticas* de algún tipo, como el volcado de toda la información que contienen en *sitios de descarga HTTP o FTP*.

Si parte de los repositorios que necesitemos usar en nuestra sesión de minería de datos proveen algún tipo de *interfaz programática de consulta online* (o como en el caso de Ensembl, una *librería de programación*), hay que evaluar esas capacidades de consulta remota. Los “*pros*” de las consultas remotas son: para recopilación de información puntual o dispersa es lo más óptimo, ya que no es necesario descargar en local los repositorios de datos ni instalar ningún sistema gestor de bases de datos para realizar consultas; el procesamiento de la consulta se hace en los servidores que proporcionan esa interfaz, disminuyendo los requisitos *hardware* locales a la hora de realizar la minería de datos. Los “*contras*” de las consultas remotas son: dificultan la reproducibilidad de los análisis, al poder cambiar sin previo aviso los datos que haya por detrás de esas interfaces; son bastante lentas cuando hay que realizar un gran número de consultas, e inefficientes cuando hay que recuperar un gran volumen de resultados; son más propensas a fallos, al depender de una conexión de internet. Si al final no se usa ningún tipo de interfaz de consulta remoto, entonces hay que recurrir a la descarga parcial o total del repositorio de datos.

Una vez descargados el repositorio o los registros de datos, comienza la etapa de *filtrado y extracción* de la información. Las herramientas usadas en estos casos suelen ser, entre otras: la utilización de comandos estándar de UNIX ('grep', 'awk', 'sed'...), bastante común a la hora de tratar con contenidos tabulares; el uso de programas especializados o programas *ad-hoc*, escritos para tal propósito; o la carga de los datos en alguna instancia de bases de datos, para aprovechar su motor de consulta a la hora de hacer minería de datos.

El *formato de los registros de datos* y qué queramos hacer sobre ellos influirá en las estrategias que

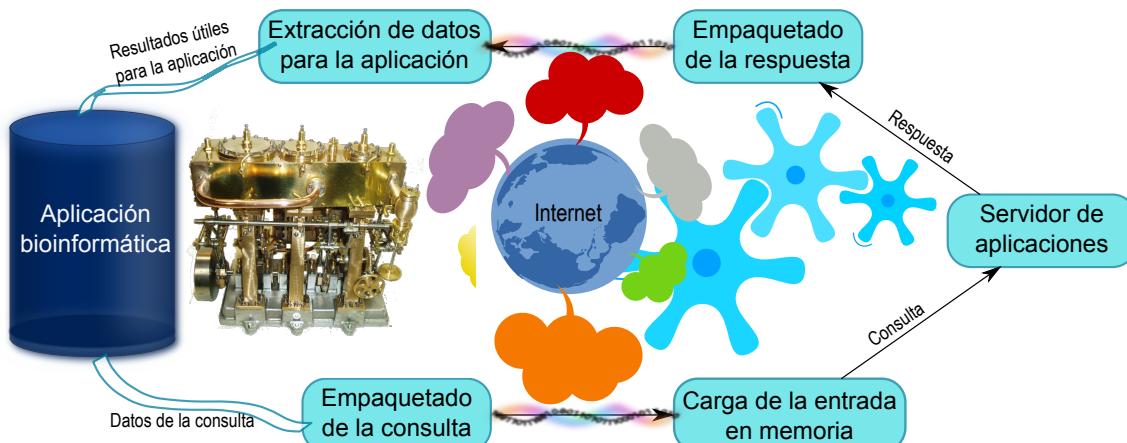
sigamos. A nivel de representación física, los registros de datos de un repositorio pueden venir en formatos muy dispares:

- Formato *tabular*: por ejemplo, los formatos BED, SAM, VCF, etc. usados en el área de ultrasecuenciación; el formato MITAB 2.5 para la representación de interacciones entre proteínas; volcados de tablas de bases de datos relacionales, como en el caso de Ensembl o antiguamente InterPro; etc...
- Formato *textual* propio: las bases de datos, principalmente las más antiguas, usan sus propios formatos textuales, como en el caso de UniProt, PDB, RefSeq, ....
- Formato *jerárquico*, tipo XML o ASN.1: Un caso de uso de XML es en casi todas las bases de datos de interacciones, ya que siguen el estándar de representación PSI-MI 2.5 a la hora de describir de forma detallada las interacciones entre proteínas y todo lo que se sabe de ellas. Otro caso es el de GenBank y otras bases de datos almacenadas en NCBI, que están disponibles en formato ASN.1, además de en su formato textual propio desde hace más de 20 años.
- *Resource Description Framework* (RDF): Poco a poco va siendo común que algunos repositorios de datos también se distribuyan en las distintas serializaciones de la representación en RDF, que está más orientado hacia el contenido semántico que hacia su representación sintáctica. Es el caso de la base de datos de rutas metabólicas Reactome.

### 5.3.2. Servicios web

Un *servicio web* es un programa, base de datos o recurso computacional que encapsula una o varias funcionalidades, y que puede ser usado de forma remota. El paradigma de los servicios web se lleva usando desde que existe Internet, y está ampliamente respaldado por la industria y la comunidad científica. De los servicios web existentes, muchos fueron creados para ser usados por interfaces gráficas, o de forma interactiva por los humanos, como por ejemplo Google Maps, Dropbox o Flickr. Pero también hay un gran número de servicios web enfocados a uso programático.

Los servicios web en los que estamos interesados a nivel bioinformático son aquellos que nos permitan hacer recuperación y minería de datos en una base de datos biológica existente (por ejemplo, los de Ensembl) o que implementen algún algoritmo bioinformático (por ejemplo, búsquedas de motivos en secuencias mediante HMMs).



**Figura 5.4:** Esquema de llamada de un servicio web.

En la Figura 5.4 vemos cómo funciona a nivel conceptual la llamada a un servicio web. *Llamar a un servicio web* es como llamar a un método local en cualquier lenguaje de programación, pero con algunos pasos intermedios. Por eso, una aplicación que necesite usar uno o varios servicios web tendrá que saber a priori qué hacen los servicios web que va a llamar, qué datos de entrada espera el servicio y qué tipo de respuestas va a proporcionar. Pero además, tendrá que saber dónde se encuentra el servicio, cómo espera el servicio web que le lleguen físicamente las consultas y cómo están formateadas las respuestas.

Basándonos en la forma de comunicación con el servicio web, existen diversos *estándares de llamada de servicios*, como REST, SOAP o XMLRPC, para los cuales hay librerías de programación para casi todos los lenguajes. Hay incluso algunos estándares específicos de la bioinformática, como por ejemplo DAS, BioMart, SoapLab o BioMOBY, que usan REST o SOAP como capa de transporte. Estos estándares son los que establecen las reglas de cómo se estructuran los mensajes usados para comunicarse con los servicios (el idioma) y los protocolos de comunicaciones (las formas). A su vez, estos mensajes pueden encapsular casi cualquier representación de los datos de consulta y respuesta, así que muchos de los formatos bioinformáticos existentes son usados en servicios web bioinformáticos. Ver en el Código 5.1 un ejemplo de llamada a un servicio web SOAP desde Perl.

Código 5.1: Llamada desde Perl a un servicio SOAP de iHOP.

```
1 #!/usr/bin/perl -W
2 use strict;
3 use SOAP::Lite;
4 use XML::LibXML;
5 my $uri    = 'http://www.pdg.cnb.uam.es/UniPub/iHOP/xml';
6 my $proxy  = 'http://ubio.bioinfo.cnio.es/biotools/iHOP/cgi-bin/iHOPSOAP';
7 my $soap   = new SOAP::Lite(uri    => $uri, proxy => $proxy);
8 $soap->outputxml('true');
9 my($unparsedsom)= $soap->call('getSymbolInteractionsFromIHOP','68679');
10 # El siguiente paso es necesario para evitar problemas con código XML dentro del mensaje
     SOAP
11 $unparsedsom =~ s/&#([0-9]+);/&#$1;/g;
12 my($parser)=XML::LibXML->new();
13 print STDERR "antes\n";
14 my($doc)=$parser->parse_string($unparsedsom);
15 print STDERR "después\n";
16 my($result)=($doc->getElementsByLocalName('result'))[0];
17 print $result->toString();
```

Aunque los servicios web tienen grandes ventajas, porque delegan parte del trabajo (a veces muy pesado) en otros sistemas e instalaciones de cálculo científico, y permiten consultar grandes volúmenes de conocimiento científico sin tener que descargarlo en local, a su vez tienen varios puntos débiles. El primero de ellos es la *disponibilidad*, ya que en caso de que haya problemas de red, o que el servidor que hospeda al servicio esté caído, afecta a todos los programas que usen ese servicio. El segundo es el *rendimiento*, ya que todo el empaquetado y desempaquetado de mensajes introduce una sobrecarga de tiempo de computación y memoria a la hora de realizar muchas llamadas, y el tiempo de ejecución depende de cómo de cargado esté el servicio (al ser un recurso compartido por toda la comunidad científica).

Una limitación de los servicios web es saber encontrar dichos servicios, saber qué hacen y saber cómo esperan ser llamados y qué van a responder, todo ello, a ser posible, de forma programática. Para mitigar este problema, e intentar resolverlo en parte, existen *repositorios de servicios web*, donde se pueden encontrar listados de dichos servicios. En estos repositorios los desarrolladores publican las especificaciones de uso de sus servicios. Por ejemplo, instituciones como EBI<sup>3</sup> y NCBI<sup>4</sup> documentan de

<sup>3</sup>EMBL-EBI Web Services. <http://www.ebi.ac.uk/Tools/webservices>

<sup>4</sup>Entrez Programming Utilities Help. <http://www.ncbi.nlm.nih.gov/books/NBK25501>

forma sistematizada su muy extensa biblioteca de servicios web en sus páginas web.

Algo muy importante a la hora de trabajar con servicios web es poder saber si los servicios en los que estamos interesados siguen estando operativos, funcionando y al día. Para que cualquier desarrollador pudiera registrar servicios web genéricos relacionados con las ciencias de la vida, y estuvieran en un lugar centralizado donde pudieran ser periódicamente comprobados, se creó BioCatalogue [6], que a finales de 2013 tenía registrados más de 2300 servicios.

Finalmente, cuando la tarea de recopilación de datos requiere del uso de varios servicios web, lo más común es construir un *workflow* de recuperación y procesamiento inicial de la información. Un *workflow* es el equivalente a una receta o a un procedimiento de laboratorio, en el que se describen los pasos que hay que dar para realizar una tarea de integración de datos. Un *workflow* puede realizar una tarea muy especializada y puntual (lo cuál suele ser bastante común) o bien una tarea más genérica. Físicamente, los pasos que vaya a dar un *workflow* pueden estar plasmados en un programa escrito en cualquier lenguaje de programación, o bien ser descritos de una forma más abstracta y reutilizable en algún lenguaje de representación de *workflows*. Por ello, también existe un repositorio de *workflows* bioinformáticos, llamado myExperiment [22], y que a finales de 2013 albergaba alrededor de 2500 *workflows*.

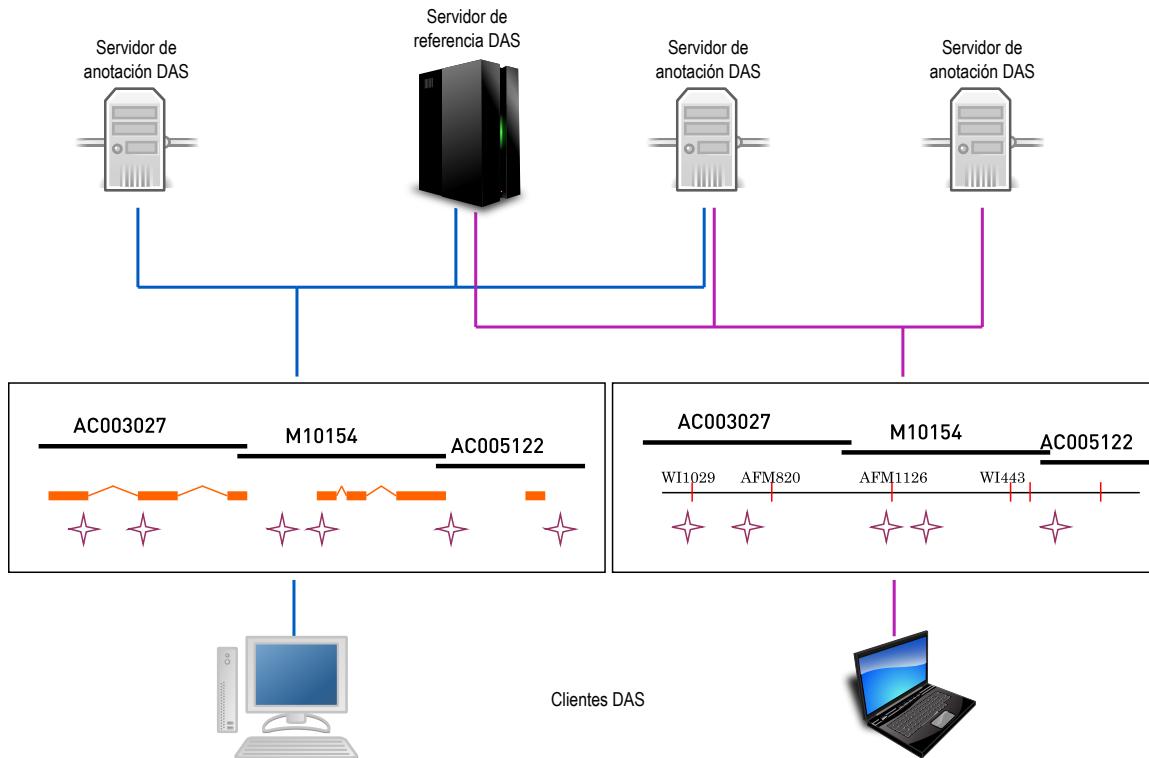
## Distributed Annotation System

El sistema y protocolo de anotaciones *Distributed Annotation System* (DAS) [9] está basado en un modelo cliente-servidor de intercambio de mensajes XML, que siguen el formato DASXML. Fue inicialmente creado para coordinar el trabajo de anotación en los distintos proyectos de secuenciación, y el sistema se diseñó para que fuera independiente de los sistemas y esquemas de bases de datos usados para almacenar los genomas de referencia y sus anotaciones.

Las entradas (normalmente secuencias) almacenadas en el servidor de referencia usan un sistema de coordenadas jerárquico (cromosomas, scaffolds, contigs, genes, ...). Las anotaciones de cada entrada usan coordenadas relativas a la entrada, de forma que si, por ejemplo, por un reensamblaje, se mueve la secuencia con respecto a su ancestro, no queden invalidadas las anotaciones. Además permite separar los roles del genoma de referencia y de las anotaciones, de manera que varios grupos de expertos pueden simultáneamente anotar una misma versión de un genoma de referencia, usando para ello diferentes servidores de anotaciones que hablan el mismo protocolo (Figura 5.5). De esta manera, es posible obtener una superposición de las anotaciones, y verlas conjuntamente con clientes que entiendan el protocolo DAS.

La especificación inicial del protocolo DAS fue escrita por Lincoln Stein y Robin Dowell. La especificaciones DAS más usadas son la 1.53E y la 1.6, que contienen muchas mejoras y extensiones sobre la versión 1.0, manteniendo siempre que es posible la compatibilidad hacia atrás. Aunque el protocolo DAS fue diseñado para la anotación de genomas, pronto fue llevado este paradigma a otros tipos de conceptos biológicos, como por ejemplo las proteínas, las estructuras tridimensionales o las interacciones. Como ejemplo de la proliferación en el uso del protocolo DAS, a finales de 2013 había registrados casi 1600 fuentes de datos en DAS Registry [44].

Bases de datos como UniProt, UCSC o Ensembl proporcionan sus propios servidores de referencia y de anotación DAS. Una de las claves de la popularidad de los servidores DAS es que se pueden consultar a mano, sin tener que escribir una sola línea de programa, y construir consultas de recuperación muy potentes. Esto se hace escribiendo las URIs siguiendo la convención establecida en las especificaciones de DAS. A continuación unos ejemplos de URLs de *consultas DAS* a servidores de Ensembl y UniProt:



**Figura 5.5:** Esquema de consulta distribuida en el sistema DAS, donde distintos clientes pueden combinar distintos servidores de anotaciones con el mismo servidor de referencia.

- Fragmento de secuencia del cromosoma 33 de *Canis familiaris* desde la posición 3000001 a la posición 3100000:
 

```
http://www.ensembl.org/das/Canis_familiaris.BROAD2.reference/sequence?segment=33:3000001,3100000
```
- Subcomponentes de secuencia identificados en el mismo sitio del genoma de *Canis familiaris*:
 

```
http://www.ensembl.org/das/Canis_familiaris.BROAD2.reference/features?segment=33:3000001,3100000
```
- Secuencia de la proteína de reparación de ADN RAD51 de *Homo sapiens*:
 

```
http://www.ebi.ac.uk/das-srv/uniprot/das/uniprot/sequence?segment=RAD51_HUMAN
```
- Todas las anotaciones existentes en UniProt para la proteína RAD51 de humanos:
 

```
http://www.ebi.ac.uk/das-srv/uniprot/das/uniprot/features?segment=RAD51_HUMAN
```

## BioMart

*BioMart* [61] es un sistema de bases de datos biológicas federadas, donde las bases de datos pueden estar distribuidas físicamente en distintos países y continentes y seguir siendo usadas en una consulta conjunta. El sistema puede usar cualquier base de datos que esté preparada para el sistema BioMart, o que entienda el protocolo de consultas de BioMart (basado en SOAP y REST). Ensembl, UniProt o proyectos como ICGC [60] (International Cancer Genome Consortium) usan en sus sitios web la tecnología de BioMart para almacenar, publicar o permitir consultar todo el conocimiento que gestionan.

Dada la popularidad de esta tecnología, se puede acceder a las bases de datos BioMart casi desde cualquier lenguaje de programación usando su API SOAP, y desde muchos sistemas de análisis de datos (p.ej. *plugin* de BioMart para Cytoscape, paquete ‘biomaRt’ de R).

The figure consists of three main panels. The top panel is a 'Gene retrieval' search interface for the Ensembl dataset. It has two main sections: '1. SELECT DATASETS' on the left listing various species' genes, and '2. RESTRICT SEARCH' on the right where specific parameters like chromosome, gene start/end, and gene biotype can be set. The middle panel shows the search results for chromosome 14, displaying gene names like CGNK and CCDC8C along with their biotypes and status. The bottom panel is a modal window titled 'REST / API Query' which contains the XML code for the search performed.

**Figura 5.6:** Ejemplo de consulta gráfica en BioMart 0.8, su resultado, y la consulta en XML, lista para usarla desde programa.

Actualmente existen dos versiones de BioMart. La versión estable (pero que ya no se mantiene) es la 0.7, está escrita en Perl y es ampliamente usada en Ensembl, UniProt y otros sistemas. Por otro lado, la versión 0.8 está re-escrita por completo en Java, y ha sido usada hasta la release 13 de datos de ICGC. Los servidores BioMart proporcionan una interfaz web que permite preparar de forma sencilla consultas relativamente complejas, y a su vez reflejar esas consultas en un formato compatible con las APIs programáticas de BioMart, como se puede ver en la Figura 5.6. Además, cada versión de BioMart es capaz de proporcionar para las consultas realizadas un trozo de código escrito en el lenguaje de programación usado para esa versión de BioMart, que permite realizar esa misma consulta de forma programática.

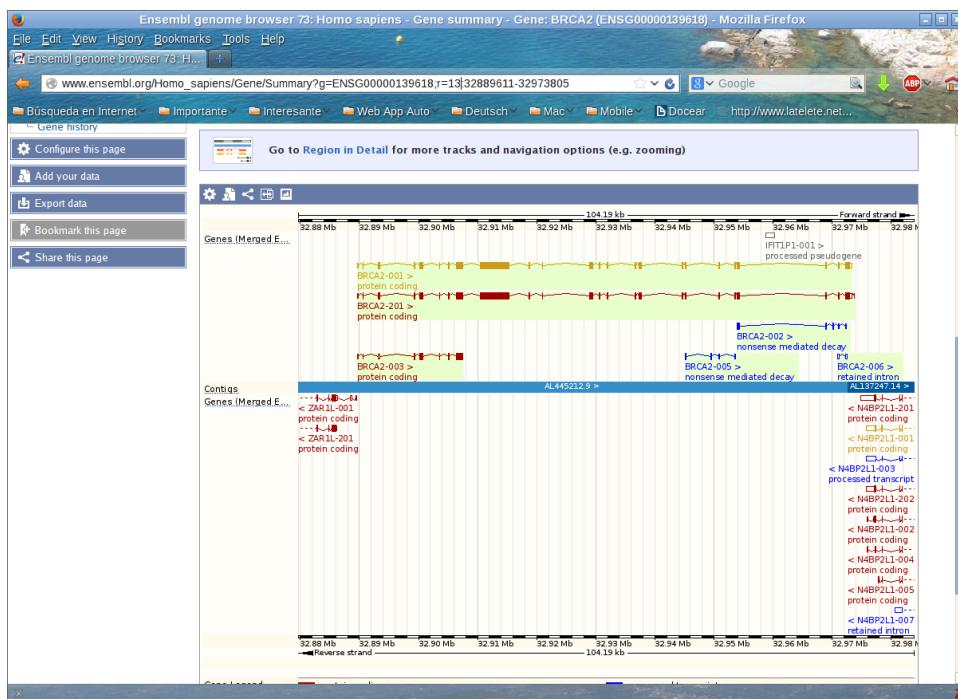
## Ensembl

El proyecto Ensembl [14] nació en 1999 con el objetivo de *anotar el genoma humano*, antes incluso de que estuviera disponible el primer *draft*, siguiendo un proceso lo más automatizado posible, relacionando dichas anotaciones con otros datos biológicos disponibles. Desde el primer momento se planteó que su interfaz de acceso principal fuera vía web, para que tuvieran acceso el mayor número posible de científicos. Desde entonces, además del genoma humano, se han ido añadiendo genomas de otros organismos de cordados a medida que sus respectivos proyectos de secuenciación se han ido completando. A finales de 2013 había integrados en Ensembl *datos de cordados de casi 80 especies* diferentes. Además

de contener *datos genómicos* como tal, Ensembl también provee actualmente datos de *genómica comparativa* entre los distintos organismos registrados, datos de *variabilidad* de 17 especies y datos de *regulación génica* provenientes de proyectos como ENCODE [7].

Cada *release* de Ensembl toma como base la última versión estable del ensamblaje de cada genoma de referencia que publica (GRCh37, GRCm38, etc...), y a partir de ahí empieza a mapear dentro de cada genoma de referencia cada uno de los genes, tránscritos, exones y proteínas que han sido identificados por el sistema de anotación de Ensembl, así como los datos biológicos y anotaciones relacionados publicados y las interrelaciones y consistencia de estos elementos. Para mantener la coherencia siempre que sea posible entre los datos de las distintas versiones de Ensembl, se mantiene un histórico de los *identificadores de genes, tránscritos, exones y proteínas*, para saber si ese elemento etiquetado ha sufrido cambios, ha sido discontinuado, etc...

Existen varias alternativas a la hora de consultar esta ingente cantidad de información: interfaz web, BioMart, DAS, API programática o incluso descargar una copia del software y datos necesarios para hacer una instalación local. La interfaz web de Ensembl es bastante versátil, ya que permite realizar consultas basadas tanto en coordenadas genómicas como en identificadores de genes, tránscritos, etc... de Ensembl o de otras bases de datos biológicas que hayan sido mapeadas. Entre las muchas vistas de datos que proporciona está la *vista de navegación genómica* (Figura 5.7), que muestra todos los genes, tránscritos y otros elementos anotados en esa zona. Ésta y todas las demás vistas son ampliamente configurables, ya que permiten mostrar y ocultar pistas completas de información: por ejemplo, *datos de variantes de splicing* o *anotaciones de otros repositorios* de datos para esa misma zona o los genes de esa zona. Cada vista permite *exportar los datos* representados en ella a distintos formatos, como por ejemplo formatos tabulares, formatos de anotaciones, FASTA (para los fragmentos de secuencia mostrados en la vista) y otros formatos adicionales.



**Figura 5.7:** Parte de la vista gráfica correspondiente al gen BRCA2 en Ensembl.

La *API programática de Ensembl* sólo está disponible como librerías para el lenguaje de programación

Perl. Esas librerías van evolucionando con cada nueva versión de Ensembl. Aunque es posible usar versiones antiguas de las librerías de Ensembl para consultar una versión actual, no se garantiza que cada una de las funciones opere correctamente, ya que su funcionamiento depende de la estructura de la base de datos Ensembl. Como se puede ver en el ejemplo representado en el Código 5.2 (extraído del tutorial *online* de Ensembl), el primer paso antes de consultar una base de datos Ensembl es conectarse a su registro, para poder saber qué organismos y qué tipo de información hay disponible. Tras eso, el siguiente paso dado en el ejemplo es obtener un adaptador de genes, para poder buscar genes a partir de identificadores. Una vez encontrado el gen, se puede obtener tanto su correspondiente identificador de Ensembl, como los tránscritos asociados a ese gen. Otra forma de buscar información, tal como se muestra en el ejemplo, es mediante un adaptador de slices, con el cuál se puede recuperar todo lo disponible en un rango, que puede ser definido tanto directamente mediante coordenadas cromosómicas como de forma simbólica (por identificador, por cromosoma, etc...).

Código 5.2: Ejemplos de varias llamadas programáticas a Ensembl.

```

1 #!/usr/bin/perl -W
2 use strict;
3 use Bio::EnsEMBL::Registry;
4 my $registry = 'Bio::EnsEMBL::Registry';
5 $registry->load_registry_from_db(
6     -host => 'ensembldb.ensembl.org', # alternatively 'useastdb.ensembl.org'
7     -user => 'anonymous'
8 );
9 # get a gene adaptor for the human core database
10 my $gene_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Gene' );
11 # Get the 'COG6' gene from human
12 my $gene = $gene_adaptor->fetch_by_display_label('COG6');
13 print "GENE ", $gene->stable_id(), "\n";
14 print_DBEntries( $gene->get_all_DBEntries() );
15 foreach my $transcript ( @{$gene->get_all_Transcripts()} ) {
16     print "TRANSCRIPT ", $transcript->stable_id(), "\n";
17     print_DBEntries( $transcript->get_all_DBEntries() );
18     # Watch out: pseudogenes have no translation
19     if ( defined $transcript->translation() ) {
20         my $translation = $transcript->translation();
21         print "TRANSLATION ", $translation->stable_id(), "\n";
22         print_DBEntries( $translation->get_all_DBEntries() );
23     }
24 }
25 # get a slice adaptor for the human core database
26 my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
27 # Obtain a slice covering the entire chromosome X
28 my $slice = $slice_adaptor->fetch_by_region( 'chromosome', 'X' );
29 # Obtain a slice covering the entire clone AL359765.6
30 $slice = $slice_adaptor->fetch_by_region( 'clone', 'AL359765.6' );
31 # Obtain a slice covering an entire NT contig
32 $slice = $slice_adaptor->fetch_by_region( 'supercontig', 'NT_011333' );
33 # Obtain a slice covering the region from 1MB to 2MB (inclusively) of
34 # chromosome 20
35 $slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 1e6, 2e6 );

```

### 5.3.3. Librerías de programación

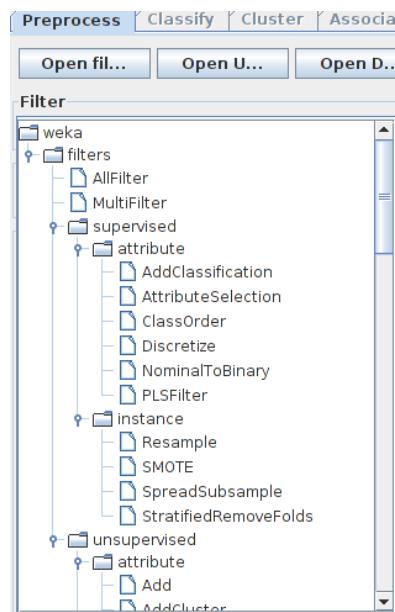
Debido a la gran cantidad de programas, servicios web y formatos de representación existentes en el área de la bioinformática, existen para los distintos lenguajes de programación librerías que facilitan la tarea de extraer información, convertir entre formatos y consultar los repositorios de datos *online*

más comunes. Algunas de estas librerías son ‘BioPerl’, ‘BioRuby’, ‘BioPython’ y ‘BioJava’ para los lenguajes de programación Perl, Ruby, Python y Java, respectivamente. Aunque todas estas librerías sean nombradas de una manera similar, no todas tienen las mismas funcionalidades ni son igual de maduras. Ejemplos de uso de BioPerl se encuentran disponibles en el Capítulo 4, Subsección 4.4.4.

## 5.4. Preprocesamiento y transformación de datos

Una vez recopilados y seleccionados los datos, la siguiente fase del proceso de minería de datos implica modificar la forma o representación de la información asociada a cada instancia. Este paso es imprescindible para que los algoritmos de AA puedan extraer conocimiento de los datos en las condiciones adecuadas, sin ver afectado su resultado por ‘artefactos’ que puedan impedir o limitar su capacidad de aprendizaje, mejorando la eficiencia del algoritmo y la calidad de los resultados.

Se pueden realizar decenas de preprocesamientos distintos sobre los datos, como se puede observar al desplegar los múltiples filtros que proporciona la herramienta Weka para estos fines (ver Figura 5.8). Entre las *operaciones de preprocesamiento*, según la clasificación de los filtros de la herramienta Weka, unas dependen del atributo clase y otras no, y también unas afectan a las instancias y otras a los atributos.



**Figura 5.8:** Fragmento de los filtros disponibles en Weka.

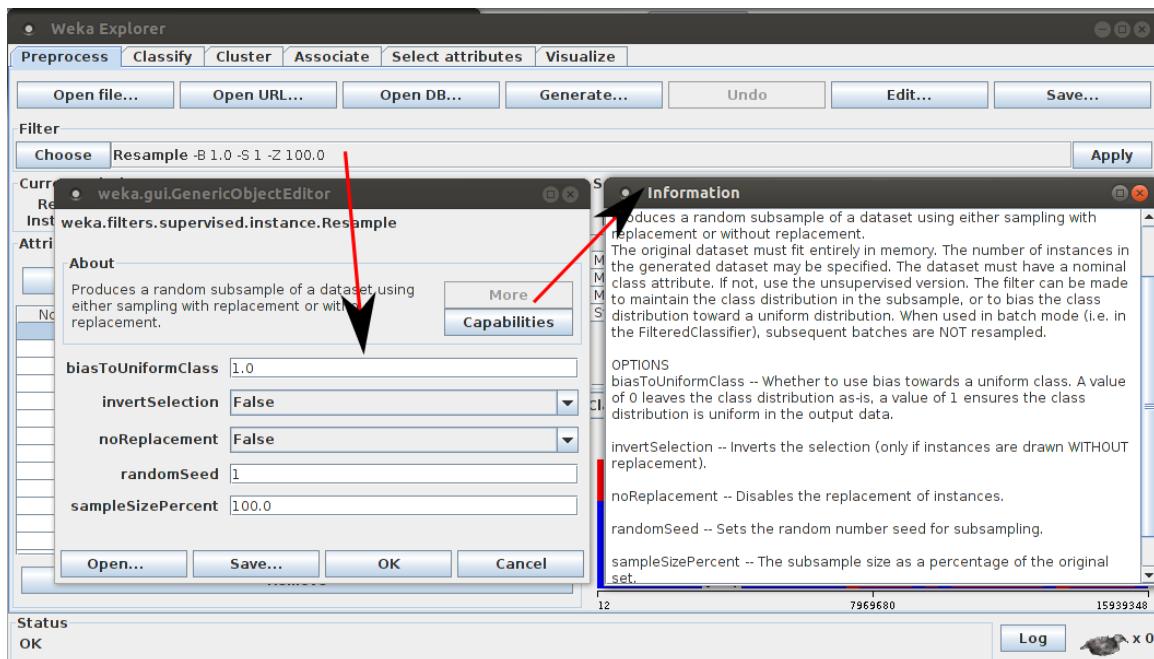
Pero hay un subconjunto de operaciones que incluyen las más frecuentes en los procesos de minería de datos, entre las que seleccionar las imprescindibles en cada caso, para poder llevar a cabo un proceso de AA de forma aceptable. En conclusión, no hace falta ser un experto en minería de datos para conseguir realizar las transformaciones mínimas en los datos. Dicho conjunto de preprocesamientos y transformaciones más comunes son las que se presentan a continuación, detallando su objetivo y la forma de aplicarlas en la práctica con la herramienta Weka. No obstante, también existe la posibilidad de realizar estas operaciones de forma manual, o con cálculos sobre la base de datos donde se tengan los datos almacenados, a elección del usuario.

### 5.4.1. Balanceo de clases

En un problema de clasificación, se debe comprobar cuál es la proporción de instancias de las diferentes clases, para evitar que esté muy desviada hacia una clase concreta. Si la mayoría de las instancias fueran de la misma clase, el sistema de AA sólo aprendería la clase mayoritaria, y el predictor resultante siempre clasificaría cualquier ejemplo como perteneciente a esa misma clase. Esta ausencia de poder de clasificación tiene una justificación lógica, pues clasificando sólo las instancias de la clase mayoritaria, las probabilidades de acertar son mucho mayores que si se obtienen patrones de clasificación de todas las clases minoritarias.

La forma de conseguir el objetivo de balanceo de clases es eliminar instancias de las clases mayoritarias, o duplicar las de las minoritarias. Dependiendo de las necesidades del problemas, se puede elegir entre obtener una distribución completamente uniforme de las clases, o simplemente conseguir unas proporciones cercanas, pero manteniendo ligeramente las diferencias originales.

Para realizar este balanceo con Weka (barra ‘Filter’, botón ‘Choose’), por ejemplo, en el caso de buscar una distribución uniforme entre todas las clases, se debe aplicar el filtro (‘weka/filters’) supervisado (‘supervised’) para instancias (‘instance’) denominado ‘Resample’, con los parámetros que se indican en la Figura 5.9, según la explicación de cada uno aportada por la interfaz gráfica de Weka. El filtro en



**Figura 5.9:** Transformación del conjunto de datos en Weka, para conseguir una distribución uniforme de todas las clases.

formato gráfico de la Figura 5.9 es el mismo que se ejecutaría con el siguiente comando en un terminal:

```
$ java -Xmx1024M -classpath weka.jar weka.filters.supervised.instance.Resample
-B 1.0 -S 1 -Z 100.0 -i entrada.arff -o salida.arff
```

Cambiando estos parámetros o utilizando otros filtros de los paquetes ‘weka.filters.supervised.instance’ y ‘weka.filters.unsupervised.instance’, se pueden obtener otros conjuntos balanceados diferentes.

Por otro lado, además de evitar una distribución descompensada entre las diferentes clases en el conjunto de datos principal, también es importante tener presente que la misma distribución entre clases se

mantenga tanto en el/los conjunto/s de entrenamiento como en el/los conjunto/s de prueba utilizado/s, para evitar sesgos en la evaluación de los resultados.

#### 5.4.2. Gestión de valores desconocidos

Muy frecuentemente es difícil tener rellenas todas las celdas de la tabla “instancias x atributos”. Es decir, que la mayoría de las veces no se conoce el valor de todas las propiedades para todas las muestras, especialmente al trabajar con datos biológicos. Las razones de ello pueden ser variadas, entre las que se encuentran huecos vacíos en las fuentes originales de datos, errores en el proceso de recogida de datos, inexistencia del valor en el momento de la recogida por falta de experimentación o error, imposibilidad de cálculo, ruido en procesamientos previos, etc.

Por ejemplo, pongamos el caso de un problema en el que cada instancia sea un gen y los atributos sean la longitud de la secuencia de nucleótidos, la posición de inicio de la transcripción de la isoforma principal, la cantidad de interacciones físicas de la proteína correspondiente, y la función molecular según el vocabulario FunCat. La longitud de la secuencia es un atributo que tendrá valor para todas las instancias. La posición de inicio de la transcripción no es conocida para todos los genes. Sobre las interacciones, si se han realizado experimentos de interacción a pequeña o gran escala, se podrá tener un valor (aunque no sea definitivo). Por último, la función molecular puede ser desconocida por no estar el gen anotado en la base de datos FunCat, aunque se conozca su función por estar en otros catálogos funcionales. De forma que de los cuatro atributos propuestos en este ejemplo sencillo, sólo se puede asegurar la existencia del valor de un atributo para todas las instancias.

Aunque existen excepciones, muchos algoritmos de AA no se pueden aplicar en presencia de valores desconocidos. Por lo tanto, generalmente es necesario un paso de preprocesamiento que los rellene. Las formas tradicionales de llenar dichos valores desconocidos consiste en tomar la distribución de valores de todas las instancias con un valor existente para ese atributo (es decir, la columna completa) y calcular la media para atributos numéricos continuos, o la moda para atributos numéricos discretos o nominales.

La forma de realizar este preprocesamiento con Weka (barra ‘Filter’, botón ‘Choose’) es con el filtro (‘weka/filters’) no supervisado (‘weka/filters/unsupervised’) de atributos (‘weka/filters/unsupervised/attribute’) denominado ‘ReplaceMissingValue’ con los parámetros por defecto que presenta la interfaz gráfica de Weka. La llamada por línea de comandos sería la siguiente:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.ReplaceMissingValue -i entrada.arff  
-o salida.arff
```

También es frecuente utilizar valores extremos para llenar los valores desconocidos, como el mínimo o el máximo de la distribución, o incluso con un valor fuera del rango, marcando de esta forma que se trata de valores fuera de la distribución. La forma de realizar este preprocesamiento con Weka, para llenar los valores desconocidos del atributo ‘2’ con la etiqueta ‘-10’, es con el filtro ‘AddValues’ con los parámetros que se indican en el siguiente comando:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.AddValues -C 2 -L -10  
-i entrada.arff -o salida.arff
```

No obstante, a veces se pierde la semántica llenando valores desconocidos siguiendo los criterios típicos anteriormente expuestos, y trabajando en problemas biológicos dicha semántica puede ser muy importante. Por ejemplo, no tiene sentido llenar la función molecular de un gen con la moda de todos los demás genes de la muestra, porque no sigue ningún criterio lógico desde el punto de vista

biológico. Si hay una diferencia relevante entre los valores existentes y los no existentes, se puede optar por sustituir los valores desconocidos por una etiqueta que así lo indique, y sea un valor más a utilizar en el proceso de aprendizaje. O esforzarse por intentar ajustarse a las condiciones de un algoritmo que sí permita la existencia de valores desconocidos, asegurándose que no los rellena internamente con alguno de los criterios típicos explicados anteriormente.

En conclusión, la gestión de los valores desconocidos es un gran problema en el contexto bioinformático, y es posible que más de una vez sea recomendable pensar un mecanismo particular para ciertos atributos, principalmente cuando se tiene interés en mantener la semántica biológica.

#### 5.4.3. Normalización

Para facilitar la comprensión de los resultados, o por exigencias de algunos algoritmos de AA, puede ser conveniente o necesario que el rango de valores de cada atributo numérico sea homogéneo, limitado a un mínimo y máximo fijo. Por ejemplo, los valores de expresión génica de los *arrays*.

La forma de realizar este preprocesamiento con Weka es con el filtro ('weka/filters') no supervisado ('unsupervised') de atributos ('attribute') denominado 'Normalize'. Si se quiere obtener una normalización de todos los atributos en el rango [0,1] (la que establece Weka por defecto), se utilizarán como parámetros: un factor de escala del rango de salida de 1 (S=1) y una traslación del rango de salida de 0 (T=0, porque el mínimo del rango deseado es 0). En resumen, el filtro se aplicaría según indica el siguiente comando:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.Normalize -S 1.0 -T 0.0  
-i entrada.arff -o salida.arff
```

Si se quiere otro rango, se pueden modificar los parámetros en Weka. Por ejemplo, para el rango [-1,1], se utilizaría un factor de escala de 2 (ancho del intervalo) y una traslación de -1 (distancia con respecto al mínimo por defecto), como indica la siguiente llamada:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.Normalize -S 2.0 -T -1.0  
-i entrada.arff -o salida.arff
```

#### 5.4.4. Discretización

Una serie de algoritmos de AA, como los conocidos árboles de decisión, exigen que los distintos valores de todos los atributos sean discretos y finitos. Por lo que para poder aplicar ciertos algoritmos a conjuntos de datos con atributos numéricos continuos, se necesita discretizarlos. Es decir, establecer una serie de intervalos fijos y limitados en los que poder distribuir los distintos valores existentes para cada instancia.

Existen muchos criterios para hacer discretización de variables continuas. En Weka existen dos filtros alternativos sobre atributos para realizar este preprocesamiento. Ambos se denominan 'Discretize', pero uno es supervisado (depende de la clase) y otro no supervisado. Cada uno tiene diversos parámetros, que permiten abarcar múltiples alternativas de discretización.

En la *discretización supervisada* de Weka, sólo es imprescindible establecer el rango de atributos que se quieren discretizar (si son todos, R=first-last). El comando correspondiente es el siguiente:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.supervised.attribute.Discretize -R first-last  
-i entrada.arff -o salida.arff
```

Para calcular las particiones, por defecto se emplea una heurística que minimiza la entropía [12]. Modificando los atributos adicionales de este filtro supervisado, se puede utilizar otra heurística ('-K'), elegir los puntos de separación de particiones más eficientemente ('-E'), o establecer que para cada uno de dichos puntos se cree un atributo binario diferente ('-D'), en lugar de uno solo con más de dos valores.

La *discretización no supervisada* no utiliza ninguna heurística que implique a la clase, sino que simplemente reparte los valores en distintas particiones, cuyos puntos de separación se calculan por número o anchura. El primer criterio de discretización establece un número de particiones fijas entre las que distribuir los valores de forma uniforme, y el segundo criterio determina un ancho fijo de partición, independientemente de la cantidad de ellas a las que dé lugar, que pueden ser establecidas previamente o dejar a Weka que optimice dicha cantidad. Por defecto, Weka discretiza todos los atributos (R=first-last), estableciendo un máximo de 10 particiones (B=10) por atributo de igual anchura, lo que correspondería al siguiente comando:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.Discretize -B 10 -R first-last  
-i entrada.arff -o salida.arff
```

Existen parámetros adicionales para modificar este comportamiento por defecto. Se puede obviar decidir a priori el número de particiones incluyendo el parámetro '-O'. Si en vez de particiones de igual anchura se prefiere dividir el rango de valores en particiones con igual frecuencia, se incluye la opción '-F', con la cantidad estimada de instancias que se quieren por cada partición (opción '-M'). Todos estos parámetros y algunos más se pueden combinar, acorde a sus incompatibilidades, para conseguir la discretización deseada. Por ejemplo, para una *discretización de igual frecuencia*, para los 5 primeros atributos, con unas 50 instancias por partición, se utilizaría el siguiente comando:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.unsupervised.attribute.Discretize -F -M 50 -R first-5  
-i entrada.arff -o salida.arff
```

#### 5.4.5. Selección de atributos

La motivación de este tipo de preprocesamiento es la *eliminación de atributos redundantes* o irrelevantes respecto al objetivo de predicción. Por ejemplo, un atributo claramente irrelevante para la predicción de función de una proteína es su identificador. Por otro lado, un atributo redundante sería una anotación de ruta biológica de KEGG, si ya se ha incluido en otro atributo la anotación de ruta biológica de la base de datos Reactome. Otro objetivo por el que se realiza selección de atributos es *reducir las dimensiones* de un problema, por razones técnicas (si no hay suficiente espacio en memoria para ejecutar el algoritmo deseado), o para facilitar la comprensión de los resultados por tener menos elementos, o para acelerar la ejecución del algoritmo.

La selección de atributos se puede hacer de forma manual o automática. Manualmente, se eliminan directamente las propiedades que se sabe que semánticamente no aportan nada (si no se han descartado ya en la fase de recopilación y selección). Si se tienen claros posibles atributos irrelevantes para el objetivo de predicción, es primordial eliminarlos a priori, porque un método automático sin la semántica del dominio y el conocimiento de los datos no lo podrá detectar ni eliminar tan correctamente como el dueño o recolector del conjunto de datos. Automáticamente, se optimiza el subconjunto de atributos que ofrece mejores resultados, dependiendo o no del algoritmo a aplicar. Para la selección automática existen diversas técnicas, que se pueden clasificar entre aquellas que evalúan los atributos individualmente, y las que los evalúan en conjunto. Cabe destacar como desventaja de las técnicas de selección de atributos evaluados independientemente, que son incapaces de detectar atributos relevantes en combinación con

otros, si de forma individual no lo son. Sin embargo, estas técnicas de evaluación individual son más rápidas. En ambos casos se trata de obtener una medida de la bondad de cada atributo, par de atributos o subconjunto de atributos, para posteriormente poder seleccionar los de mayor valor. Las medidas de bondad pueden ser varias, pero principalmente se trata de correlaciones entre los atributos y el valor de la clase (válidas en aprendizaje supervisado), o de valores de precisión de la predicción, calculados sobre diferentes subconjuntos. El mejor método de búsqueda de atributos relevantes es la búsqueda exhaustiva, que analiza todas las posibilidades de combinación de atributos, pero también es el más costoso en tiempo y recursos, por lo que generalmente se aplican otras técnicas menos exactas, pero también menos costosas.

En Weka existen diversas técnicas de selección de atributos implementadas, dedicando una pestaña completa a este tipo de preprocesado. Para cada técnica hay que seleccionar la medida de evaluación, y el método de búsqueda de atributos relevantes, que combinadas entre sí dan lugar a muy variadas formas de seleccionar atributos. También cabe destacar que, aunque se experimente con diferentes opciones en la pestaña ‘Select attributes’, para poder utilizar el fichero de datos con los atributos filtrados, se debe realizar la selección de atributos desde la pestaña ‘Preprocess’ de la interfaz gráfica. Se debe usar el filtro supervisado de atributos denominado ‘AttributeSelection’, incluyendo todos los parámetros correspondientes. A continuación se presentan algunos ejemplos. Una eliminación de atributos irrelevantes muy sencilla sería eliminar aquellos cuyos valores no varíen mucho en todo el conjunto de instancias, porque discriminarán poco (en Weka con el filtro ‘weka.filters.unsupervised.attribute.RemoveUseless’).

Para aplicar una selección de atributos con evaluación individual usando como medida la chi-cuadrado, se tendría que utilizar la siguiente instrucción:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.supervised.attribute.AttributeSelection  
-E weka.attributeSelection.ChiSquaredAttributeEval  
-S weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1  
-i entrada.arff -o salida.arff
```

Otro ejemplo sería una selección evaluando subconjuntos de atributos, en función de su correlación y con el algoritmo de búsqueda en escalada. En este caso la línea de comandos para ejecutar dicho preprocesamiento en Weka sería:

```
$ java -Xmx1024M -classpath weka.jar  
weka.filters.supervised.attribute.AttributeSelection  
-E weka.attributeSelection.CfsSubsetEval  
-S weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308  
-N -1 -i entrada.arff -o salida.arff
```

Weka también permite realizar reducción de dimensiones aplicando un *Análisis de Componentes Principales* (ver Capítulo 3, Subsección 3.6.2) con el filtro ‘weka.filters.unsupervised.attribute.PrincipalComponents’.

Es importante mencionar que la ejecución de una de estas operaciones, generalmente, es mucho más costosa que la de cualquiera de los filtros expuestos en los apartados anteriores, porque en este caso se requiere un mayor cálculo computacional, que las sencillas búsquedas y sustituciones de los preprocesados previos.

Para tener más detalles sobre las diferentes técnicas de selección de atributos se recomienda consultar [47] y el manual de Weka que indica qué métodos de selección de atributos están implementados y cómo.

#### 5.4.6. Extracción de características

Al contrario que en el tipo de preprocesamiento anterior, ahora no se trata de eliminar atributos, sino de generar otros nuevos, bien para incrementar los que existen, o bien para sustituirlos por otros más elaborados.

Se trata de realizar cálculos intermedios con los datos de los que se dispone para cada instancia, para generar un atributo que aporte información relevante o discriminatoria para el proceso de AA. Pueden ser cálculos agregados sobre un grupo de elementos (suma, media, mínimo, máximo, ...), incluyendo desde algo sencillo como el número de dominios de señal de una proteína, hasta algo más complejo como si una instancia es un complejo de proteínas, calcular la moda de sus anotaciones de función molecular en *Gene Ontology*. También es interesante realizar extracción de características cuando se tiene información que no se puede incluir directamente como atributos numéricos o nominales, como es la información estructurada. Por ejemplo, datos procedente de una red de interacción, donde es interesante incluir, para cada proteína o subgrafo, la cantidad de proteínas vecinas con las que interacciona, el camino más corto a otro subgrupo de la red, la densidad del subgrupo, etc.

Sobre este preprocesamiento, al ser muy concreto dependiendo del tipo de información y de dominio estudiado, Weka sólo incluye algunos filtros generales y sencillos para *añadir nuevos atributos* al conjunto de datos, en el paquete de filtros no supervisados sobre atributos ('weka/filters/unsupervised/attributes'). Por ejemplo, añadir el resultado de una expresión matemática calculada sobre los atributos existentes (filtro 'AddExpression'); añadir el identificador del grupo en el que se clasifica la instancia tras aplicar un algoritmo de agrupación sobre los datos (filtro 'AddCluster'); o añadir una copia de atributos existentes (filtro 'Copy'), para realizar una modificación posterior.

#### 5.4.7. Conjuntos no redundantes

En un proceso clásico de minería de datos, este paso de preprocesamiento no suele ser frecuente, por falta de necesidad, ante la ausencia de redundancia entre instancias. Pero en un problema bioinformático es fundamental, dada la gran cantidad de relaciones subyacentes que existen entre unos elementos biológicos y otros, y la información asociada a ellos, que puede sesgar los resultados del proceso.

Por un lado, hay que evitar que existan secuencias homólogas entre el conjunto de entrenamiento y el de prueba. Porque las anotaciones muchas veces se transfieren por homología, y por tanto, si para entrenar el sistema se utiliza una proteína 'P1' con sus anotaciones 'A1', y para evaluar el sistema se usa la proteína 'P2', homóloga de 'P1', con las mismas anotaciones 'A1', en realidad el sistema se probaría con un mismo conjunto de propiedades que ya se usó para entrenar ('A1'), aunque con un identificador de proteína diferente ('P1' vs 'P2'). Este hecho podría provocar una evaluación de la calidad del sistema más positiva de la realidad, es decir, de la que se obtendría al aplicar el sistema sobre proteínas con combinaciones de propiedades (anotaciones) no utilizadas directamente durante el entrenamiento.

Así, para evitar sesgos en la evaluación debidos a pares o grupos de genes o proteínas muy similares divididas entre el conjunto de entrenamiento y el de prueba, generalmente, lo que se hace es partir del conjunto completo de genes o proteínas y calcular un conjunto sin ninguna redundancia en homología interna, que posteriormente se divide en entrenamiento y prueba. La *reducción de redundancia* es un proceso conservador típico, siendo la mejor opción cuando la relación entre el origen evolutivo y las características de la secuencia no es fácil de determinar.

Para ello existen diversos algoritmos y programas ya implementados. Así, algunas opciones para obtener un conjunto no redundante serían usar la base de datos sin homólogos PDBselect [23, 25] (que busca

proteínas con estructuras 3D diferentes almacenadas en PDB), o las herramientas CD-HIT [36] o RedHom [37], que reducen un conjunto de secuencias a un subconjunto representativo con baja similitud de secuencia. No obstante, la eliminación de redundancia por homología puede mermar en más de un 50 % el tamaño del conjunto de datos. Por tanto, con el objetivo de no perder tantas instancias, se han ideado algoritmos para evitar redundancia entre distintas particiones del conjunto de muestras, aunque exista redundancia interna en cada partición [31].

Por otro lado, también hay que evitar redundancia entre la información que se utiliza para aprender (los atributos) y el objetivo de predicción (la clase), que deben ser ortogonales. En este caso, se debe tener cuidado y no usar propiedades y objetivo de predicción procedentes de bases de datos con información semejante, o que una utilice la otra para calcular su contenido.

## 5.5. Modelado. Aprendizaje automático

Una vez que se tienen los datos recopilados en un formato adecuado, el siguiente paso en un proceso de minería de datos es construir el *modelo de predicción*. Es la tarea más representativa de la minería de datos. Se trata de hacer un reconocimiento automático de patrones, o extracción de regularidades en los datos de entrada, mediante la aplicación de algún algoritmo de AA sobre el conjunto de datos de partida, denominado *conjunto de entrenamiento*.

Entre las diversas clasificaciones que existen de los sistemas de AA, a continuación se presenta una que depende del tipo de tarea a resolver:

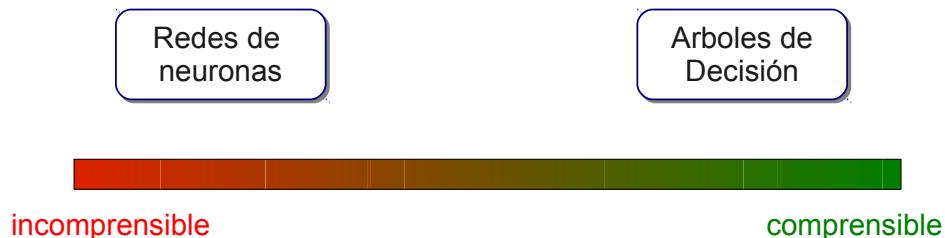
- *Clasificación*: se trata de aprender la clase o etiqueta a asignar a cada ejemplo, partiendo de un conjunto previamente etiquetado. Se incluye dentro de la categoría de aprendizaje supervisado, por el conocimiento inicial de las etiquetas. La regresión es un caso particular de la tarea de clasificación, en el que la salida no son clases discretas, sino valores numéricos en un rango continuo. La clasificación incluye la mayor parte de los algoritmos de AA, siendo muy frecuente cuando se usa AA.
- *Caracterización*: esta tarea también se conoce como de extracción de reglas de asociación. Se trata de determinar en qué se parecen varios ejemplos entre sí, teniendo presente la clase o no, cuya existencia no es obligatoria. Así, se reconoce cómo la ocurrencia de un suceso (un valor de un atributo o un conjunto de ellos) puede generar la aparición de otros.
- *Agrupación* (o *clustering*): los algoritmos que resuelven esta tarea se incluye en la categoría de aprendizaje no supervisado. Es debido a que en este caso no se tiene ninguna etiqueta previa para cada ejemplo, sino que se pretenden obtener grupos naturales de elementos, los cuales abarquen instancias que tengan gran similitud entre sí y diferencias con las de otros grupos, para entender el comportamiento de cada grupo y tratarlo de manera adecuada. En algunos casos interesa obtener un prototipo por grupo, que describa las propiedades generales de todos los elementos del mismo.

Existen multitud de algoritmos, tanto para resolver tareas de clasificación, de caracterización o de agrupamiento. La descripción de dichos algoritmos queda fuera del ámbito de este capítulo, pudiéndose consultar una descripción detallada de todos los algoritmos clásicos y relevantes de AA en [41]. Además, ciertos algoritmos tienen una base conceptual estadística, como por ejemplo los bayesianos, los de regresión lineal o los de *clustering*, por lo que para tener una visión completa de todos los algoritmos de AA también es conveniente consultar el Capítulo 3, Sección 3.4.

Según un estudio del 2011 [30] los métodos de AA supervisado más usados en biología son las *redes de neuronas*, las *máquinas de vector de soporte*, los *modelos de Markov* y los *árboles de decisión*, con la

aparición creciente de los *bosques de árboles aleatorios* (del inglés, *random forests*). No obstante, para seleccionar el algoritmo de AA más adecuado para cada situación, se deben tener en cuenta diferentes factores, cuyos valores varían de un algoritmo a otro. Entre dichos factores cabe mencionar:

- *Capacidad de interpretación*: se trata de la legibilidad y facilidad de interpretación del modelo de salida. Es un criterio relevante para decidir. Se trata de elegir según la forma deseada para la salida del predictor (un conjunto ramificado de condiciones, una lista de reglas, un modelo probabilístico, el conjunto completo de ejemplos entrenamiento, etc.) Si se está interesado en poder justificar los resultados de la salida del proceso de minería de datos, se debe seleccionar un algoritmo que genere un modelo de predicción simbólico, como son los árboles de decisión. Por ejemplo, si se quieren conocer los valores de los atributos por los que el modelo predictivo determina que la localización celular de una proteína es un orgánulo concreto y no otro. Si lo único que importa es la predicción en sí misma, se puede utilizar un algoritmo de “caja negra” o sub-simbólico, como son las redes de neuronas, cuya salida no se puede interpretar, como muestra la Figura 5.10. Por ejemplo, cuál es la probabilidad de que una proteína se exprese en el núcleo, sin importar las razones que el modelo predictivo utiliza para llegar a esa conclusión.



**Figura 5.10:** Esquema de la capacidad de interpretación de los modelos de AA.

- *Tipo de datos de entrada* (numéricos naturales o reales, discretizados, nominales o etiquetados, etc.): unos algoritmos exigen un cierto formato en sus datos de entrada, aunque otros sean más flexibles. Por ejemplo, un árbol de decisión requiere que los valores de todos los atributos sean nominales. Por lo que si la entrada original es numérica, y se quiere aplicar este algoritmo por la fácil interpretación de su salida, en la fase anterior de preprocesamiento y transformación, se deberán discretizar dichos valores numéricos continuos a un conjunto fijo y finito de rangos discretos, siempre que sea posible.
- *Volumen de datos*: puede limitar el uso de algunos algoritmos que no soporten una gran cantidad de ejemplos, o de atributos, o de valores diferentes de los atributos, según la cantidad de datos que necesite cargar en memoria para realizar sus cálculos necesarios. Por ejemplo, un modelo bayesiano como es AODE [56], incluido en la herramienta Weka, necesita calcular la probabilidad condicionada de que un atributo tome un valor determinado, frente a otro atributo con otro valor concreto; por lo tanto, se limita la cantidad de valores diferentes de cada uno de los atributos, que no pueden ser de cientos o miles, como puede suceder al discretizar atributos numéricos.
- *Tiempo de modelado*: este factor está directamente relacionado con el anterior, pues en general, un gran volumen de datos requiere un mayor tiempo de modelado. Pero también con otras implicaciones, dependiendo si se requiere repetir el entrenamiento en tiempo real, donde se necesita un algoritmo rápido, o basta con un modelo precalculado, aunque sea de cálculo lento.
- *Tolerancia al ruido*: unos algoritmos soportan mejor la existencia de ruido en los datos que otros,

y se debe tener muy en cuenta si en los datos puede haber mucho ruido o inexactitud, como suele ocurrir en los conjuntos de datos biológicos de origen experimental.

Por lo tanto, aunque sobre cualquier conjunto de datos se puedan aplicar muchos algoritmos de AA (de todos los que se cumplan las exigencias mínimas), se recomienda tener presentes estos criterios de selección, para obtener el modelo de predicción que más se ajuste a nuestras necesidades, nuestro objetivo y nuestros datos, lo que posiblemente redundará en mejores resultados.

## 5.6. Evaluación e interpretación de resultados

La última fase del proceso de minería de datos analiza los resultados obtenidos, evaluándolos de forma cuantitativa e interpretando su significado. Se realiza una *evaluación cuantitativa* para poder tener una estimación sobre cuál será la fiabilidad del sistema cuando se aplique para predecir sobre nuevos conjuntos de datos. Además, dicha evaluación cuantitativa también permite que se comparan los resultados de varios algoritmos de AA, incluso para poder elegir cuál es el mejor, en términos de rendimiento. Por otro lado, también es conveniente realizar una *interpretación* de los resultados, sobre todo en problemas bioinformáticos, donde no se está interesado sólo en un número de fiabilidad de la predicción, sino que se quiere comprender qué se predice y por qué.

Esta sección se centra en la evaluación e interpretación en tareas de clasificación, que son las más comunes del AA.

### 5.6.1. Estimación del rendimiento

La evaluación de un sistema de clasificación consiste en calcular los valores estimados de cómo funcionará el modelo aprendido cuando se aplique sobre nuevos datos, para prever la bondad o fiabilidad del sistema.

El *rendimiento* cuantifica la predicción correcta o incorrecta, pudiendo ser una medida de error o su complementaria en positivo. Existen múltiples medidas de rendimiento, siendo las más frecuentes presentadas en la Subsección 5.6.2 y la Subsección 5.6.3.

Así, el valor del rendimiento del sistema según dichas medidas se estima sobre un conjunto de datos con salida o clase conocida. Para cada instancia del conjunto, se compara la predicción que proporciona el sistema (o clase predicha), con la clase real de los datos. El conjunto de clase conocida sobre el que se realiza la estimación puede ser diferente, teniendo en cuenta su relación con el conjunto de entrenamiento. Las opciones más comunes se exponen a continuación:

#### Conjunto de entrenamiento

Se podría utilizar el mismo conjunto de datos usados para aprender, para calcular el rendimiento también, dado que es un conjunto de datos con clase conocida. Pero en este caso no se mediría si el sistema generaliza, porque se le presentan los mismos ejemplos con los que aprendió, sin ninguno desconocido para el sistema. Por lo tanto, el conjunto de entrenamiento no es válido como una evaluación independiente, pues no sirve como estimación de la fiabilidad del sistema ante nuevas muestras.

## Conjunto de prueba

Se denomina conjunto de prueba a un conjunto de datos con clase conocida, pero que no se ha utilizado en la fase de entrenamiento, por lo que el sistema no los ha visto, y sí sería una evaluación independiente. Tradicionalmente en AA este conjunto se obtiene inicialmente, dividiendo el conjunto original de datos etiquetados en 2/3 para el conjunto de entrenamiento y 1/3 para el conjunto de prueba.

Un conjunto de prueba obtenido de esta manera, aunque independiente, puede incluir sesgos por las instancias concretas que quedan en este conjunto con respecto a las que van a la partición usada para el entrenamiento. Así, un paso más para mejorar la estimación del rendimiento sería repetir varias divisiones entrenamiento/prueba, como propone la siguiente opción.

## Validación cruzada

El sistema de validación cruzada propone evitar los sesgos de la división del conjunto entre entrenamiento y prueba, calculando varias particiones, repitiendo varias fases de entrenamiento y prueba, y promediando dichas evaluaciones independientes. El número de evaluaciones es igual al de particiones, de forma que la validación cruzada se asegura que una instancia usada en el entrenamiento no se use a la vez en el test, y además, que todas las instancias se utilicen una y sólo una vez en el test, para no sesgar la evaluación promedio. Conviene que las particiones sean estratificadas, es decir, que se mantenga en todas las particiones la proporción que existe entre las clases en el conjunto de datos completo.

Así, de las  $k$  particiones, en la evaluación 1, se utilizan las particiones 1 a  $n - 1$  para entrenar, y la última para test; en la evaluación 2, las particiones 1 a  $n - 2$  y  $n$  para entrenar, y la penúltima para test; y así sucesivamente, hasta llegar a las  $n$  evaluaciones. El número de particiones típico es  $n = 10$ , aunque desde 3 se considera un valor aceptable.

Weka permite estos tres tipos de evaluación del rendimiento (en la herramienta, ver caja ‘Test options’, en la pestaña ‘Classify’ a la izquierda).

### 5.6.2. Medidas de rendimiento unidimensionales

Partiendo de un conjunto de instancias con una clase predicha y una clase real asociada a cada una, se pueden aplicar cualquiera de las medidas de rendimiento que se presentan a continuación.

#### TP, FP, FN y TN: Matriz de confusión

Se trata del conjunto de medidas básicas, siendo las utilizadas como axiomas por la mayoría del resto de medidas que se presentan a continuación.

Dado un problema de clasificación binaria (i.e. con dos clases, o multi-clase descompuesto en problemas binarios), asumiendo que una clase se denomina positiva y otra negativa, el conjunto de instancias predichas se pueden clasificar en 4 categorías diferentes y disjuntas:

- *TP (True Positive)* (verdadero positivo): instancia de la clase positiva predicha correctamente como positiva.
- *FP (False Positive)* (falso positivo): instancia de la clase negativa, predicha erróneamente como positivo.

- *FN (False Negative)* (falsos negativos): instancia de la clase positiva, predicha erróneamente como negativo.
- *TN (True Negative)* (verdadero negativo): instancia de la clase negativa, predicha correctamente como negativa.

		clase real	
		positivo	negativo
clase predicha	positivo	verdaderos positivos (TP)	falsos positivos (FP)
	negativo	falsos negativos (FN)	verdaderos negativos (TN)

**Figura 5.11:** Matriz de confusión.

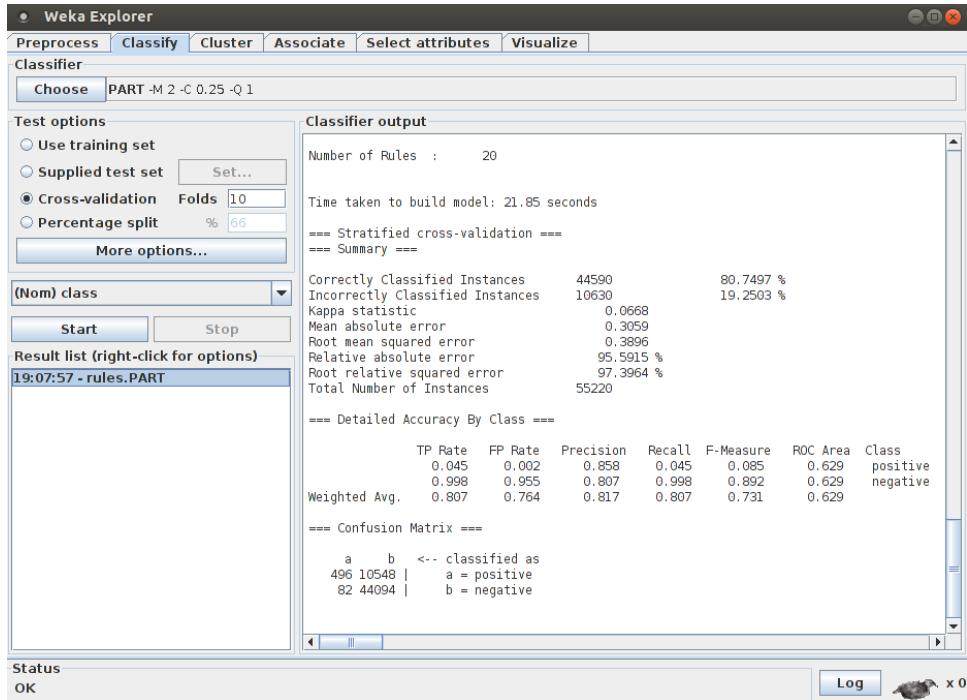
La Figura 5.11 esclarece esta distribución en categorías. Al conjunto de los 4 valores, para un conjunto de evaluación, distribuidos de esta forma, se le conoce como *matriz de confusión*. La situación ideal de acierto completo en la predicción sólo tendría valores en la diagonal principal formada por TP y TN; sin ningún fallo, representado por FP y FN. Si el problema tiene más de dos clases, la dimensiones de la matriz de confusión aumentan de forma proporcional al número de clases; pero el rendimiento se seguirá midiendo acorde a la diagonal principal, siendo un sistema mejor cuantos más ceros tenga fuera de la diagonal principal.

Si la salida de un algoritmo de AA son clases discretas, la distribución entre TP, FP, FN y TN es clara, pues sólo hay que comparar la clase predicha con la real y ver si coinciden o no. Sin embargo, cuando la salida es numérica, la forma de definir cuál es la clase predicha depende de un *umbral de confianza* de separación entre la predicción positiva y la negativa. Así, si la confianza de la predicción de una instancia está por encima de ese umbral, la predicción es positiva, y si está por debajo, la predicción es negativa. El umbral que se considera por defecto, para una salida normalizada entre 0 y 1, es 0,5; pero no tiene que ser el más adecuado, sino que dependerá del caso, de la precisión de las predicciones que se quieran obtener, de la cantidad de las mismas, etc.

Para poder aplicar dicho umbral, Weka asocia a cada instancia del conjunto de evaluación, un valor de salida del algoritmo o la probabilidad de pertenecer a la clase en cuestión. Se puede obtener dicho listado de probabilidades para cualquier algoritmo, incluso con clases discretas, activando la opción ‘Output predictions’ en el botón ‘More options...’ de la caja ‘Test options’; o por la línea de comandos incluyendo la opción ‘-p’.

La Figura 5.12, muestra un ejemplo de resultados de evaluación de la herramienta Weka, que aparece en la mayor caja, titulada ‘Classifier output’, en la pestaña ‘Classify’ a la derecha. En esta figura se puede localizar la matriz de confusión, y otras medidas que se describen más adelante.

Así, la matriz de confusión, con las 4 medidas complementarias de rendimiento que incluye, permite calcular las principales métricas de rendimiento en clasificación que se presentan en los siguientes apartados.



**Figura 5.12:** Ejemplo de pantalla de resultados de clasificación en Weka.

## Tasa de aciertos global

Es una medida genérica, que representa la proporción de instancias correctamente predichas entre todas las que se le presentan al sistema:

$$\text{Tasa de aciertos global (accuracy)} = \frac{TP + TN}{TP + FP + FN + TN} \quad (5.1)$$

## Sensibilidad, Especificidad, Precisión y *Recall*

Se trata de tasas o ratios parciales, tanto de aciertos como de fallos.

- La *precisión* representa la proporción de ejemplos predicha correctamente, entre todos los ejemplos que se predicen como positivos:

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (5.2)$$

- La *sensibilidad* (*o recall*) representa la proporción de ejemplos que se predicen entre todos los que se deberían predecir, los que son realmente de la clase positiva:

$$\text{Sensibilidad o Ratio de verdaderos positivos (Recall)} = \frac{TP}{TP + FN} \quad (5.3)$$

- La *especificidad* representa la proporción de ejemplos excluidos (clasificados como negativos) correctamente, entre los que son realmente de la clase negativa:

$$\text{Especificidad o Ratio de verdaderos negativos (Specificity)} = \frac{TN}{TN + FP} \quad (5.4)$$

- $1 - \text{especificidad}$  representa la proporción de ejemplos que se deberían excluir y se están prediciendo erróneamente como positivos, entre todos los ejemplos negativos:

$$1 - \text{especificidad} \text{ o } \text{Ratio de falsos positivos} = \frac{FP}{FP + TN} \quad (5.5)$$

La sensibilidad y especificidad se suelen usar en conjunto como medidas complementarias, que indican la cantidad de aciertos en clase positiva y negativa, respectivamente. La precisión y el *recall* por su lado, también se utilizan como medidas complementarias que miden la exactitud en el acierto frente a la cobertura, respectivamente. La sensibilidad y (1-especificidad) son los ejes de la curva ROC, mientras que la precisión y el *recall* son los ejes de la curva PR, ambas explicadas en la Subsección 5.6.3.

Cabe destacar que también se pueden asociar "costes" diferentes a los diferentes tipos de error de clasificación. Estos costes pueden incluirse explícitamente en las fórmulas de estimación del error, o pueden utilizarse para elegir la medida más adecuada en cada caso. Por ejemplo, en un problema de predicción de implicación de un gen en el desarrollo de un tumor, se prefiere predecir bastantes genes, aunque todos no estén implicados finalmente (existencia de FP) que dejar fuera algún gen realmente implicado sin predecir (existencia de FN); es decir, se prefieren errores en FP que en FN, o se utilizaría como medida de estimación del rendimiento el *recall* frente a la precisión, que prima la cobertura.

## Medida F

La medida *F* (*F-measure*) combina la precisión y la sensibilidad en una sola métrica, siendo frecuentemente utilizada, con distintas aproximaciones [54]:

$$F = 2 \times \frac{\text{precisión} \times \text{sensibilidad}}{\text{precisión} + \text{sensibilidad}} \quad (5.6)$$

## Coeficiente de correlación de Matthews

Del inglés, *Matthews Correlation Coefficient* (MCC), esta medida unifica los cuatro valores de la matriz de confusión [38]:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5.7)$$

## AUROC y AUPRC

Son dos valores numéricos unidimensionales correspondientes a dos medidas bidimensionales de rendimiento en formato de curva, explicados en la Subsección 5.6.3: las curvas ROC y las curvas PR. La idea que se adopta para representar una curva con un solo número es calcular el área bajo dicha curva.

En la práctica, este área se calcula por aproximaciones basadas en sumatorios de áreas de trapecios. Weka presenta el AUROC en sus resultados automáticamente (ver Figura 5.12). Tanto AUROC como AUPRC se pueden calcular con R, y con el software libre específico para ello denominado 'AUC calculator' [8].

### Media-micro y media-macro (multi-clase)

Cuando se trabaja con más de una clase, además de dar los valores de rendimiento para cada clase independiente, puede requerirse una medida unificada del rendimiento del sistema completo. Para ello, existen dos opciones básicas que promedien los resultados [58]. La *media-micro* (del inglés, *micro-average*) que calcula la matriz de confusión de forma global para todas las clases, y posteriormente aplica la fórmula de la métrica elegida; y la *media-macro* (del inglés, *macro-average*) que construye matrices de confusión individuales y calcula la métrica de forma independiente para cada clase, y posteriormente calcula la media aritmética de todas ellas.

Existe una gran distinción entre las medidas promedio macro y micro [48], pudiendo dar resultados bastante diferentes, sobre todo si las clases tienen distintas frecuencias sobre el conjunto de ejemplos. La media-micro da un peso equivalente a todos los ejemplos, y por lo tanto se considera una media por ejemplo, específicamente, una media por pares ejemplo-clase. Sin embargo, la media-macro da el mismo peso a cada una de las clases, sin importar su frecuencia, siendo por tanto una media por clase.

Usar una u otra medida depende de los requisitos del problema. Por ejemplo, en los problemas de anotación funcional no suele existir una distribución homogénea de ejemplos en clases. Es decir, una función a nivel de proceso puede necesitar la colaboración de cientos de proteínas para llevarse a cabo, aunque otra función diferente pueda concluirse con éxito con sólo diez proteínas. Por lo tanto, en este caso, convendría utilizar la media-macro, para no favorecer que las funciones más frecuentes se predigan mejor que las minoritarias.

### Medidas en regresión

Cuando los valores de la clase no son discretos, sino continuos, la tarea de clasificación se denomina *regresión*, porque se trata de aproximar una función desconocida cuya salida es el valor de la clase. En este caso, las medidas de rendimiento miden el error cometido por la función aprendida. Entre las más frecuentes se encuentra el *coeficiente de correlación* (entre -1 y 1) y la *raíz del error cuadrático relativo* (RRSE, del inglés, *Root Relative Squared Error*) (en porcentaje):

$$RRSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}} \quad (5.8)$$

Una vez que se tiene calculado el rendimiento, a veces se quieren comparar varios modelos de AA. Para ello, se suelen aplicar test estadísticos, para verificar si las diferencias entre los dos modelos son relevantes cuantitativamente (ver Capítulo 3).

#### 5.6.3. Medidas de rendimiento bidimensionales

Además de las métricas unidimensionales, puede resultar interesante disponer de un gráfico en dos dimensiones en el que evaluar más de una medida de rendimiento a la vez; como es el caso de buscar un equilibrio entre dos de ellas. Las medidas utilizadas más frecuentemente son la curva ROC y la curva PR, que son las que se presentan en esta sección, aunque existen otras como las curvas de coste [10].

La *curva ROC* (del inglés, *Receiver Operating Characteristic*) representa el ratio de verdaderos positivos o sensibilidad (eje *y*) frente al ratio de falsos positivos o 1-especificidad (eje *x*) [11]. La *curva PR* (del inglés, *Precision-Recall*) representa el ratio de acierto en los positivos predichos o precisión (eje *y*) frente al ratio de acierto en los positivos reales o sensibilidad (eje *x*) [8]. Cabe destacar que a cada

punto de la curva ROC le corresponde un punto de la curva PR [8]. La Figura 5.13 presenta un ejemplo de ambas curvas.

Al utilizar como medida una de estas curvas, no se está midiendo el rendimiento de un clasificador, sino de muchos de ellos, obtenidos todos a partir de una única ejecución de un algoritmo de AA, representado cada uno por un punto en la curva. Para obtener cada uno de estos puntos, se debe ir variando el umbral de confianza de separación entre la predicción positiva y la negativa, y calcular una matriz de confusión diferente para cada umbral, a partir de la cual obtener las dos medidas de rendimiento unidimensionales correspondientes. De esta forma, la curva completa también es útil para elegir el umbral de confianza, sin asumir que el valor por defecto de 0,5 sea el mejor; sino que se puede seleccionar el punto en el que se obtenga un valor que se considere aceptable para cada una de las medidas representadas en cada eje.

Se debe tener en cuenta que en estas curvas, en general, como se usan medidas complementarias, cuando el valor de una de las medidas representadas es elevado, el de la otra medida es bajo; aumentando y disminuyendo de forma paralela, respectivamente. Por lo tanto, en la mayoría de los casos no se podrá elegir el mayor valor de las dos medidas representadas, sino que habrá que decidir qué se prefiere primar. Por ejemplo, en las curvas ROC, si interesa una tasa elevada de aciertos en positivos (valor elevado en eje y), o una baja cantidad de falsos positivos (valor elevado en eje x).

Si se comparan ambas curvas, las PR se centran principalmente en analizar los aciertos de la clase positiva, sin prestar apenas atención a los aciertos de la clase negativa, a los que las curvas ROC les da una mayor importancia. Además, las curvas PR son más adecuadas que las curvas ROC [8] cuando se trabaja con conjuntos de datos muy sesgados, es decir, con mucha diferencia en el número de ejemplos de cada clase. Por lo tanto, al menos según estas consideraciones, se debe elegir si en el problema concreto interesa ser certero en las predicciones positivas o si también importan las negativas. Por ejemplo, elegir una curva PR para primar un mayor acierto en las funciones que tiene asociadas un gen o proteína, aunque perdiendo precisión en las funciones no relacionadas.

Estas medidas bidimensionales también permiten comparar los resultados de varios algoritmos de AA, dibujando más de una curva en el mismo gráfico, de forma que se pueda observar qué algoritmo y con qué umbral es mejor en cada región del espacio bidimensional representado.

## Cómo dibujar una curva ROC o PR

La Figura 5.13 explica cómo obtener en Weka una curva ROC (por defecto) y su curva PR equivalente, a partir de la ventana de resultados.

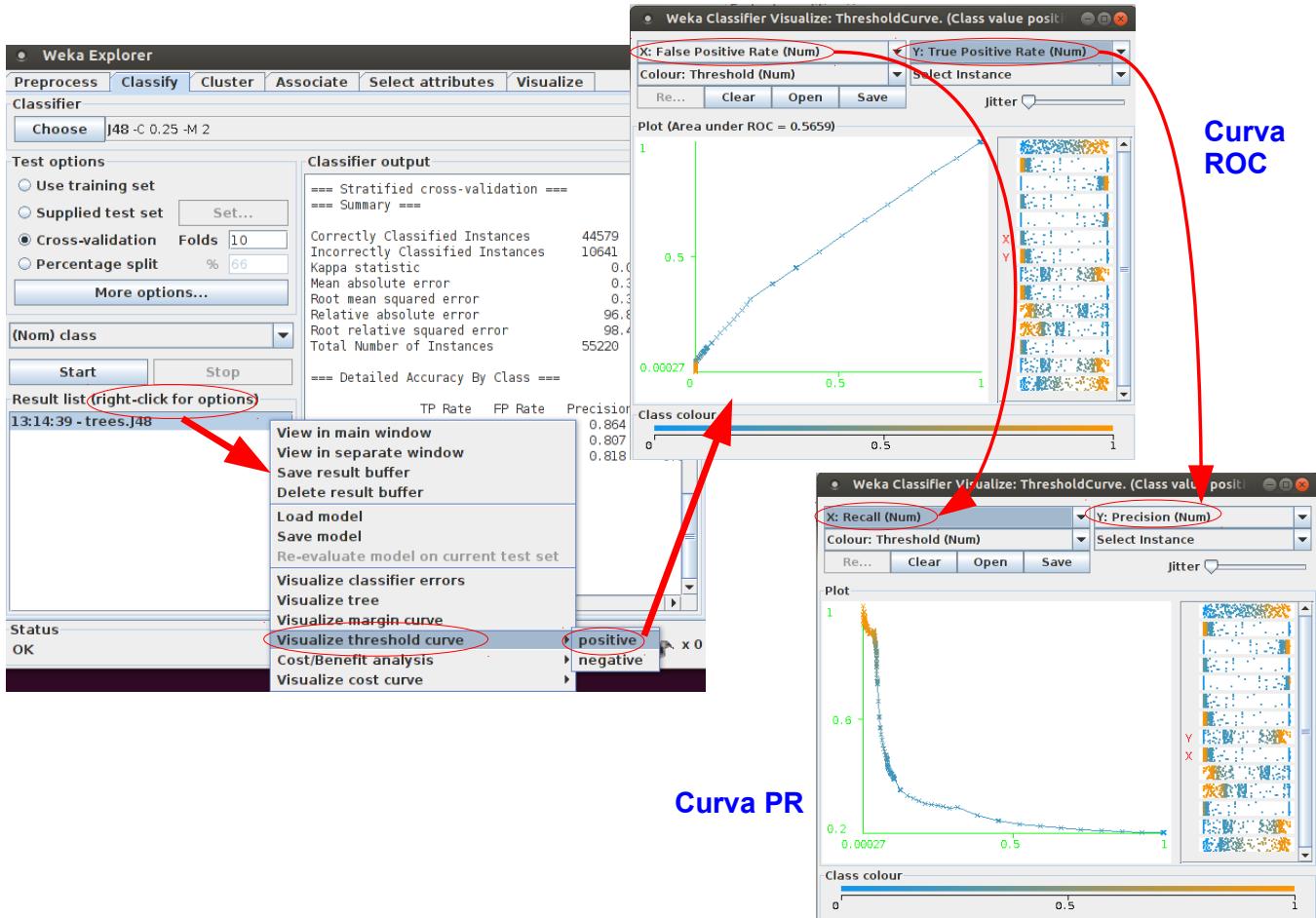
En R, se puede hacer con los siguientes tres comandos del paquete ROCR, para una curva ROC:

```
> pred <- prediction(predictions, labels)
> perf <- performance(pred, measure = "tpr", x.measure = "fpr")
> plot(perf, col=rainbow(10))
```

y para una curva PR, cambiando la línea central por:

```
> perf <- performance(pred, measure = "prec", x.measure = "rec")
```

No obstante, Weka y R permiten representar otros pares de medidas de rendimiento, variando los ejes dentro de un conjunto variado (ratio de verdaderos/falsos positivos, ratio de error, medida F, MCC, etc.).



**Figura 5.13:** Obtención de curvas ROC y PR con Weka.

#### 5.6.4. Interpretación biológica

En un proceso de minería de datos, además de utilizar las medidas expuestas en las secciones anteriores para realizar una evaluación cuantitativa de los resultados, en bioinformática resulta fundamental *interpretar y analizar la relevancia biológica de las predicciones*, para comprender los resultados obtenidos y valorarlos en su contexto de problema real.

Para esta tarea, se pueden usar medidas muy variadas, según el objetivo biológico específico buscado. En general, se trata de métricas sencillas, como por ejemplo, la cobertura de la predicción, la relevancia de los atributos utilizados, la similitud semántica, la búsqueda de anotaciones funcionales en bases de datos y en la literatura científica, la diversidad en las predicciones en una misma clase, el solapamiento entre predicciones en distintas clases, o el significado del modelo aprendido (si se puede estudiar, como en los árboles o reglas de decisión).

Entre ellas, cabe destacar las *medidas de similitud semántica basadas en ontologías*, que estiman cuantitativamente la similitud funcional entre genes y productos genéticos, a través de sus anotaciones funcionales [43]. Esta métrica se utiliza ante la falta de una evaluación experimental de anotaciones nuevas; y también para otros análisis en los que se necesita conocer el grado de relación de dos o más genes o proteínas en términos de sus anotaciones.

## 5.7. Bibliografía

- [1] Metagenomics versus moore's law. *Nat Meth*, 6(9):623–623, 2009.
- [2] A. Al-Shahib, R. Breitling, and D. Gilbert. Feature selection and the class imbalance problem in predicting protein function from sequence. *Applied Bioinformatics*, 4(3):195–203, 2005.
- [3] J. Amberger, C. A. Bocchini, A. F. Scott, and A. Hamosh. McKusick's online mendelian inheritance in man (OMIM(R)). *Nucleic Acids Research*, 37(Database):D793–D796, Jan. 2009.
- [4] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. Bradford Books, Cambridge, Massachusetts, U.S.A., Massachusetts, 2001.
- [5] T. Barrett, D. B. Troup, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, R. N. Muertter, M. Holko, O. Ayanbule, A. Yefanov, and A. Soboleva. NCBI GEO: archive for functional genomics data sets—10 years on. *Nucleic Acids Research*, 39(Database issue):D1005–D1010, Jan. 2011. PMID: 21097893 PMCID: PMC3013736.
- [6] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, R. Lopez, and C. A. Goble. BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic Acids Research*, 38(Web Server):W689–W694, May 2010.
- [7] T. E. P. Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, Sept. 2012.
- [8] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning*, pages 233–240. ACM, 2006.
- [9] R. Dowell, R. Jokerst, A. Day, S. Eddy, and L. Stein. The distributed annotation system. *BMC Bioinformatics*, 2(1):7, 2001.
- [10] C. Drummond and R. C. Holte. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130, 2006.
- [11] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Technical report, HP Laboratories, 2003.
- [12] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.
- [13] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, 1996.
- [14] P. Flicek, I. Ahmed, M. R. Amode, D. Barrell, K. Beal, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gil, C. Garcia-Giron, L. Gordon, T. Hourlier, S. Hunt, T. Juettemann, A. K. Kahari, S. Keenan, M. Komorowska, E. Kulesha, I. Longden, T. Maurel, W. M. McLaren, M. Muffato, R. Nag, B. Overduin, M. Pignatelli, B. Pritchard, E. Pritchard, H. S. Riat, G. R. S. Ritchie, M. Ruffier, M. Schuster, D. Sheppard, D. Sobral, K. Taylor, A. Thormann, S. Trevanion, S. White, S. P. Wilder, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, J. Harrow, J. Herrero, T. J. P. Hubbard, N. Johnson, R. Kinsella, A. Parker, G. Spudich, A. Yates, A. Zadissa, and S. M. J. Searle. Ensembl 2013. *Nucleic Acids Research*, 41(D1):D48–D55, Nov. 2012.
- [15] G. B. Fogel. Computational intelligence approaches for pattern discovery in biological systems. *Briefings in Bioinformatics*, 9(4):307–316, July 01 2008.
- [16] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten. Data mining in bioinformatics using weka. *Bioinformatics*, 20(15):2479–2481, October 12 2004.
- [17] M. Y. Galperin and X. M. Fernandez-Suarez. The 2012 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Research*, 40(D1):D1–D8, Dec. 2011.
- [18] B. García-Jiménez, D. Juan, I. Ezkurdia, E. Andrés-León, and A. Valencia. Inference of functional relations in predicted protein networks with a machine learning approach. *PLoS ONE*, 5(4):e9969, 04/01 2010.
- [19] N. García-Pedrajas and A. de Haro García. *Output Coding Methods: Review and Experimental Comparison*, pages 327–344. Pattern Recognition Techniques, Technology and Applications. InTech, Austria, 2008.
- [20] P. Gaudet, A. Bairoch, D. Field, S. Sansone, C. Taylor, T. K. Attwood, A. Bateman, J. A. Blake, C. J. Bult, J. M. Cherry, R. L. Chisholm, G. Cochrane, C. E. Cook, J. T. Eppig, M. Y. Galperin, R. Gentleman, C. A. Goble,

- T. Gojobori, J. M. Hancock, D. G. Howe, T. Imanishi, J. Kelso, D. Landsman, S. E. Lewis, I. Karsch Mizrachi, S. Orchard, B. F. Ouellette, S. Ranganathan, L. Richardson, P. Rocca-Serra, P. N. Schofield, D. Smedley, C. Southan, T. W. Tan, T. Tatusova, P. L. Whetzel, O. White, and C. Yamasaki. Towards BioDBcore: a community-defined information specification for biological databases. *Database: The Journal of Biological Databases and Curation*, 2011, Jan. 2011. PMID: 21205783 PMCID: PMC3017395.
- [21] J. E. Gewehr, M. Szugat, and R. Zimmer. BioWeka extending the Weka framework for bioinformatics. *Bioinformatics*, 23(5):651–653, March 1 2007.
  - [22] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Research*, 38(Web Server):W677–W682, May 2010.
  - [23] S. Griep and U. Hobohm. PDBselect 1992-2009 and PDBfilter-select. *Nucleic Acids Research*, 38(suppl 1):D318–D319, January 01 2010.
  - [24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.
  - [25] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of representative protein data sets. *Protein Science*, 1(3):409–417, 1992.
  - [26] G. Holmes, D. Fletcher, and P. Reutemann. An application of data mining to fruit and vegetable sample identification using gas chromatography-mass spectrometry. In *Proceedings of the International Congress on Environmental Modelling and Software (IEMSS)*, Leipzig, Germany, 2012.
  - [27] H. Huang, Z. Z. Hu, C. N. Arighi, and C. H. Wu. Integration of bioinformatics resources for functional analysis of gene expression and proteomic data. *Frontiers in Bioscience: a Journal and Virtual Library*, 12:5071–5088, Sep 1 2007.
  - [28] I. Inza, B. Calvo, R. Armañanzas, E. Bengoetxea, P. Larrañaga, and J. A. Lozano. Machine learning: an indispensable tool in bioinformatics. *Methods in Molecular Biology*, 593:25–48, 2010.
  - [29] R. Jansen and M. Gerstein. Analyzing protein function on a genomic scale: the importance of gold-standard positives and negatives for network prediction. *Current Opinion in Microbiology*, 7(5):535–545, 10 2004.
  - [30] L. J. Jensen and A. Bateman. The rise and fall of supervised machine learning techniques. *Bioinformatics*, 27(24):3331–3332, December 15 2011.
  - [31] L. J. Jensen, R. Gupta, H. H. Staerfeldt, and S. Brunak. Prediction of human protein function according to Gene Ontology categories. *Bioinformatics*, 19(5):635–642, March 22 2003.
  - [32] A. Juncker, L. J. Jensen, A. Pierleoni, A. Bernsel, M. Tress, P. Bork, G. von Heijne, A. Valencia, C. Ouzounis, R. Casadio, and S. Brunak. Sequence-based feature prediction and annotation of proteins. *Genome Biology*, 10(2):206, 2009.
  - [33] S. Kawashima and M. Kanehisa. Aaindex: Amino acid index database. *Nucleic acids research*, 28(1):374–374, January 01 2000.
  - [34] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, March 01 2006.
  - [35] R. Leinonen, R. Akhtar, E. Birney, L. Bower, A. Cerdeno-Tárraga, Y. Cheng, I. Cleland, N. Faruque, N. Goodgame, R. Gibson, G. Hoad, M. Jang, N. Pakseresht, S. Plaister, R. Radhakrishnan, K. Reddy, S. Sobhany, P. Ten Hoopen, R. Vaughan, V. Zalunin, and G. Cochrane. The european nucleotide archive. *Nucleic Acids Research*, 39(Database issue):D28–D31, Jan. 2011. PMID: 20972220 PMCID: PMC3013801.
  - [36] W. Li and A. Godzik. CD-HIT: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, July 01 2006.
  - [37] O. Lund, K. Frimand, J. Gorodkin, H. Bohr, J. Bohr, J. Hansen, and S. Brunak. Protein distance constraints predicted by neural networks and probability density functions. *Protein Engineering*, 10(11):1241–1248, November 01 1997.
  - [38] B. W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta*, 405(2):442–451, Oct 20 1975.

- [39] M. L. Metzker. Sequencing technologies – the next generation. *Nature Reviews Genetics*, 11(1):31–46, Dec. 2009.
- [40] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD ’06, pages 935–940, New York, NY, USA, 2006. ACM.
- [41] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [42] T. M. Mitchell. Machine learning and data mining. *Communications of the ACM*, 42:30–36, 1999.
- [43] C. Pesquita, D. Faria, A. O. Falcão, P. Lord, and F. M. Couto. Semantic similarity in biomedical ontologies. *PLoS Computational Biology*, 5(7):e1000443, 07/31 2009.
- [44] A. Prlic, T. Down, E. Kulesha, R. Finn, A. Kahari, and T. Hubbard. Integrating sequence and structural biology with DAS. *BMC Bioinformatics*, 8(1):333, 2007.
- [45] B. Rost, J. Liu, R. Nair, K. O. Wrzeszczynski, and Y. Ofran. Automatic prediction of protein function. *Cellular and Molecular Life Sciences*, 60(12):2637–2650, Dec 2003.
- [46] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2009.
- [47] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, October 01 2007.
- [48] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.
- [49] A. C. Smith and J. G. Cleary. All of medline indexed to the gene ontology. In *Proc Sixth Annual Bio-Ontologies Meeting*, pages 7–11, Brisbane, Australia, 2003.
- [50] A. L. Tarca, V. J. Carey, X. wen Chen, R. Romero, and S. Draghici. Machine learning and its applications to biology. *PLoS Computational Biology*, 3(6):e116, 2007.
- [51] The Gene Ontology Consortium. The gene ontology: enhancements for 2011. *Nucleic Acids Research*, 40(D1):D559–D564, Nov. 2011.
- [52] The UniProt Consortium. Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Research*, 40(D1):D71–D75, Nov. 2011.
- [53] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.
- [54] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.
- [55] S. Velankar, Y. Alhroub, C. Best, S. Caboche, M. J. Conroy, J. M. Dana, M. A. Fernandez Montecelo, G. van Ginkel, A. Golovin, S. P. Gore, A. Gutmanas, P. Haslam, P. M. S. Hendrickx, E. Heuson, M. Hirshberg, M. John, I. Lagerstedt, S. Mir, L. E. Newman, T. J. Oldfield, A. Patwardhan, L. Rinaldi, G. Sahni, E. Sanz-García, S. Sen, R. Slowley, A. Suarez-Uruena, G. J. Swaminathan, M. F. Symmons, W. F. Vranken, M. Wainwright, and G. J. Kleywegt. PDBe: protein data bank in europe. *Nucleic Acids Research*, 40(D1):D445–D452, Jan. 2012. PMID: 22110033 PMCID: PMC3245096.
- [56] G. I. Webb, J. R. Boughton, and Z. Wang. Not So Naive Bayes: Aggregating One-Dependence Estimators. *Machine Learning*, 58(1):5–24, 2005.
- [57] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, third edition, 2011.
- [58] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1):69–90, 1999.
- [59] H. Yu, P. Braun, M. A. Yildirim, I. Lemmens, K. Venkatesan, J. Sahalie, T. Hirozane-Kishikawa, F. Gebreab, N. Li, N. Simonis, T. Hao, J.-F. Rual, A. Dricot, A. Vazquez, R. R. Murray, C. Simon, L. Tardivo, S. Tam, N. Svrzikapa, C. Fan, A.-S. de Smet, A. Motyl, M. E. Hudson, J. Park, X. Xin, M. E. Cusick, T. Moore, C. Boone, M. Snyder, F. P. Roth, A.-L. Barabasi, J. Tavernier, D. E. Hill, and M. Vidal. High-quality binary protein interaction map of the Yeast interactome network. *Science*, 322(5898):104–110, October 03 2008.
- [60] J. Zhang, J. Baran, A. Cros, J. M. Guberman, S. Haider, J. Hsu, Y. Liang, E. Rivkin, J. Wang, B. Whitty, M. Wong-Erasmus, L. Yao, and A. Kasprzyk. International cancer genome consortium data portal—a one-stop shop for cancer genomics data. *Database*, 2011(0):bar026–bar026, Sept. 2011.

- [61] J. Zhang, S. Haider, J. Baran, A. Cros, J. M. Guberman, J. Hsu, Y. Liang, L. Yao, and A. Kasprzyk. BioMart: a data federation framework for large collaborative projects. *Database*, 2011(0):bar038–bar038, Sept. 2011.

## Capítulo 6

# Computación paralela y clústeres de cálculo

*Álvaro Sebastián y José María Fernández*

Ya desde la década de 1950 con los primeros ordenadores comerciales, los ingenieros de IBM intentaron crear un “supercomputador” que fuera al menos 100 veces más rápido que el IBM 704<sup>1</sup>, con lo que tendría que tener una velocidad de cálculo de al menos 4 MIPS<sup>2</sup>. Aunque aquel proyecto fue un fracaso y no consiguieron más que 1,2 MIPS y una mejora de alrededor de 30 veces con respecto al IBM 704 a un precio de 7,78 millones de dólares, la investigación y desarrollo de aquel ordenador supusieron las bases de los actuales microprocesadores y varias de las técnicas de procesamiento en paralelo. Actualmente nuestros *smartphones* llevan procesadores que superan fácilmente los 2000 MIPS, siendo desde hace un par de años común los sistemas de 2, 4 y 8 núcleos, y cuyo precio es de unos pocos euros.

Las principales razones para el uso de máquinas de cómputo paralelo son: disminuir el tiempo total de ejecución de una aplicación, conseguir resolver problemas más complejos, de grandes dimensiones y permitir la ejecución simultánea de tareas concurrentes. Existen además motivos económicos y que se han alcanzado unos límites físicos muy difíciles de mejorar tecnológicamente en transmisión de datos y velocidad de CPU de los procesadores actuales. Por ello, la única forma de seguir aumentando el rendimiento de los ordenadores es mediante el cambio del paradigma de computación secuencial al paralelo, con múltiples procesadores, núcleos e hilos (ver Subsección 6.2.1). La *ley de Moore* postula que aproximadamente cada dos años se duplica el número de transistores que se pueden meter en un circuito integrado de tamaño estándar. Esta ley es empírica, formulada por el cofundador de Intel, Gordon E. Moore, el 19 de abril de 1965, cuyo cumplimiento se ha podido constatar hasta hoy [?]. Las habilidades de muchos dispositivos electrónicos digitales están ligadas a esta ley: velocidad y capacidad de procesamiento, capacidad de memoria, resolución de sensores, etc... Como se puede comprobar en la Figura 6.1, la ley de Moore ligada a la capacidad de procesamiento se mantiene vigente gracias a la *tecnología multinúcleo* (múltiples núcleos en un único procesador), dado que desde al año 2006 existen muy pocos avances respecto a la velocidad de las CPUs.

Aunque desde hace varias décadas existe la posibilidad de usar múltiples ordenadores y procesadores para los cálculos, el paralelismo está ganando un gran interés en estos últimos años debido al desarrollo de ordenadores personales *multiprocesador* y *multinúcleo*, y al desarrollo de plataformas de cálculo

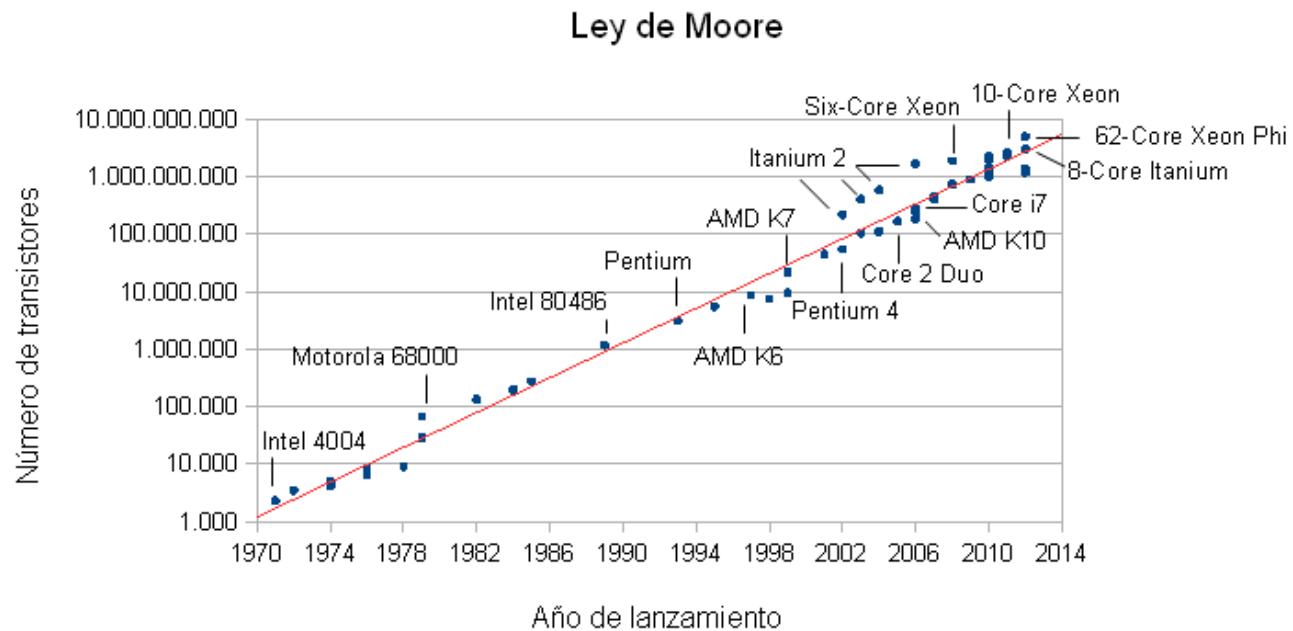
---

<sup>1</sup>Proyecto IBM 7030. [http://es.wikipedia.org/wiki/IBM\\_7030](http://es.wikipedia.org/wiki/IBM_7030)

<sup>2</sup>Millones de instrucciones por segundo

basadas en los *chips GPU* (acrónimo del inglés *graphics processing unit*). La *tecnología GPU* permite usar los miles de unidades de computación de una simple tarjeta gráfica para cálculos aritméticos complejos. Sin embargo dicha tecnología está limitada por la memoria que puede utilizar cada unidad de computación, que suele estar ligada a la arquitectura y a la memoria de la tarjeta gráfica. A pesar del rápido desarrollo de nuevas tecnologías, los *clústeres* de cálculo siguen siendo la opción preferida por la mayoría de los científicos para el procesado de datos. Un *clúster* es un grupo de múltiples ordenadores que son usados como un único ordenador, con las ventajas en términos de potencia y velocidad que ello conlleva (ver Sección 6.2).

La computación paralela es especialmente importante en el campo de la bioinformática debido a que permite mejorar la velocidad en el proceso de ingentes cantidades de datos (ej. secuenciación masiva) o algoritmos de gran complejidad (ej. dinámica molecular, Capítulo 17) para conseguir resultados y conclusiones en un tiempo razonable. En el presente capítulo se presentarán las posibilidades de la computación paralela y cómo hacer uso de los clústeres o supercomputadores presentes en los laboratorios de investigación. No se entrará en profundidad en detalles técnicos, si se desea un mayor conocimiento del tema existen libros especializados como el de Francisco Almeida et al. [? ].



**Figura 6.1:** Ley de Moore o evolución del número de transistores en los procesadores de ordenador.

## 6.1. Computación paralela y ejemplos de programación

La *computación paralela*<sup>3</sup> es una forma de cómputo en la que se ejecutan simultáneamente varias instrucciones dirigidas a resolver un problema común. Cualquier problema resoluble a nivel informático está descrito por su flujo de ejecución, en el cuál normalmente hay partes independientes entre sí que pueden ser susceptibles de ser ejecutadas en paralelo. Por tanto, en la computación paralela el objetivo

<sup>3</sup>No hay que confundir la computación paralela con la programación concurrente, a pesar de estar íntimamente relacionadas.

es intentar dividir problemas grandes en una serie de problemas más pequeños interdependientes entre sí, que permitan luego ser resueltos simultáneamente (en paralelo) cuando es factible [? ].

Por ello, los programas informáticos para computación paralela son más difíciles de escribir que los secuenciales puesto que deben tener en cuenta la coordinación en la ejecución de múltiples subtareas de una forma eficiente y la agrupación coherente de sus múltiples resultados. Dependiendo de la tarea que se tenga que realizar, las múltiples tareas de un programa paralelo deben intercambiar información entre sí y comunicarse de alguna manera. Atendiendo a la estrategia de comunicación entre tareas, podemos clasificar los sistemas informáticos dedicados a la programación paralela en dos modelos:

- *Memoria compartida*: los múltiples procesos o *hilos* (del inglés, *threads*) se ejecutan en núcleos de procesamiento donde todos tienen acceso a áreas de memoria comunes, intercambiando datos e información a través de esta memoria. Es el paradigma de paralelismo preferido para computadoras multiprocesador y sistemas de computación en GPU. Existen estándares de programación en memoria compartida, siendo los más extendidos OpenMP<sup>4</sup> y *pthreads*.
- *Memoria distribuida*: no todos los núcleos de procesamiento participantes en la computación tienen acceso a un área de memoria compartida común, pero sí todos tienen un medio de intercambiar datos mediante *paso de mensajes*. Por ello, para intercambiar datos han de realizar peticiones explícitas entre núcleos de procesamiento y recibir respuestas con los datos, usando habitualmente *Message Passing Interface* o MPI. Suele ser el sistema preferido para granjas de computación, clústeres o sistemas multicamputador. El estándar MPI<sup>5</sup> está disponible como una serie de librerías para varios lenguajes de programación, como por ejemplo C, C++ y Fortran.

Cuando nos enfrentamos a un problema a paralelizar, tenemos que tener en cuenta los distintos tipos de paralelismo que podemos aplicar en los distintos sub-bloques del mismo.

- *Paralelismo de grano grueso*: Consiste en ejecutar en paralelo grandes tareas secuenciales. Este tipo de paralelismo es bastante común en muchos problemas bioinformáticos, como por ejemplo, predecir la estructura tridimensional o anotar funcionalmente todas las proteínas de un organismo. En este ejemplo, hay que aplicar las mismas tareas de minería de datos, extrapolación y predicción sobre cada una de las proteínas, y los resultados para una proteína no dependen de los resultados para otra.
- *Paralelismo de grano fino*: Consiste en ejecutar en paralelo bloques de código cuya duración no es excesivamente larga, pero que por su número dentro de la tarea madre tarda mucho en terminar. Este tipo de paralelismo suele buscarse en problemas ya expresados de forma secuencial que son realizados muchísimas veces, como por ejemplo una búsqueda de motivos de proteína, un alineamiento múltiple de secuencias, cálculo de árboles filogenéticos, uso de algoritmos de *clustering* o una simple búsqueda por homología.

Tomando este último ejemplo, si queremos buscar los homólogos de muchas secuencias en una base de datos de secuencias, tenemos varias opciones de paralelización. Sin tener en cuenta la longitud de las secuencias para las que queremos buscar sus homólogos, podemos o dividir la base de datos de secuencias en tantos trozos como núcleos de computación asignados a la tarea, o si el conjunto de secuencias a buscar es lo suficientemente grande, dividir ese conjunto en varios más pequeños. Si las secuencias a buscar son extremadamente largas, incluso podríamos pensar en dividir las secuencias más largas en trozos más pequeños, y usar una versión modificada del algoritmo de búsqueda por homología.

---

<sup>4</sup>OpenMP. <http://www.openmp.org>

<sup>5</sup>Message Passing Interface Forum. <http://www.mpi-forum.org>

Ambos tipos de paralelismo no son incompatibles, pudiendo encontrarse muy a menudo ambos en flujos de trabajo complejos, como por ejemplo casi todos los actualmente desarrollados para *Next Generation Sequencing*. Un paradigma de programación que está siendo ampliamente usado en los últimos años para procesar enormes conjuntos de datos de manera paralela y distribuida es *MapReduce*[? ]

Todos los lenguajes de programación disponen de módulos o librerías que nos permiten aplicar estrategias de programación paralela en nuestros programas y *scripts*. Para los lenguajes de programación Perl y Python los módulos más usados son ‘*threads*’ en Perl, y ‘*multiprocessing*’ y ‘*subprocess*’ en Python.

### 6.1.1. Paralelización en Perl (**threads**)

A partir de Perl 5.8 se recomienda el módulo ‘*threads*’ para escribir código paralelo. Dicho módulo permite crear diferentes hilos de ejecución en nuestros *scripts*. Sin embargo, hay que tener en cuenta que por defecto todas las variables que crea son independientes entre hilos (memoria distribuida), con lo que si se quieren usar variables compartidas hay que recurrir al módulo adicional *threads::shared*.

Ambos módulos deberán ser instalados por el administrador del sistema si no existen previamente:

```
$ sudo cpan -i threads threads::shared
```

Para usarlos hay que invocarlos al comienzo de nuestro *script*:

```
use threads;
use threads::shared;
```

y posteriormente indicar las variables que van a ser compartidas:

```
my ($scalar, @array, %hash) :shared;
```

O:

```
my ($scalar, @array, %hash);
share($scalar);
share(@array);
share(%hash);
```

Dichas variables sólo podrán almacenar escalares o referencias a otras variables compartidas.

Los hilos de ejecución se han de crear especificando una subrutina a ejecutar junto con sus argumentos:

```
my $thread1 = threads->create(\&subroutine, $argument);
my $thread2 = threads->create(\&subroutine, $argument);
```

y el comando ‘*join*’ espera a que el hilo se ejecute y recoge los resultados:

```
$result1 = $thread1->join();
$result2 = $thread2->join();
```

Existen muchas más opciones del módulo *threads* que se pueden consultar en su *manual online*<sup>6</sup>. Otra opción equivalente es *forks*<sup>7</sup> junto con *forks::shared*.

A continuación expondremos un simple ejemplo de uso de hilos con Perl para contar los 1000000000 primeros números naturales. En el Código 6.1 se calcula de forma secuencial (no paralela), sumando de 1 en 1. El Código 6.2 realiza el mismo cálculo usando 4 hilos paralelos de ejecución. En el último ejemplo el cálculo se lleva a cabo mediante un bucle de 10000 repeticiones que ejecuta cada vez 4 hilos (40000 cálculos). He aquí los resultados:

<sup>6</sup>Manual de Perl *threads*. <http://perldoc.perl.org/threads.html>

<sup>7</sup>Módulo Perl *forks*. <http://search.cpan.org/~rybskej/forks/lib/forks.pm>

Tiempo empleado por el cálculo de forma secuencial: 49 segundos

Tiempo empleado por el cálculo paralelo con 4 hilos: 13 segundos

Tiempo empleado por 40000 cálculos en 4 hilos paralelos: 88 segundos

Los 3 *scripts* realizan el mismo número de operaciones de cálculo, pero el Código 6.2 es casi 4 veces más rápido que el Código 6.1 (en un ordenador actual multinúcleo), lo cual demuestra la *eficiencia de la paralelización*. Sin embargo el Código 6.3 es sensiblemente más lento que los otros dos a pesar de realizar también cálculos paralelos. La explicación es muy sencilla, el coste de lanzar los miles de hilos y recoger sus resultados no se compensa con el tiempo de cálculo de cada uno. Como conclusión, el código paralelo es muy útil y eficaz, pero un uso incorrecto puede hacerlo más lento que el código secuencial y deberemos probar su eficacia con módulos como ‘Benchmark’ antes de realizar los cálculos en nuestro ordenador o clúster.

Código 6.1: Cálculo de un número de forma secuencial con Perl.

```
1 # Importa los módulos necesarios
2 use Benchmark;
3
4 # Define el número a calcular
5 my $number = 1000000000;
6
7 # Función que cuenta hasta el número especificado
8 sub calc {
9     my ($number) = @_;
10    my $i = 0;
11    while ($i<$number) {
12        $i++;
13    }
14    return $i;
15 }
16
17 # Comienza el cronómetro
18 my $start = new Benchmark;
19
20 # Realiza el cálculo de forma secuencial (no paralela)
21 my $total = calc($number);
22
23 # Para el cronómetro
24 my $end = new Benchmark;
25
26 # Calcula el tiempo total invertido
27 my $diff = timediff($end, $start);
28
29 # Imprime el resultado
30 print "Total = $total\n";
31
32 # Muestra el tiempo empleado
33 print "Tiempo empleado por el cálculo de forma secuencial: ", timestr($diff, 'all'), "\n"
";
```

Código 6.2: Cálculo de un número en Perl con 4 hilos de ejecución.

```
1 # Importa los módulos necesarios
2 use threads;
3 use threads::shared;
4 use Benchmark;
5 use Config;
6
```

```

7 # Define el número a calcular
8 my $number = 1000000000;
9
10 # Crea una variable compartida por los hilos
11 my $total = 0;
12 share($total);
13
14 # Función que cuenta hasta el número especificado
15 sub calc {
16     my ($number) = @_;
17     my $i = 0;
18     while ($i<$number) {
19         $i++;
20     }
21     $total += $i;
22 }
23
24 # Comienza el cronómetro
25 my $start = new Benchmark;
26
27 # Crea 4 hilos de Perl que ejecutan simultáneamente los cálculos
28 my $thr1 = threads->create(\&calc, $number/4);
29 my $thr2 = threads->create(\&calc, $number/4);
30 my $thr3 = threads->create(\&calc, $number/4);
31 my $thr4 = threads->create(\&calc, $number/4);
32
33 # Comprueba la finalización de los hilos y recoge los resultados
34 my $res1 = $thr1->join();
35 my $res2 = $thr2->join();
36 my $res3 = $thr3->join();
37 my $res4 = $thr4->join();
38
39 # Para el cronómetro
40 my $end = new Benchmark;
41
42 # Calcula el tiempo total invertido
43 my $diff = timediff($end, $start);
44
45 # Imprime el resultado
46 print "Total = $total\n";
47
48 # Muestra el tiempo empleado
49 print "Tiempo empleado por el cálculo paralelo con 4 hilos: ", timestr($diff, 'all'), "\n";

```

Código 6.3: Cálculo de un número en Perl con un bucle de 10000 repeticiones con 4 hilos de ejecución.

```

1 # Importa los módulos necesarios
2 use threads;
3 use threads::shared;
4 use Benchmark;
5 use Config;
6
7 # Define el número a calcular
8 my $number = 1000000000;
9
10 # Crea una variable compartida por los hilos
11 my $total = 0;
12 share($total);
13
14 # Función que cuenta hasta el número especificado

```

```

15 sub calc {
16     my ($number) = @_;
17     my $i = 0;
18     while ($i<$number) {
19         $i++;
20     }
21     $total += $i;
22 }
23
24 # Comienza el cronómetro
25 my $start = new Benchmark;
26
27 # Divide el proceso en 40000 cálculos más pequeños
28 # con 4 hilos de Perl que ejecutan simultáneamente los cálculos
29 my $div = 10000;
30 for (my $i=0; $i<$div; $i++) {
31     my $thr1 = threads->create(\&calc, $number/(4*$div));
32     my $thr2 = threads->create(\&calc, $number/(4*$div));
33     my $thr3 = threads->create(\&calc, $number/(4*$div));
34     my $thr4 = threads->create(\&calc, $number/(4*$div));
35     my $res1 = $thr1->join();
36     my $res2 = $thr2->join();
37     my $res3 = $thr3->join();
38     my $res4 = $thr4->join();
39 }
40
41 # Para el cronómetro
42 my $end = new Benchmark;
43
44 # Calcula el tiempo total invertido
45 my $diff = timediff($end, $start);
46
47 # Imprime el resultado
48 print "Total = $total\n";
49
50 # Muestra el tiempo empleado
51 print "Tiempo empleado por $div cálculos con 4 hilos paralelos: ", timestr($diff, 'all')
      , "\n";

```

### 6.1.2. Paralelización en Python (**multiprocessing** y **subprocess**)

En el lenguaje Python es el módulo ‘multiprocessing’ el que permite dividir tareas en múltiples procesos paralelos y ganar eficiencia. Existe también el módulo ‘threading’ en Python, sin embargo su mayor dificultad de uso y similar rendimiento hace que ‘multiprocessing’ sea el favorito de los programadores. Desafortunadamente el módulo multiprocessing sólo permite parallelizar código de Python, no permite ejecutar y controlar en paralelo programas externos, por ello también explicaremos el uso de ‘subprocess’ que sí nos da esa opción.

Ambos módulos requieren su instalación por el administrador del sistema, en Ubuntu:

```
$ sudo apt-get install python-multiprocessing python-subprocess
```

Comenzaremos por explicar el funcionamiento de ‘multiprocessing’, para usarlo hay que importarlo al comienzo de nuestro *script*:

```
import multiprocessing
```

Como en Perl, hay que especificar el nombre y tipo de las variables que van a ser compartidas:

```
scalar = multiprocessing.Value('d', 0.0)
array = multiprocessing.Array('i', range(10))
```

Los procesos de ejecución se han de crear especificando una función a ejecutar junto con sus argumentos:

```
proc1 = multiprocessing.Process(target=function, args=(scalar, array))
proc2 = multiprocessing.Process(target=function, args=(scalar, array))
```

y con los comandos ‘start’ y ‘join’ ejecutaremos los procesos y recogeremos los resultados:

```
proc1.start()
proc2.start()
proc1.join()
proc2.join()
```

Los resultados serán almacenados en las variables compartidas.

Existen más opciones del módulo `multiprocessing` que se pueden consultar en su *manual online*<sup>8</sup>.

Reescribiendo el Código 6.2 en Python con el módulo ‘`multiprocessing`’ obtenemos el Código 6.4. Si comparamos su eficiencia con sus códigos homólogos, secuencial y mal paralelizado (equivalentes a Código 6.1 y Código 6.3 en Perl), los resultados son:

```
Tiempo empleado por el cálculo de forma secuencial: 44 segundos
```

```
Tiempo empleado por el cálculo paralelo con 4 procesos: 19 segundos
```

```
Tiempo empleado por 40000 cálculos en 4 procesos paralelos: 74 segundos
```

No entraremos a valorar qué lenguaje es más rápido y eficiente, pero volvemos a observar que el cálculo paralelo gana en eficiencia al secuencial en aquellos problemas que tienen un alto grado de paralelismo. La creación de procesos en Python también tiene su coste y por ello se vuelve a demostrar la ineficiencia de generar miles de procesos para un cálculo sencillo.

Código 6.4: Cálculo de un número en Python con 4 procesos simultáneos.

```
1 # -*- encoding: utf-8 -*-
2 # Importa los módulos necesarios
3 import multiprocessing
4 from time import time
5
6 # Define el número a calcular
7 number = 1000000000
8
9 # Crea una variable compartida por los hilos
10 total = multiprocessing.Value('i', 0)
11
12 # Cuenta hasta el número especificado
13 def calc(number):
14     i = 0
15     for i in range(number):
16         i += 1
17     total.value += i
18
19 # Comienza el cronómetro
20 start = time()
21
22 # Crea 4 procesos que ejecutan simultáneamente los cálculos
23 proc1 = multiprocessing.Process(target=calc, args=(number/4,))
24 proc2 = multiprocessing.Process(target=calc, args=(number/4,))
```

<sup>8</sup>Manual de Python `multiprocessing`. <https://docs.python.org/library/multiprocessing.html>

```

25 proc3 = multiprocessing.Process(target=calc, args=(number/4,))
26 proc4 = multiprocessing.Process(target=calc, args=(number/4,))
27
28 # Lanza los procesos
29 proc1.start()
30 proc2.start()
31 proc3.start()
32 proc4.start()
33
34 # Recoge los resultados
35 proc1.join()
36 proc2.join()
37 proc3.join()
38 proc4.join()
39
40 # Para el cronómetro
41 end = time()
42
43 # Imprime resultado
44 print 'Total = ', total.value
45
46 # Muestra el tiempo empleado
47 print 'Tiempo empleado por el cálculo paralelo con 4 procesos:', end - start, 'segundos'

```

## Colas de procesos paralelos

Python nos ofrece además la posibilidad de gestionar automáticamente la ejecución de procesos, de forma que en todo momento se estén ejecutando el número de procesos que nosotros decidamos, ni más ni menos. Esta gestión de una cola o *pool* de procesos es el equivalente en un único ordenador a un ‘sistema gestor de colas’ en un clúster (Sección 6.5). La *gestión de colas* es muy eficiente en Python y el resultado de ejecutar 40000 procesos de 4 en 4 (Código 6.5, equivalente al Código 6.3) es más rápido que el cálculo paralelo con 4 procesos sencillos (ver Código 6.4):

Tiempo empleado por 40000 cálculos paralelos en cola de procesos: 14 segundos

Tiempo empleado por el cálculo paralelo con 4 procesos: 19 segundos

Código 6.5: Cálculo de un número con colas de procesos en Python.

```

1 # -*- encoding: utf-8 -*-
2 # Importa los módulos necesarios
3 import multiprocessing
4 from time import time
5
6 # Define el número a calcular
7 number = 1000000000
8
9 # Crea una variable compartida por los hilos
10 total = multiprocessing.Value('i', 0)
11
12 # Cuenta hasta el número especificado
13 def calc(number):
14     i = 0
15     for i in range(number):
16         i += 1
17     total.value += i
18
19 # Comienza el cronómetro

```

```

20 start = time()
21
22 # Define el número de procesos a ejecutar simultáneamente
23 pool = multiprocessing.Pool(processes=4)
24
25 # Lanza 40000 procesos que serán enviados a la cola y ejecutados de 4 en 4
26 div = 40000
27 for p in range(div):
28     pool.apply_async(calc, (number/div,))
29
30 # Lanza los procesos
31 pool.close()
32
33 # Recoge los resultados
34 pool.join()
35
36 # Para el cronómetro
37 end = time()
38
39 # Imprime resultado
40 print 'Total = ', total.value
41
42 # Muestra el tiempo empleado
43 print 'Tiempo empleado por 40000 cálculos paralelos con cola de procesos:', end - start,
        'segundos'

```

## Ejecución de programas externos en paralelo

Para finalizar veremos un ejemplo de uso de ‘`subprocess`<sup>9</sup>’ que permite *ejecutar en paralelo programas externos*, como por ejemplo BLAST (aunque quizás no sea el mejor ejemplo porque posee de forma nativa la opción ‘`-num_threads`’ para ello). En el Código 6.6 ejecutaremos en paralelo 4 comandos que imprimen en pantalla la versión que tenemos instalada de varios programas del paquete de herramientas BLAST+<sup>10</sup>. Se trata de un ejemplo sencillo que permite ver la utilidad de este módulo de Python a la hora de ejecutar múltiples comandos de forma paralela en un ordenador o clúster.

Código 6.6: Ejemplo de uso del módulo de Python `subprocess`.

```

1 # -*- encoding: utf-8 -*-
2 # Importa los módulos necesarios
3 import subprocess
4
5 # Define los comandos a ejecutar
6 commands = []
7 commands.append('blastp -version')
8 commands.append('blastn -version')
9 commands.append('psiblast -version')
10 commands.append('tblastx -version')
11
12 # Crea 4 procesos que ejecutan simultáneamente los comandos
13 processes = []
14 for i in range(len(commands)):
15     processes.append(subprocess.Popen(commands[i], shell=True, stdout=subprocess.PIPE,
                                         stderr=subprocess.PIPE))
16
17 # Comprueba la finalización de los procesos y recoge los resultados

```

<sup>9</sup>Manual de Python `subprocess`. <https://docs.python.org/library/subprocess.html>

<sup>10</sup>BLAST Command Line Applications User Manual. <http://www.ncbi.nlm.nih.gov/books/NBK1763>

```

18 results = []
19 for p in processes:
20     p.wait()
21     results.append(''.join(p.stdout))
22
23 # Muestra la salida de los comandos ejecutados
24 for i in range(len(commands)):
25     print 'Comando:', commands[i]
26     print 'Resultado:', results[i]

```

## 6.2. Clústeres de cálculo

Un *clúster de cálculo* o *granja de computación* es un conjunto físico (montados en una misma cabina) o virtual (situados en diferentes localizaciones) de ordenadores que poseen un sistema de *hardware* y/o *software* que coordina los trabajos de computación que realizan. Generalmente todos los miembros del clúster usan el mismo *software* y pueden tener diferentes configuraciones de *hardware*, lo que permite mejorar el mismo con la tecnología más actual sin tener que desechar los procesadores más antiguos pero igualmente válidos. Existen también clústeres con diferentes configuraciones de *hardware* y *software*, generalmente virtuales.

Los clústeres se pueden clasificar en tres categorías de acuerdo a sus características técnicas y aplicaciones:

- *Alto rendimiento (High Performance Computing)*: están preparados para ejecutar tareas que requieren de gran capacidad de cálculo y grandes cantidades de memoria, por ejemplo el ensamblaje de genomas.
- *Alta eficiencia (High Throughput Computing)*: su objetivo es ejecutar la mayor cantidad de tareas en el menor tiempo posible. Las tarjetas gráficas son el mejor ejemplo, permiten realizar miles de millones de operaciones matemáticas sencillas en milésimas de segundo. Suelen estar limitados por la memoria que pueden emplear para cada proceso, pero son especialmente interesantes para ciertos cálculos en investigación, por ejemplo en mecánica cuántica o dinámica molecular (Capítulo 17) y cribado molecular (Capítulo 17).
- *Alta disponibilidad (High Availability Computing)*: garantizan la máxima disponibilidad y estabilidad de los servicios que ofrecen. Para ello emplean los recursos de forma redundante, en detrimento de la eficiencia, por ejemplo si falla un ordenador del clúster, el resto siguen funcionando. Son generalmente usados por empresas.

Los *clústeres de alto rendimiento* son los clústeres más habitualmente empleados en investigación, por ello usaremos la palabra “*clúster*” a lo largo del capítulo para referirnos a ellos (Figura 6.2). Son también los clústeres más costosos y pueden disponer desde decenas a millones de procesadores y *terabytes* de memoria. Sin embargo, para su correcto aprovechamiento, los usuarios deben tener unos mínimos conocimientos del mismo y emplear los recursos adecuados para sus necesidades.

Para que un clúster funcione como tal, no basta sólo con conectar entre sí los ordenadores. Es necesario proveer un sistema de manejo del clúster que coordine sus componentes, que se encargue además de actuar como intermediario entre el usuario y los procesos que corren en los ordenadores integrantes del clúster para optimizar el uso de los mismos.



**Figura 6.2:** Aspecto típico de un armario de clúster.

### 6.2.1. Nodos, memoria, procesadores, núcleos, procesos e hilos

Es importante conocer la estructura de nuestro clúster. En primer lugar deberemos conocer el número de nodos del mismo. Un *nodo* es cada una de las máquinas independientes que conforman el clúster. Pueden ser computadores en formato de sobremesa, torre, *rack* (Figura 6.3) u otro. En teoría, si un nodo se estropea, podríamos reemplazarlo sin alterar al resto, aunque en algunas configuraciones o casos especiales puede ser necesario apagar temporalmente todo el clúster. Quedaría puntualizar que un clúster no necesita teclado, ratón y pantalla (sólo para su instalación inicial) puesto que se administra y utiliza de forma remota, habitualmente mediante un terminal de texto en cualquier ordenador conectado a la red.

El *sistema de almacenamiento de archivos* puede ser interno, mediante discos conectados directamente a uno o varios nodos del clúster, o mediante el acceso a un *servidor de almacenamiento externo*. Los servidores externos más utilizados son los de tipo NAS (*Network Attached Storage*) y el protocolo NFS (*Network File System*), que permiten compartir ficheros entre los nodos del clúster y otros ordenadores de la red.

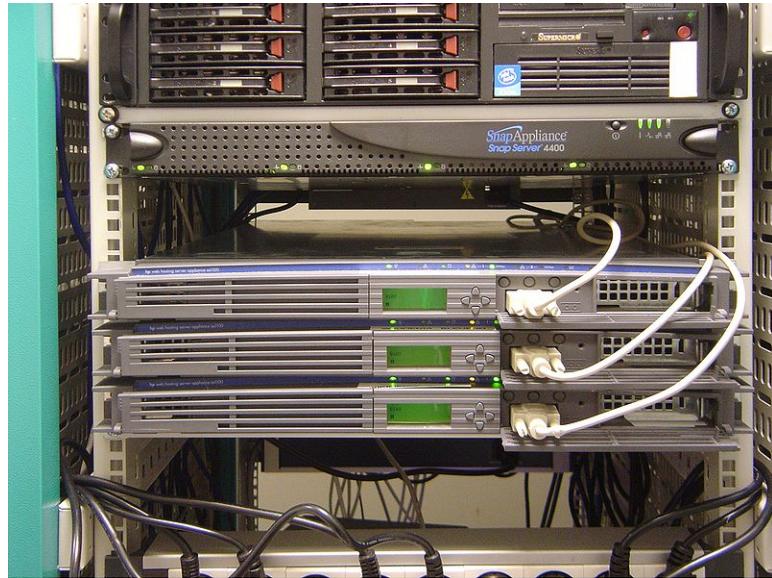
Cada nodo poseerá uno o más procesadores. Los *procesadores multi-núcleo* actuales permiten incorporar varias unidades centrales de procesamiento independientes llamadas *núcleos* o *cores*. Cada núcleo tiene en la práctica el mismo comportamiento que un procesador individual lo que permite paralelizar el trabajo de procesamiento dentro del nodo. Y a su vez, cada núcleo permite la ejecución de varios *subprocesos*, *hilos* o *threads*, lo cual permite dividir un *proceso* en múltiples hilos y así aumentar el rendimiento de cálculo.

Finalmente, cada nodo posee su propia *memoria* (RAM, *Random-Access Memory*). Dicha memoria es compartida entre los diferentes núcleos de procesamiento del nodo.

Es importante conocer cuántos nodos y cuántos núcleos y memoria por nodo tiene el clúster a utilizar. Cada nodo se comporta como una computadora individual que tendrá que repartir sus recursos de memoria entre sus núcleos. Por ejemplo, un clúster con 20 nodos de 40 núcleos y 60 GB de memoria por nodo tendrá en total 800 núcleos y 1.2 TB de memoria. Cada núcleo podrá usar como máximo los

60 GB de memoria de su nodo, pero si los 40 núcleos del nodo están trabajando simultáneamente, cada núcleo dispondrá de media de 1.5 GB. Para aumentar la complejidad, los clústeres no son homogéneos y puede haber nodos con diferente número de núcleos y tamaño de memoria.

A lo largo del capítulo nos referiremos con la palabra ‘proceso’ a cada tarea, programa, *script* o sesión interactiva que ejecutaremos en un clúster. En la Subsección 6.6.6 se explicará cómo asignar límites de memoria a nuestros procesos y en la Subsección 6.6.7 cómo asignar múltiples núcleos.



**Figura 6.3:** Varios servidores de un clúster en formato *rack*.

### 6.2.2. Rendimiento

Para medir la potencia de un ordenador para cálculo científico se suele contabilizar el *número de operaciones aritméticas* (por ejemplo de coma flotante, o sea, de números decimales) que realiza por segundo. Cuantas más operaciones pueda realizar en un segundo, más potente será y más rápido se ejecutarán las aplicaciones (obviando que puede haber otros factores que limiten la ejecución de la aplicación como los accesos a memoria). Se definen como *FLOPS* de un procesador a la cantidad de operaciones de coma flotante que realiza en un segundo (del inglés *Floating point Operations Per Second*). Como habitualmente es un valor muy alto, usaremos las abreviaturas habituales como *Giga* (G) y equivale a 1000 millones de FLOPS. Un procesador actual con dos núcleos tiene una potencia de unos 50 GFLOPS y una tarjeta gráfica normal de 128 núcleos puede alcanzar los 500 GFLOPS. Sin embargo, aunque la tarjeta gráfica posea 10 veces más de capacidad de cálculo, la memoria disponible en la tarjeta ha de ser dividida entre los 128 núcleos lo que limita mucho el tipo de cálculos que podemos realizar con ella. Un clúster con 20 nodos de 20 procesadores como el anterior (50 GFLOPS) tendrá una potencia de 20000 GFLOPS o 20 Tera-FLOPS ( $20 \times 20 \times 50$ ). En el sitio web TOP500 List<sup>11</sup> podemos consultar los supercomputadores más potentes del momento. En noviembre de 2013 el clúster número 1 era ‘Tianhe-2’ del National Super Computer Center de Guangzhou, con más de 3 millones de núcleos y una potencia de 33,86 Peta-FLOPS ( $\times 1000000$  GFLOPS) (Figura 6.4).

Otras magnitudes y medidas del rendimiento son [?]:

<sup>11</sup>TOP500 List. <http://www.top500.org>



**Figura 6.4:** Supercomputador Tianhe-2 (Fuente: TOP500 List<sup>11</sup>).

- *Tiempo de ejecución* (o respuesta) de un proceso ( $T$ ).
- *Productividad (throughput)*: número de procesos que es capaz de procesar por unidad de tiempo ( $1/T$ ).
- *Eficiencia*: es la porción útil de trabajo total realizado por  $n$  procesadores, se calcula como el cociente entre el tiempo empleado en realizar un proceso por un único procesador entre el producto del tiempo empleado por el clúster y el número de procesadores:

$$E(n) = \frac{T(1)}{nT(n)}$$

- *Factor de mejora del rendimiento (speed-up)*: mide el grado de ganancia de velocidad, es el cociente entre el tiempo empleado en realizar un proceso por un único procesador entre el tiempo empleado por el clúster:

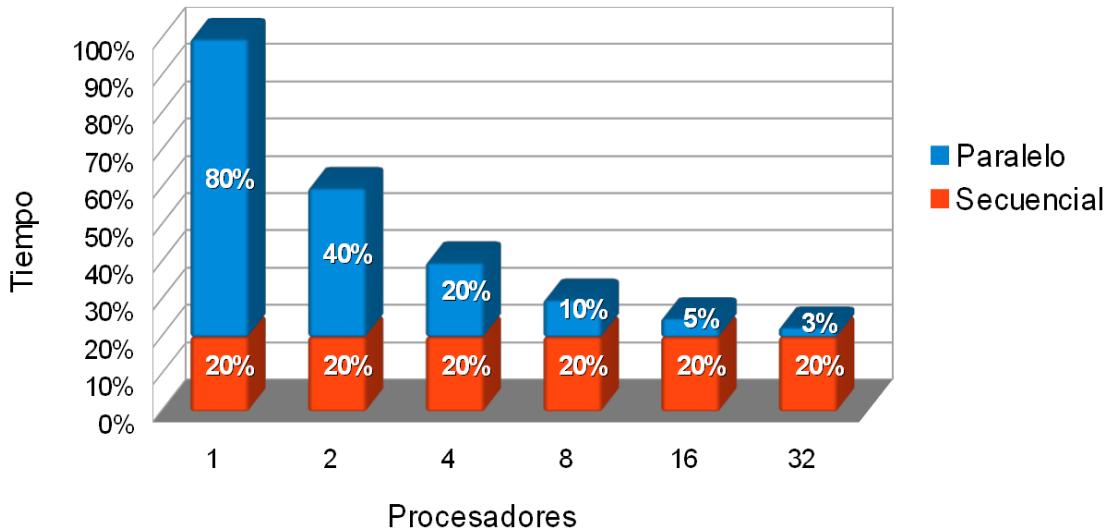
$$S(n) = \frac{T(1)}{T(n)} = nE(n)$$

Normalmente todos los sistemas y programas tienen un determinado número de procesadores a partir del cuál la eficiencia empieza a disminuir de forma más o menos brusca. Veamos ahora el problema típico que nos mostrará lo importante que es que la parte paralelizable de un programa sea lo mayor posible si queremos obtener el máximo rendimiento a un sistema paralelo

Sea un programa que posee un tiempo de ejecución de 100 unidades de tiempo (por ejemplo segundos). El 80 % de su código es perfecta y absolutamente paralelizable. Calculemos el rendimiento y la eficiencia de este programa cuando se está ejecutando sobre 1, 2, 4, 8, 16 y 32 procesadores.

Procesadores	1	2	4	8	16	32
Eficiencia	1	100/(2x60)=0.83	0.62	0.42	0.25	0.14
Rendimiento	1	100/(40+20)=1.7	2.5	3.3	4.0	4.4

## Estudio del rendimiento



Como se puede observar el *rendimiento no crece al mismo ritmo que el número de procesadores*, y llega un momento en que incrementar el número de procesadores apenas mejora el rendimiento. Esto es debido a que la parte de código no paralelizada está actuando como limitante. Se consideran aceptables eficiencias superiores a 0.5, por debajo de este umbral es preferible mejorar la paralelización del *software* que incrementar el número de procesadores.

### 6.3. Conexión remota

La mayoría de los clústeres de computación usan el *protocolo de consola segura (Secure Shell)* para permitir conexiones desde computadores externos, por ejemplo en casa o en la oficina. Para ello debemos tener instalado un cliente SSH en el ordenador remoto y conectarnos al clúster mediante una consola con un intérprete de comandos (ver Sección 4.3).

#### 6.3.1. Protocolo SSH

*Secure Shell (SSH)* es un protocolo de red criptográfico que permite la transferencia de datos, inicio de sesión de usuario, ejecución de comandos y otros servicios seguros de red entre dos computadores, servidor (el clúster) y cliente (nuestro ordenador). Dicho protocolo sustituye a otros antiguos (*telnet*, *rlogin*, *rsh*) que enviaban información a través de la red sin encriptar. SSH utiliza un sistema de encriptación asimétrica basado en una clave pública y otra privada, ligadas entre sí, para realizar transferencias seguras de datos.

Para poder comunicarnos con el clúster vía SSH necesitaremos instalar un *cliente SSH* en nuestra computadora. Existen múltiples clientes, para sistemas Linux OpenSSH<sup>12</sup> es el más extendido y en

<sup>12</sup>OpenSSH. <http://www.openssh.com>

Windows el cliente *Putty*<sup>13</sup>. Instalar OpenSSH en un sistema Ubuntu es muy sencillo (otras distribuciones Linux tienen similares métodos de instalación):

```
$ sudo apt-get install openssh-client
```

El acceso al un clúster es a través de la conexión a alguno de sus nodos de acceso. Para ello, es necesario iniciar el cliente SSH indicando el nombre de usuario y la dirección IP o el nombre de dominio del servidor SSH (el clúster). También puede ser necesario indicar el puerto de conexión si es diferente al puerto por defecto 22:

```
$ ssh -p 2222 clusteruser@cluster.domain.com
```

La primera vez que nos conectemos, aparecerá un mensaje similar al siguiente, al que deberemos responder ‘yes’ y pulsar Enter:

```
The authenticity of host `cluster.domain.com (123.4.56.789)' can't be established.  
RSA key fingerprint is 3c:6d:5c:99:5d:b5:c6:25:5a:d3:78:8e:d2:f5:7a:01.  
Are you sure you want to continue connecting (yes/no) ?
```

Entonces pedirá la contraseña que nos haya facilitado el administrador del clúster. Si surgiera algún problema siguiendo las instrucciones anteriores, podemos intentar conectarnos en modo ‘verbose’ para buscar información sobre el error de conexión:

```
$ ssh -vvv -p 2222 clusteruser@cluster.domain.com
```

### 6.3.2. Nodo de acceso

Tras autenticarnos en el clúster, iniciaremos sesión remota con una línea de comandos similar a la siguiente:

```
[clusteruser@headnode ~ ]$
```

Lo cuál significa que hemos iniciado una sesión en el clúster con el usuario “clusteruser” y que nos encontramos en el *nodo de acceso (muchas veces el nodo principal) del clúster* (“headnode”). El nodo de acceso debe ser usado únicamente para conectarnos al clúster, y es muy importante saber que *no se deben ejecutar procesos en este nodo*. El nodo de acceso es el computador del clúster al cuál acceden todos los usuarios conectados mediante consolas remotas y debe ser únicamente utilizado para administrar archivos, editar texto o enviar procesos al resto de nodos. Normalmente si el administrador del clúster encuentra trabajos de larga duración en el nodo de acceso, los parará y eliminará sin notificarnos y perderemos los datos asociados. Entonces, ¿qué hay que hacer para ejecutar trabajos en un clúster? La respuesta se encontrará en la Sección 6.5. A continuación se explicará como transferir archivos entre nuestra máquina y el clúster.

## 6.4. Transferencia remota de archivos

Existen dos protocolos mayoritariamente usados para transferir archivos y carpetas entre computadores remotos, éstos son *SCP (Secure Copy Protocol)* / *SFTP (Secure File Transfer Protocol)* y *FTP (File Transfer Protocol)*. Existen algunas diferencias entre ambos:

Para poder usar cualquiera de los dos protocolos, éste debe estar habilitado en el clúster mediante un *sotware* servidor instalado y deberemos instalar en nuestra computadora un programa cliente (de forma similar al cliente SSH, ver Subsección 6.3.1). Un cliente gratuito y muy completo para transferencias

---

<sup>13</sup>Putty. <http://www.putty.org>

Protocolo	Encriptación	Autenticación	Puerto	Extensión
SCP / SFTP	Sí	Usuario/Contraseña	22	Archivos y carpetas del usuario
FTP	No	Usuario/Contraseña	21 ó 20	Carpeta FTP en el servidor

FTP en Windows y Linux es Filezilla<sup>14</sup>, que soporta todos estos protocolos. Para transferencias SCP, los clientes SSH explicados anteriormente: OpenSSH<sup>12</sup> y Putty<sup>13</sup> integran este protocolo. OpenSSH integra adicionalmente el protocolo SFTP.

#### 6.4.1. Protocolo SCP

Los comandos necesarios para *copiar archivos entre nuestro ordenador y el clúster* con el *protocolo SCP* son muy similares a los utilizados en la conexión segura mediante SSH (Subsección 6.3.1). Se deben especificar el usuario, la dirección IP o nombre de dominio del servidor SSH/SCP (el clúster) y el puerto de conexión si es diferente al 22 por defecto. El comando ‘scp’ utilizará como argumentos la ruta a los archivos origen y destino, en este orden:

```
$ scp -P 2222 clusteruser@cluster.domain.com:[ORIGEN REMOTO] [DESTINO LOCAL]
$ scp -P 2222 [ORIGEN LOCAL] clusteruser@cluster.domain.com:[DESTINO REMOTO]
```

Si queremos copiar un archivo desde nuestro directorio local a nuestro directorio ‘home’ del clúster:

```
$ scp -P 2222 archivo clusteruser@cluster.domain.com:/home/clusteruser/
```

o más fácilmente (porque el directorio ‘home’ es el directorio de destino por defecto):

```
$ scp -P 2222 archivo clusteruser@cluster.domain.com:
```

Si se desea copiar un archivo desde el clúster al directorio local actual (‘.’):

```
$ scp -P 2222 clusteruser@cluster.domain.com:/home/clusteruser/archivo .
```

Finalmente, existen tres opciones muy útiles: ‘-r’ que permite copiar archivos y carpetas completas recursivamente entre máquinas, ‘-p’ que permite preservar la fecha de los archivos y ‘-d’ que permite preservar los enlaces simbólicos:

```
$ scp -P 2222 -drp carpeta clusteruser@cluster.domain.com:/home/clusteruser/
```

Todos los archivos y directorios de la carpeta local serán copiados al directorio ‘home’ remoto conservando sus fechas de creación, modificación y acceso.

### 6.5. Sistema gestor de colas

La ejecución de procesos en un clúster requiere un mecanismo de control que optimice la asignación de los recursos disponibles en cada momento a los usuarios y que permita el control de la ejecución de los mismos. Dicho control se suele realizar a nivel de *software* con un *sistema gestor de colas* o de *procesado en batch*, del inglés *Distributed Resource Management System* (DRMS). Un DRMS permite ejecutar un número óptimo de procesos según los recursos del clúster y *poner en cola* el excedente de procesos a ejecutar, de forma parecida a la cola en un supermercado según el número de empleados en caja. Un DRMS debe ser tolerante a un posible fallo en los trabajos y matar procesos que excedan los límites de tiempo o de memoria asignados para no perjudicar al resto. Además el DRMS debe ofrecer

---

<sup>14</sup>Filezilla. <https://filezilla-project.org/>

una interfaz unificada a los usuarios para enviar sus trabajos al clúster. Un DRMS consiste básicamente en tres módulos [?]:

- *Gestor de colas*: recibe las peticiones de ejecución de trabajos de los usuarios y las pone en cola.
- *Gestor de recursos*: recibe la información sobre el estado de los nodos, su carga y los recursos de cálculo disponibles.
- *Planificador de procesos*: define el modo en que los procesos son ejecutados, es decir cuándo y dónde. Para ello recibe la información de los gestores de colas y de recursos, y de acuerdo a los recursos disponibles y los criterios definidos por el administrador del clúster manda procesos a los nodos para su ejecución.

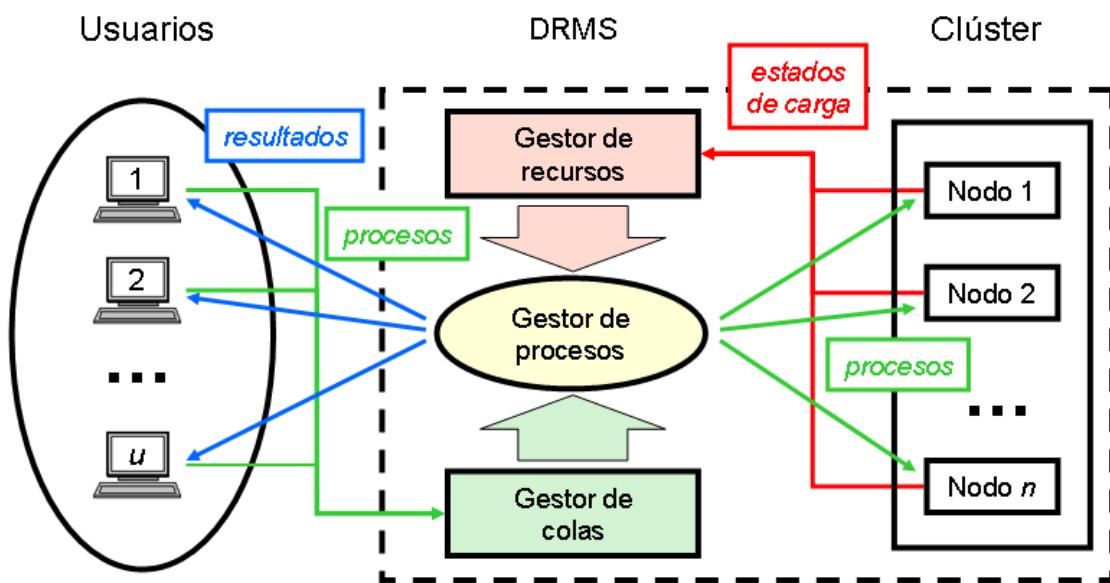


Figura 6.5: Esquema de funcionamiento de un DRMS.

Existen numerosos DRMS<sup>15</sup>, los más usados son *Open Grid Scheduler*<sup>16</sup> (Antiguo *Sun Grid Engine*<sup>17</sup>), *Condor*<sup>18</sup> y *TORQUE*<sup>19</sup> (Antiguo *Portable Batch System* (PBS)). El principal problema de los DRMSs de código libre es su falta de desarrollo con respecto a otras versiones comerciales mejoradas, aún así siguen siendo operativos para la mayoría de clústeres usados en laboratorios de bioinformática.

En la presente sección nos centraremos en explicar cómo mandar y gestionar procesos en un clúster usando Sun Grid Engine. Aunque dicho DRMS está obsoleto, sigue siendo uno de los más usados en los clústeres de laboratorios de investigación y sus comandos y modo de uso son muy similares, cuando no idénticos, a los usados por Open Grid Scheduler, PBS Y TORQUE.

<sup>15</sup>Listado de software para clústeres. [http://en.wikipedia.org/wiki/Comparison\\_of\\_cluster\\_software](http://en.wikipedia.org/wiki/Comparison_of_cluster_software)

<sup>16</sup>Open Grid Scheduler. <http://sourceforge.net/projects/gridscheduler>

<sup>17</sup>Sun Grid Engine. [http://es.wikipedia.org/wiki/Sun\\_Grid\\_Engine](http://es.wikipedia.org/wiki/Sun_Grid_Engine)

<sup>18</sup>Condor. <http://research.cs.wisc.edu/htcondor>

<sup>19</sup>TORQUE. <http://www.adaptivecomputing.com/products/open-source/torque>

## 6.6. Sun Grid Engine

*Sun Grid Engine* (SGE) es, como ya se ha explicado, un DRMS o sistema de colas que permite gestionar el envío de trabajos, orden de ejecución y reglas de uso en un conjunto de ordenadores o clústeres. Permite la ejecución remota de trabajos por lotes o interactivos y ver el estado actual de los trabajos enviados, así como modificarlos o borrarlos. Es un DRMS altamente configurable, sencillo de administrar y de utilizar por parte de los usuarios. Su escalabilidad lo ha llevado a ser el sistema de mayor éxito entre los mayores clúster de supercomputación. Al igual que Condor y PBS, SGE trabaja con *scripts* de consola que definen los requisitos de los trabajos de los usuarios, pero además puede tratar directamente con binarios e incluso ejecutar trabajos interactivos.

SGE fue inicialmente un producto de la histórica compañía Sun Microsystems, que en 2001 liberó el código y adoptó el modelo de desarrollo de código abierto. En 2010 Oracle compró Sun Microsystems y SGE pasó a llamarse Oracle Grid Engine y el nuevo *software* dejó de ser *open source*. Por ello se creó una variante libre de SGE denominada proyecto Open Grid Scheduler<sup>16</sup> y mantenido por la comunidad. Actualmente Open Grid Scheduler sigue siendo un proyecto de código abierto activo y Oracle Grid Engine ha pasado a denominarse Univa Grid Engine, de código propietario soportado comercialmente por la compañía Univa<sup>20</sup>.

SGE ofrece varios *comandos de consola* para el envío y administración de nuestros trabajos en el clúster. Normalmente sólo necesitaremos conocer el manejo de cuatro comandos para llevar a cabo todos nuestros trabajos en el clúster:

- qrsh: ejecuta un programa o abre una sesión interactiva en el clúster.
- qsub: envía una cola de trabajos al clúster.
- qstat: consulta el estado de nuestros trabajos en el clúster.
- qdel: borra un trabajo de la cola de ejecución del clúster.

### 6.6.1. Ejecución ordinaria de trabajos (qrsh)

Ya se ha remarcado anteriormente que no se deberían ejecutar cálculos en el nodo de acceso del clúster (Subsección 6.3.2). Todos los programas y trabajos interactivos deben ejecutarse en nodos secundarios. Para ello existe el comando ‘qrsh’ que abre una consola para *ejecutar nuestros programas* o directamente los ejecuta *en un nodo del clúster diferente al principal*. qrsh es un equivalente de ssh (Subsección 6.3.1), pero únicamente puede ser usado una vez hemos iniciado una sesión remota en el clúster y no necesita ningún parámetro. Su ejecución implica ‘saltar’ a la consola de un nodo diferente al de acceso o ejecutar comandos y programas sin afectar el rendimiento de dicho nodo. Por defecto qrsh usará un único *slot* o núcleo en uno de los nodos configurados para dicho uso.

La ejecución de qrsh es muy sencilla, simplemente teclear el comando en la consola del nodo de acceso, tras iniciar sesión en el clúster:

```
[clusteruser@headnode ~]$ qrsh
```

El símbolo del sistema (*command prompt*), cambiará a:

```
[clusteruser@n30 ~]$
```

Lo que significa que estamos usando un *slot* en el nodo 30. En esta nueva consola remota podemos ejecutar programas de la manera habitual y también ejecutar sesiones interactivas, por ejemplo de R:

---

<sup>20</sup>Univa. <http://www.univa.com>

```
[clusteruser@n30 ~]$ R
```

De esta simple manera podemos *ejecutar programas disponibles en el clúster*, incluso de forma interactiva y realizar trabajos de computación sin afectar al nodo de acceso (si docenas de usuarios ejecutaran programas a la vez en el nodo de acceso, éste colapsaría).

Deberemos *recordar cerrar la sesión* ('exit' o 'Control+D'). De otra forma, estaremos ocupando de forma innecesaria un *slot* que no estará disponible para otros trabajos en el clúster. Para salir y volver al nodo de acceso, teclear:

```
[clusteruser@n30 ~]$ exit
```

qrsh también puede *ejecutar un programa sin abrir una consola*:

```
[clusteruser@headnode ~]$ qrsh -cwd python script.py
```

La opción '*-cwd*' indica a SGE *ejecutar el script en el clúster desde el directorio de trabajo actual* (de otra forma será ejecutado desde el directorio '/home', que probablemente no es lo deseado).

Si deseamos *usar varios núcleos o requerimientos especiales de memoria* con 'qrsh', deberemos especificar parámetros adicionales al ejecutarlo, de la misma manera que se explicará para el comando 'qsub' (Subsección 6.6.6 y Subsección 6.6.7).

### 6.6.2. Procesamiento por lotes

El *procesamiento por lotes* (en inglés, *batch processing* o *job scheduling*) consiste en la ejecución de una serie de procesos o trabajos (*jobs*) en el clúster de forma controlada por un sistema gestor de colas (Sección 6.5), sin ser necesaria la supervisión por parte del usuario. Ello implica que todos los datos de entrada de los programas son procesados por medio de *scripts*, comandos o instrucciones del DRMS. Al contrario, los procesos interactivos requerían la entrada de información por parte del usuario. En el procesamiento por lotes, un programa toma una serie de ficheros de datos como entrada, procesa los datos y produce una serie de ficheros de resultados.

Los *beneficios* del procesamiento por lotes son evidentes:

- Se puede realizar el procesado de los datos cuando el clúster tiene disponibles los recursos suficientes.
- Evita los tiempos muertos que se producen cuando existe una supervisión y ejecución manual.
- Optimiza el uso global del clúster y con ello su amortización, especialmente importante si el clúster es muy caro.
- Permite al sistema usar diferentes prioridades para el procesado por lotes y los trabajos interactivos.

### 6.6.3. Ejecución de procesos por lotes (qsub)

'qsub' es el comando encargado de *ejecutar procesos por lotes*. Especificando la opción '*-b y*' qsub ejecuta programas de la misma forma que qrsh (Subsección 6.6.1) con la única condición de que el programa no puede ser interactivo:

```
$ qsub -cwd -b y python script.py
```

```
$ qsub -cwd -b y R CMD BATCH script.R
```

La opción ‘-cwd’ funciona de la misma manera que para qrsh e indica a SGE ejecutar el programa en el clúster desde el directorio de trabajo actual (de otro modo, éste sería ejecutado desde el directorio /home)

Por defecto qsub no permite ejecutar archivos binarios y requiere que cada trabajo enviado al clúster sea un *shell script*. Crear dicho *script* es muy sencillo y se explicará a continuación poniendo como ejemplo la ejecución de un proceso por lotes de R.

Existen dos pasos básicos para enviar el proceso por lotes de R al clúster:

- Primero, asumiendo que nuestro programa en R está guardado en ‘script.R’, deberemos crear un *shell script* que lo invoque y ejecute cuando se envíe al clúster. Para ello creamos un nuevo archivo en el mismo directorio llamado ‘batch.sh’ conteniendo la siguiente línea:

```
R CMD BATCH script.R
```

El archivo podrá contener líneas adicionales para especificar parámetros opcionales a SGE.

- Segundo, ‘batch.sh’ será usado como argumento de qsub para ejecutar ‘script.R’ en el clúster:

```
$ qsub -cwd batch.sh
```

Tras enviar el trabajo, podremos usar el comando qstat para comprobar el estado de nuestros procesos (Subsección 6.6.4).

El comando qsub puede ser usado con diferentes opciones, las más importantes son:

- help: Imprime un listado de todas las opciones.
- N name: Especifica un nombre para el trabajo.
- a date\_time: Define la hora y fecha para activar la ejecución del trabajo.
- cwd: Ejecuta el trabajo desde el directorio actual de trabajo.
- wd working\_dir: Ejecuta el trabajo desde el directorio especificado.
- p priority: Define la prioridad de ejecución en relación con otros trabajos. Es un número entre -1023 y 1024. La prioridad por defecto es 0.
- l resource=value, . . . : Especifica una lista de requerimientos de recursos, se explicará en Subsección 6.6.6.
- pe parallel\_environment: Será explicado en Subsección 6.6.7.

Por ejemplo, el siguiente comando enviará un trabajo denominado ‘jobname’ para ser ejecutado el 26 de abril a las 12:00 desde el directorio ‘/home/clusteruser/qsub’:

```
$ qsub -N jobname -a 04261200 -wd /home/clusteruser/qsub batch.sh
```

Las opciones de qsub también pueden ser añadidas al *shell script* (batch.sh). El Código 6.7 muestra como especificar diversos parámetros adicionales. Sólo se puede escribir una opción por línea y deben ir precedidas por ‘#’. Las líneas que comienzan por ‘#’ son comentarios.

Código 6.7: *Shell script* con parámetros adicionales de qsub.

```
1#!/bin/sh
2
3 # Cuenta de usuario para anotar el uso de cpu
4 #$ -A clusteruser
5
6 # fecha-hora para ejecutar el proceso, formato [[CC]yy]MMDDhhmm[.ss]
```

```

7 #$ -a 12241200
8
9 # Ejecución de 6 o más procesos paralelos que requieren 128MB de memoria
10 #$ -pe pvm 6 -l mem=128
11
12 # Enviar email a los siguientes usuarios
13 #$ -M clusteruser1@host,clusteruser1@host
14
15 # Enviar email al comienzo, final y/o suspensión del trabajo
16 #$ -m bes
17
18 # Exportar las siguientes variables de entorno
19 #$ -v PVM_ROOT,FOOBAR=BAR
20
21 # Ejecutar el trabajo desde el directorio de trabajo actual
22 #$ -cwd
23
24 python script_1.py
25 python script_2.py
26 ...
27 python script_n.py

```

En el manual de qsub (ejecutar ‘man qsub’ en la consola) se puede consultar más información sobre parámetros adicionales.

Finalmente, recordar de nuevo que el comando qsub no puede ser seguido directamente de un programa ejecutable, por ejemplo: ‘qsub python script.py’. El trabajo será asignado y aparecerá en cola si ejecutamos inmediatamente qstat, pero pasados unos segundos será cancelado y aparecerá un mensaje de error en un archivo. La forma correcta de ejecución sería: ‘qsub -b y python script.py’

#### 6.6.4. Control del estado de los trabajos (qstat)

‘qstat’ es el comando encargado de mostrarnos el *estado de los trabajos* enviados al clúster. Si hemos iniciado una sesión en un nodo del clúster con qrsh, podemos ver su estado ejecutando qstat desde una nueva sesión iniciada en el nodo de acceso:

```
$ qstat -u clusteruser
```

y obtendremos una respuesta similar a la siguiente:

job-ID	prior	name	user	state	submit/start at	queue	slots
141	0.55500	QRLOGIN	clusteruser	r	09/17/2013 14:12:16	all.q@n29.acid	1

Lo que significa que tenemos activa una sesión interactiva con qrsh en el nodo 29, ocupando 1 *slot* y con el nombre ‘QRLOGIN’.

Sin embargo la principal utilidad de qstat es comprobar el estado de los trabajos enviados mediante qsub. Por defecto, qstat sin ningún argumento nos mostrará el estado de nuestros procesos enviados al clúster:

```
$ qstat
```

Si no existen trabajos en cola, qstat no imprimirá nada, pero si existen, mostrará un listado de los mismos:

job-ID	prior	name	user	state	submit/start at	queue	slots
143	0.00000	jobname2	clusteruser	qw	09/17/2013 14:25:10		1
142	0.00000	jobname1	clusteruser	r	09/17/2013 14:23:24		1

```
141 0.55500 QRLOGIN clusteruser r 09/17/2013 14:12:16 all.q@n31.acid 1
```

En la columna ‘state’ se puede ver el estado de cada proceso según los siguientes códigos:

r: el trabajo se está ejecutando.

t: el trabajo se está transfiriendo a un nodo.

qw: el trabajo permanece en cola (sin ejecutar).

Eqw: ha habido un error al ejecutar el trabajo.

En el manual de qstat (ejecutar ‘man qstat’ en la consola) se puede encontrar más información sobre los códigos de estado.

Una información muy importante es el ‘job-ID’ de cada trabajo. Éste número será necesario si queremos hacer cambios en el trabajo, por ejemplo eliminarlo, como se explicará en la siguiente sección.

Ejecutando qstat con la opción ‘-j job-ID’ podemos obtener información adicional sobre uno de los trabajos (Código 6.8).

```
$ qstat -j 142
```

#### Código 6.8: Información completa de qstat sobre un trabajo.

```
1 =====
2 job_number: 142
3 exec_file: job_scripts/142
4 submission_time: Tue Sep 17 14:23:24 2013
5 owner: clusteruser
6 uid: 5032
7 group: Ldap
8 gid: 2000
9 sge_o_home: /home/clusteruser
10 sge_o_log_name: asebastian
11 sge_o_path: /usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin
12 sge_o_shell: /bin/bash
13 sge_o_workdir: /home/clusteruser/qsub
14 sge_o_host: headnode
15 account: sge
16 hard_resource_list: h_vmem=2G,mem_free=1536M
17 mail_list: clusteruser@host
18 notify: FALSE
19 job_name: jobname1
20 jobshare: 0
21 hard_queue_list: all.q
22 env_list:
23 script_file: batch.sh
```

Y ejecutando qstat con la opción ‘-f’ mostrará información detallada sobre cada nodo (núcleos totales y en uso, carga promedio, arquitectura...) y podremos ver a qué nodo han sido asignados nuestros trabajos (Código 6.9). The command ‘qstat -f’ will show a detailed information about all the cluster nodes (total and used cores per node, load average, architecture) and the assignations of our jobs.

#### Código 6.9: Información de qstat sobre los nodos del clúster.

queuename	qtype resv/used/tot.	load_avg	arch	states
all.q@n29.acid	BIP 0/0/64	0.00	linux-x64	

```

5 all.q@n30.acid           BIP    0/32/64      0.51      linux-x64
6 -----
7 all.q@n31.acid           BIP    0/1/64       0.01      linux-x64
8     142 0.55500 jobname1 clusteruser   r    09/17/2013 14:23:24      1
9 -----
10 all.q@n32.acid          BIP    0/0/64       0.02      linux-x64
11 -----

```

### 6.6.5. Eliminación de procesos (qdel), errores y salida estándar

`qdel` es el comando que permite *borrar un trabajo de la cola* del clúster. Para ello, primero hay que obtener el ‘job-ID’ ejecutando `qstat` e indicarlo como argumento de `qdel`:

```
$ qdel 142
```

Ahora dicho trabajo no debería aparecer en la lista de trabajos mostrada por `qstat`:

job-ID	prior	name	user	state	submit/start at	queue	slots
143	0.00000	jobname2	username	qw	09/17/2013 14:25:10		1
141	0.55500	QRLOGIN	username	r	09/17/2013 14:12:16	all.q@n29.acid	1

Por defecto la salida y errores de los trabajos (*standard output* y *standard error*) se guardan en dos archivos en el directorio `/home` o en la ubicación indicada mediante las opciones ‘`-cwd`’ y ‘`-wd`’ de `qsub`

[`jobname`].`o`[`job-ID`]: salida estándar (Ej. ‘`jobname.o142`’)

[`jobname`].`e`[`job-ID`]: error estándar (Ej. ‘`jobname.e142`’)

La ubicación de estos archivos puede ser indicada con las opciones ‘`-o`’ y ‘`-e`’ de `qsub`.

### 6.6.6. Requerimientos de memoria

Cuando se envían trabajos, si no se especifican ningunos requerimientos de memoria, SGE los ejecutará en el nodo disponible con la menor carga de trabajo, sin comprobar la disponibilidad de memoria en dicho nodo. Por ello, es muy importante especificar a `qsub` (o `qrsh`) cuánta memoria van a requerir nuestros trabajos aproximadamente.

A continuación se expondrá un ejemplo de porqué es importante *especificar la memoria necesaria para un trabajo*, incluso si no necesita más que unos pocos *gigabytes*. Imaginemos que un usuario ha solicitado 18 GB de memoria y esté usando todos ellos o incluso más en un nodo con 20 GB totales y 8 núcleos. Además el usuario podría estar usando un único núcleo del nodo, dejando 7 núcleos disponibles. Si nosotros enviamos al mismo clúster un trabajo sin requerimientos de memoria, dicho trabajo podría ser ejecutado en uno de los 7 núcleos del mencionado nodo. Aunque dicho nodo tiene múltiples núcleos disponibles, al ejecutar nuestro trabajo y saturar la memoria RAM del nodo, éste empezaría a usar memoria de intercambio (*swap*) que copia y lee datos en disco, con lo cual se ralentizaría considerablemente la ejecución de los dos trabajos en el nodo.

Para solucionar este problema, los administradores suelen definir unos requerimientos mínimos de memoria por defecto para cada trabajo enviado. En este caso, también sería ventajoso especificar los requerimientos de cada trabajo, porque si estos son menores que los establecidos por defecto, podremos ejecutar más trabajos en el mismo nodo al mismo tiempo.

Existen tres parámetros principales para definir las necesidades de memoria de un trabajo y se especifican en qsub con la opción ‘-l’ separados por comas:

mem\_free: es la memoria RAM mínima libre que debe estar disponible en el nodo para que ejecute nuestro trabajo.

h\_vmem: es la máxima cantidad de memoria que nuestro trabajo puede usar. Este parámetro es recomendado para parar trabajos que consumen excesiva memoria y pueden colapsar el nodo. Suele haber un límite establecido por defecto por el administrador.

h\_fsize: es el tamaño máximo permitido a los archivos generados por el trabajo. Es un parámetro igualmente útil, que evita que se colapsen los nodos por falta de espacio.

Los parámetros de memoria y espacio pueden especificarse en *megabytes* ‘M’ or *gigabytes* ‘G’.

En el siguiente ejemplo, se requieren 4 GB de memoria en un nodo para ejecutar un trabajo, el trabajo será eliminado si consume más de 6 GB de memoria o genera archivos mayores de 1 GB:

```
$ qsub -cwd -l mem_free=4G,h_vmem=6G,h_fsize=1G batch.sh
```

Un ejemplo similar para ejecutar una consola interactiva de R con qrsh en un nodo con 5 GB de memoria disponibles:

```
$ qrsh -l mf=4G,h_vmem=6G R
```

Para asignar los requerimientos de memoria, primero deberemos comprobar y testar cuánta memoria necesita nuestro proceso, a continuación se explicará como hacerlo.

## Conocer el uso de memoria de un proceso

Antes de comenzar las pruebas de uso de memoria de nuestros programas o *scripts*, deberemos preparar una versión rápida y manejable de los mismos, ya sea creando un pequeño archivo con datos de prueba o una versión rápida del programa para probarlo sin usar qsub.

La herramienta más simple para *medir el tiempo y recursos consumidos* por nuestro programa es el comando de Linux ‘time’. Con dicho comando no sólo se mide el tiempo de ejecución, sino también puede medir la *cantidad de memoria* empleada con la opción ‘-v’ (*verbose*):

```
$ /usr/bin/time -v batch.sh
```

La ruta completa ‘/usr/bin/time’ debe ser especificada para poder ejecutar el modo *verbose*, al terminar la ejecución se nos mostrará la memoria máxima requerida como ‘Maximum resident set size’, ver Código 6.10.

Código 6.10: Información mostrada por el comando ‘time’.

```
1 Command being timed: "batch.sh"
2 User time (seconds): 7.56
3 System time (seconds): 0.66
4 Percent of CPU this job got: 31 %
5 Elapsed (wall clock) time (h:mm:ss or m:ss): 0:26.45
6 Average shared text size (kbytes): 0
7 Average unshared data size (kbytes): 0
8 Average stack size (kbytes): 0
9 Average total size (kbytes): 0
10 Maximum resident set size (kbytes): 806880
11 Average resident set size (kbytes): 0
12 Major (requiring I/O) page faults: 26
```

```
13 Minor (reclaiming a frame) page faults: 51354
14 Voluntary context switches: 13921
15 Involuntary context switches: 1661
16 Swaps: 0
17 File system inputs: 2256
18 File system outputs: 40
19 Socket messages sent: 0
20 Socket messages received: 0
21 Signals delivered: 0
22 Page size (bytes): 4096
23 Exit status: 0
```

Otra opción es ejecutar nuestro programa con `qsub`, y una vez terminado, comprobar el uso de memoria con el comando ‘`gacct`’ especificando el número de proceso (ver cómo obtenerlo en la Subsección 6.6.4) y la opción ‘`-j`’:

```
$ qacct -j 142 | grep vmem
maxvmem      768.039M
```

Mientras se está ejecutando un trabajo con `qsub` también podemos ver el uso de memoria mediante `qstat` (Subsección 6.6.4):

```
$ qstat -j 142 | grep vmem
```

### 6.6.7. Uso de múltiples *slots*

Cuando el programa o trabajo a ejecutar en el clúster está preparado para ejecutar el código de forma paralela, permitiendo el uso de múltiples núcleos, deberemos requerir al DRMS un *entorno de ejecución con múltiples slots* (un *slot* se corresponderá con un núcleo).

La opción ‘`-pe local N`’ de `qsub` permite requerir  $N$  *slots* en un nodo del clúster para ejecutar trabajos de forma paralela. A su vez es muy recomendado continuar usando ‘`mem_free`’ para seleccionar un nodo que tenga suficiente memoria disponible para nuestro trabajo (Subsección 6.6.6). El valor de ‘`mem_free`’ debería referirse a la memoria total que esperamos que use nuestro trabajo. Sin embargo si especificamos ‘`h_vmem`’ deberemos dividir el valor de memoria entre el número de *slots* ( $memoria/N$ ) puesto que SGE lo multiplicará por  $N$  cuando requiramos múltiples *slots*. En el siguiente ejemplo SGE asignará 40 *slots* a nuestro trabajo en un nodo con 40 GB de memoria disponibles y fijará el límite de memoria a usar en 80 GB ( $40 \times 2GB$ ):

```
$ qsub -cwd -pe make 40 -l mem_free=40G,h_vmem=2G batch.sh
```

## Parte II

# Macromoléculas biológicas, alineamiento de secuencias y filogenia



## Capítulo 7

# Macromoléculas biológicas: proteínas, DNA y RNA

*Inmaculada Yruela y Alvaro Sebastián*

### 7.1. Introducción

El descubrimiento de las estructuras químicas del DNA y las proteínas, y de la relación entre ambas no fue tarea fácil en la historia de la biología. Desde que Mendel realizara sus experimentos a mitad del siglo XIX y Miescher aislara el DNA del esperma de salmón en 1869, tuvieron que pasar más de 70 años para descubrir la relación de los genes con el metabolismo. En 1941 Beadle y Tatum desarrollaron la hipótesis “*un gen, una enzima*” con sus experimentos sobre el metabolismo del moho *Neurospora* [1]. En 1944 Avery confirmó unos experimentos realizados por Griffith años antes que demostraban que el DNA era el material genético, hasta entonces se creía que las proteínas cargaban la información hereditaria. Unos años después Sanger [9] consiguió secuenciar parte de la secuencia de la insulina y Watson y Crick resolvieron el enigma de la estructura del DNA a partir de las imágenes de difracción de rayos X de Franklin [10]. Sin embargo se seguía sin conocer cuál era el código que permitía traducir DNA a proteínas. Hasta que casi 10 años más tarde, en 1961, Crick y Brenner demostraron que un codón consistía en 3 pares de bases de DNA [3], a la vez que Matthei y Nirenberg consiguieron dar con la clave del código genético [7] con la ayuda de la polinucleótido fosforilasa descubierta anteriormente por Severo Ochoa [4]. El código fue completado poco después por Nirenberg, Leder [6] y Khorana [5]. El descubrimiento de que 3 nucleótidos de DNA codificaban para la síntesis de 1 aminoácido fue un descubrimiento que revolucionó la ciencia. Actualmente comparando una secuencia de RNA y la secuencia de la proteína codificada sería muy sencillo descifrar el código genético y ver que 3 nucleótidos codifican 1 aminoácido, pero hace 50 años no existían dichas secuencias y llegaba a ser una tarea muy compleja, incluso filosófica. Finalmente, el método clásico de secuenciación de DNA que ha permitido conocer la secuencia de miles de genes y proteínas fue desarrollado por Sanger en 1975 [8] hasta la llegada hace pocos años de las nuevas tecnologías de secuenciación que permiten secuenciar genomas eucariotas completos en unas horas.

## 7.2. Genes y proteínas

Un *gen* es una secuencia de nucleótidos en la molécula de DNA (o RNA, en el caso de algunos virus) que contiene la información necesaria para que a través de una cascada de procesos biológicos se culmine la síntesis de una macromolécula con función celular específica, habitualmente proteínas pero también mRNA, rRNA y tRNA<sup>1</sup>. En el genoma humano se estiman pocos más de 20000 genes, siendo su secuencia codificante de proteínas sólo un 1.5% de la longitud total del genoma [2]. Si pensamos que hasta hace pocos años todos los estudios se centraban en genes y proteínas, podemos concluir que conocemos muy poco acerca de nuestro genoma.

Volviendo al dogma fundamental de la biología “*un gen, una proteína*”, podemos describir los 3 procesos biológicos que permiten la síntesis de una proteína a partir de la información contenida en un gen en organismos eucariotas:

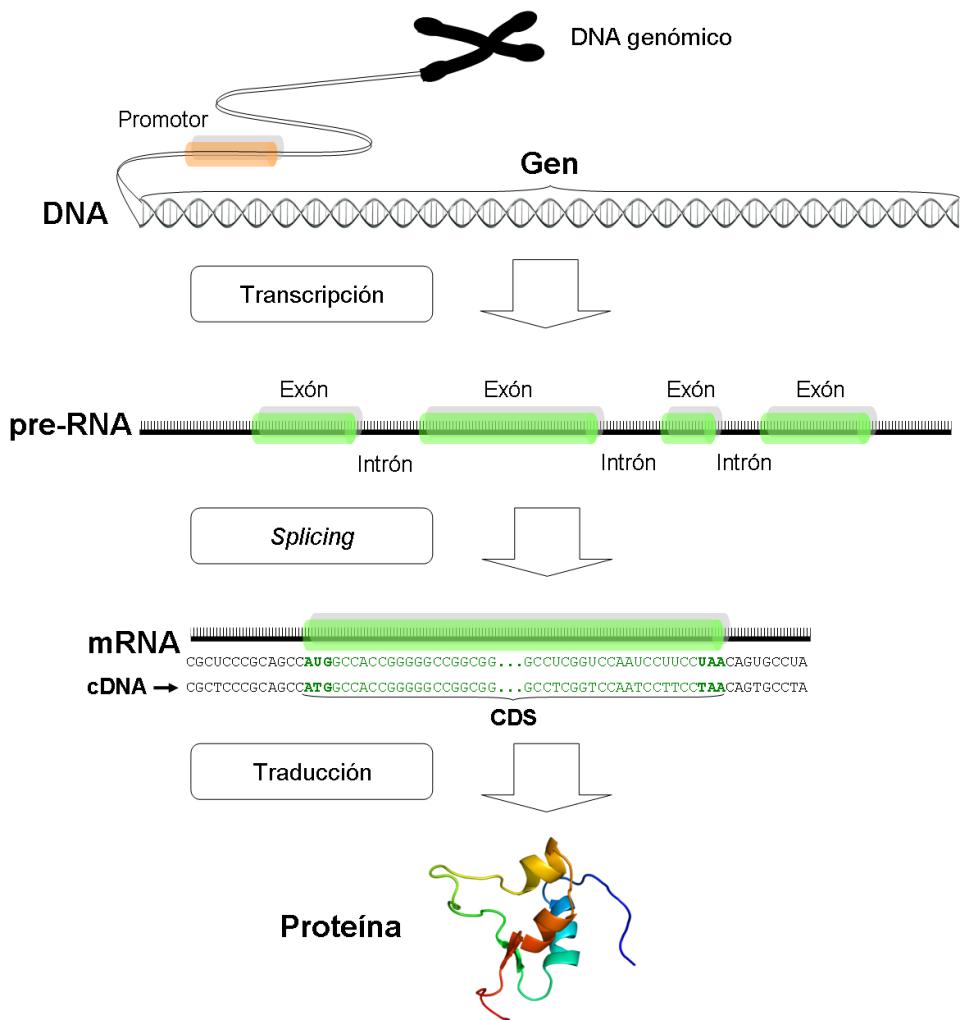
- *Transcripción*: Consiste en la síntesis de RNA a partir de una cadena molde de DNA genómico mediante la enzima RNA polimerasa, el RNA sintetizado se denomina mRNA o RNA mensajero.
- *Splicing* (o ajuste): El mRNA tras la transcripción es cortado y empalmado de forma que se eliminan secuencias llamadas intrones y se conservan secuencias llamadas exones (el *splicing* no sucede en procariotas).
- *Traducción*: Es el proceso durante el cual un mRNA maduro se usa de molde para sintetizar proteínas. Se lleva a cabo en los ribosomas, donde por cada 3 nucleótidos en el RNA (codón) se añade un aminoácido a la proteína que se sintetiza.

Los tres procesos están esquematizados en la Figura 7.1. Como se puede observar, en el proceso de *splicing* el pre-RNA pierde parte de la secuencia original del DNA (los *intrones*), conservando los *exones*, y en la traducción parte de los extremos del mRNA no se traducen a proteína (UTR o región no traducida). La complejidad de estos procesos, así como la variedad de combinaciones posibles nos hace distinguir en las bases de datos diferentes tipos de secuencias:

- *DNA genómico*: DNA tal y como se encuentra en el cromosoma o material genético objeto de estudio y que se replica de una generación a la siguiente.
- *Gen*: Consiste habitualmente, como ya se ha explicado, en una pequeña parte del DNA genómico que tiene la particularidad de poder ser transcrita a mRNA para sintetizar posteriormente una proteína.
- *Promotor*: Secuencia de DNA que no se transcribe y que precede a un gen facilitando el acoplamiento de factores de transcripción y otras proteínas fundamentales para el inicio de la transcripción.
- *cDNA* (o *DNA complementario*): Es la secuencia de mRNA maduro (sin intrones) en forma de DNA (en la secuencia sólo cambia la letra U por la T). Estas cadenas de DNA se sintetizan a partir del RNA mediante las enzimas transcriptasa reversa y DNA polimerasa. Se suele utilizar para la secuenciación y clonación de genes.
- *EST* (o *marcador de secuencia expresada*): Es una sub-secuencia de un cDNA. Se pueden usar para identificar genes que se transcriben y en secuenciación y descubrimiento de genes.
- *CDS* (*secuencia codificante*): Consiste en la parte de un gen que codifica una proteína. Es la secuencia de cDNA formada por exones y comprendida entre las UTRs o regiones no traducidas.

---

<sup>1</sup>Definición modificada de Wikipedia. <http://es.wikipedia.org/wiki/Gen>

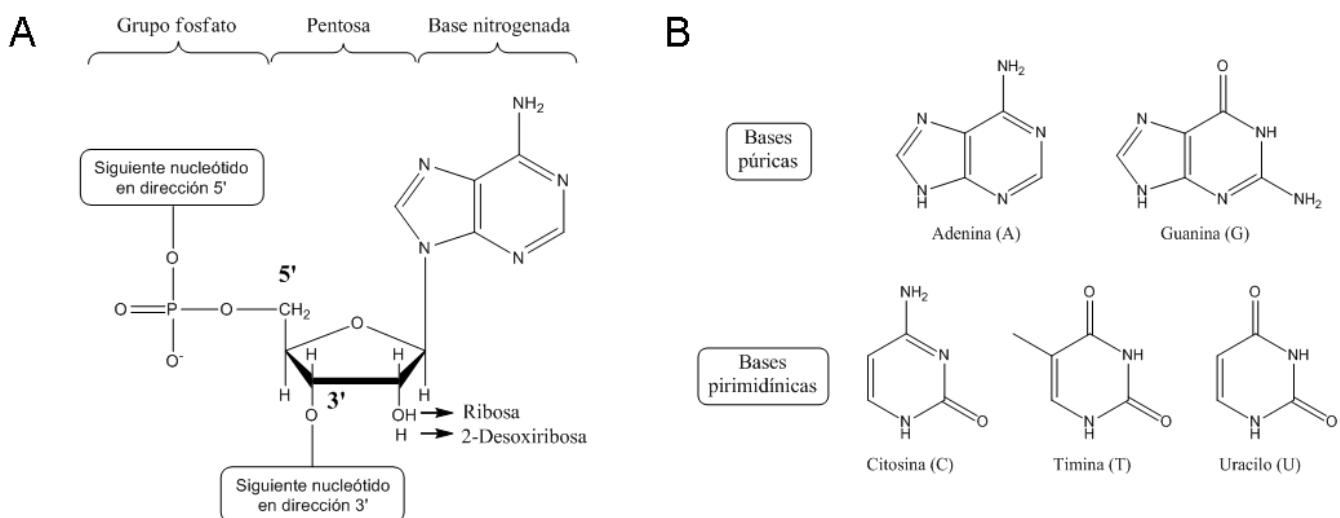


**Figura 7.1:** Esquema de los procesos de transcripción, *splicing* y traducción que conducen a la síntesis de una proteína a partir de una secuencia génica de DNA.

- *ORF* (o *marco abierto de lectura*): A veces no se conoce dónde comienzan y terminan las UTRs y no es posible asignar una CDS a un gen. En estos casos se predicen una o varias secuencias ORFs entre los supuestos inicios y finales de la traducción. Frecuentemente se confunde ORF con CDS, sin embargo la CDS real formará parte de una de las ORFs predichas.
- *Proteína*: Una proteína puede estar formada por varias cadenas peptídicas, pero vulgarmente se define como proteína a la secuencia de aminoácidos de un péptido fruto de la traducción del mRNA maduro.

### 7.3. Estructura primaria de DNA y RNA

El *DNA o ácido desoxirribonucleico* y el *RNA o ácido ribonucleico* son polímeros compuestos por unidades repetidas llamadas nucleótidos. Cada *nucleótido* está compuesto de una base nitrogenada, un azúcar de 5 carbonos que puede ser ribosa o 2-desoxiribosa y un grupo fosfato (Figura 7.2A). En los extremos del nucleótido quedan un grupo fosfato unido a la posición 5' del azúcar y un grupo OH en posición 3' del azúcar (Figura 7.2A), dichos extremos determinan la dirección de la secuencia, generalmente en biología se trabaja con las secuencias en dirección 5' → 3'.



**Figura 7.2:** A: Estructura de un nucleótido. B: Letra, nombre y fórmula de las bases nitrogenadas.

Las *bases nitrogenadas* que conforman los nucleótidos pueden ser de dos tipos: púricas y pirimidínicas según sea el anillo de purina o el de pirimidina el núcleo de su estructura. Tanto en DNA como en RNA están presentes las bases púricas llamadas *adenina* (A) y *guanina* (G) y la base pirimidínica *citosina* (C) (Figura 7.2B). Existe otra base pirimidínica denominada *timina* (T) que sólo conforma el DNA y que su equivalente en el RNA es el *uracilo* (Figura 7.2B). La particularidad de estas bases es que pueden formar enlaces de hidrógeno intermoleculares entre adenina y timina y entre guanina y citosina. Estas interacciones intermoleculares permiten al DNA formar una estructura de doble hélice formada por dos cadenas sencillas enroscadas en direcciones opuestas (antiparalelas, una en dirección 5' → 3' y otra 3' → 5') donde las bases nitrogenadas se encuentran en el interior formando enlaces de hidrógeno entre pares A-T y G-C. Debido a la estructura de doble cadena, cuando hacemos referencia al tamaño de una doble hélice de DNA solemos hablar de *pares de bases*, así el genoma humano tiene un tamaño aproximado de 3200 millones de pares de bases. El RNA también puede formar apareamientos

de bases pero tiene generalmente estructura monocatenaria. Cada nucleótido según la base nitrogenada que lo compone se designa con la letra asignada a dicha base (Figura 7.2B) y la estructura polimérica se representa con una sucesión de letras en sentido 5' → 3' denominada *estructura primaria*, ver ejemplo en la Figura 7.3. También existen letras para designar combinaciones de varias bases en una determinada posición de la secuencia<sup>2</sup>, a utilizar por ejemplo cuando existe un 50 % de probabilidad de una u otra base en dicha posición.

```
5' -ATGCCCTGTGGATGCCCTCCTGCCCTGCTGGCGCTGCTGCCCTCTGGGACCTGACCCAG  
CCGCAGCCTTGTGAACCAACACCTGTGGCTCACACCTGGTGGAAAGCTCTACCTAGTGTGCGG  
GGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGACCTGCAGGTGGGCAGGTG  
GAGCTGGCGGGGCCCTGGTGCAGGCAGCCTGCAGCCCTGGCCCTGGAGGGTCCCTGCAGAACG  
GTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCTTACCAAGCTGGAGAACTACTGCAAC-3'
```

**Figura 7.3:** Ejemplo de secuencia de la ORF (cDNA) del gen precursor de la insulina humana (INS).

## 7.4. El código genético

El código genético es el conjunto de reglas que permite a las células traducir la información del material genético (DNA o mRNA) en aminoácidos constituyentes de las proteínas. Como ya se ha mencionado, los genes se dividen en secuencias de 3 nucleótidos llamadas codones y cada codón codifica para un aminoácido en la traducción a proteína del mRNA. El código genético representado en la Tabla 7.1 es universal, aunque tiene excepciones en diferentes organismos y orgánulos celulares. Tres características fundamentales del código genético son:

- *Es un código degenerado*: varios codones pueden codificar el mismo aminoácido, por ejemplo el aminoácido serina es codificado por 6 codones, glicina por 4, sin embargo metionina sólo es codificado por 1 codón. La degeneración del código permite que determinadas mutaciones en el DNA no causen modificaciones en las proteínas codificadas.
- *Codones de inicio y parada*: según el organismo determinados codones indican el comienzo y final de la traducción. El codón de inicio más habitual es AUG que codifica metionina, mientras que los que la finalizan son: UAA, UAG, UGA.
- *Marco de lectura*: una secuencia de DNA o mRNA debe ser leída de 3 en 3 nucleótidos en el orden correcto, dicho orden se denomina marco de lectura. Una doble hélice de DNA contiene 6 posibles marcos de lectura (3 para cada cadena) por ello es muy importante conocer cuál es el que dará como resultado la secuencia proteica adecuada.

En el ejemplo de la Figura 7.4 se pueden observar los codones, el marco de lectura y los aminoácidos que codifican en el gen de la insulina humana. La conversión de secuencias de DNA a proteína en los tres marcos de lectura y de proteína a DNA se puede realizar con numerosas herramientas disponibles online<sup>3</sup>.

<sup>2</sup>Códigos IUPAC para nucléotidos y aminoácidos. <http://www.bioinformatics.org/sms/iupac.html>

<sup>3</sup>Herramientas de traducción de secuencias en la página web del EMBL-EBI. <http://www.ebi.ac.uk/Tools/st/>

		SEGUNDA BASE					
		U	C	A	G		
PRIMERA BASE	U	UUU Phe UUC UUA UUG	UCU Ser UCC UCA UGG	UAU Tyr UAC UAA FIN UAG	UGU Cys UGC UGA FIN UGG Trp	U C A G	
	C	CUU CUC Leu CUA CUG	CCU CCC CCA CGG	CAU His CAC Pro CAA Gln CAG	CGU CGC CGA CGG	U C A G	Arg
	A	AUU AUC Ile AUU AUG Met	ACU ACC ACA AGG	AAU Asn AAC Thr AAA Lys AAG	AGU Ser AGC AGA AGG Arg	U C A G	
	G	GUU GUC Val GUA GUG	GCU GCC GCA GGG	GAU Asp GAC Ala GAA Glu GAG	GGU GGC GGA GGG Gly	U C A G	

**Tabla 7.1:** Código genético representado por tripletes de bases y los aminoácidos que codifican.

atggccctgtggatgcgcctcctgcccctgctggcgctgctggccctctggggacacctgaccagcc  
M A L W M R L L P L L A L L A L W G P D P A  
gcagccttgtgaaccaacacacctgtgcggctcacacacctggtaagctctcacctagtgtgcggg  
A A F V N Q H L C G S H L V E A L Y L V C G  
gaacgaggcttcttctacacacccaagacccgccggaggcagaggacactgcaggtagggcaggtag  
E R G F F Y T P K T R R E A E D L Q V G Q V  
gagctggccggggccctggtgaggcagcctgcagccctggccctggagggtccctgcagaag  
E L G G G P G A G S L Q P L A L E G S L Q K  
cgtggcattgtggaaacaatgcttaccagcatctgctcccttaccagctggagaactactgcaac  
R G I V E Q C C T S I C S L Y Q L E N Y C N

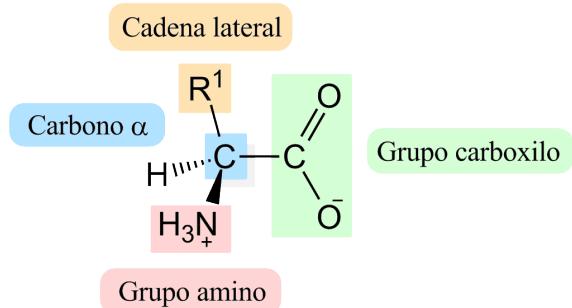
**Figura 7.4:** Traducción del ORF de la insulina humana mostrada en la Figura 7.3 a su secuencia proteica de la Figura 7.11 mostrando el aminoácido codificado por cada codón.

## 7.5. Aminoácidos y enlace peptídico

### 7.5.1. Aminoácidos

Las *proteínas* están compuestas por uno o más polipéptidos. Cada *polipéptido* es un polímero formado por aminoácidos unidos entre sí mediante enlaces peptídicos. Un *aminoácido* es un compuesto químico que cuenta con un *grupo amino* ( $-NH_2$ ) de carácter básico y un *grupo carboxilo* ( $-COOH$ ) de carácter ácido unidos a un mismo carbono llamado *carbono alfa* ( $\alpha$ ) que además une un hidrógeno y una *cadena lateral* variable ( $-R$ ) que distingue cada uno de los 20 aminoácidos presentes en los seres vivos (Figura 7.5). La glicina es el aminoácido más simple y pequeño donde el grupo R es un hidrógeno (H).

Los *aminoácidos* tienen un papel central como elementos constitutivos de las proteínas pero también



**Figura 7.5:** Estructura general de los  $\alpha$ -aminoácidos que forman las proteínas. El carbono  $\alpha$  ( $C\alpha$ ) tiene cuatro sustituyentes: un átomo de hidrógeno (H), un grupo carboxilo ( $-COOH$ ), un grupo amino ( $-NH_2$ ), y un grupo R que es diferente en cada uno de los aminoácidos. Los grupos amino y carboxilo se representan en su forma ionizada predominante a pH neutro.

juegan un importante papel como intermediarios en el metabolismo celular. *Los humanos producimos 10 de los 20 aminoácidos que forman las proteínas*, los otros 10 deben ser adquiridos a través de la alimentación. Esto es debido a que no tenemos los enzimas requeridos para la biosíntesis de todos los aminoácidos. Los 10 aminoácidos que podemos producir son alanina, asparagina, ácido aspártico (aspartato), cisteina, ácido glutámico (glutamato), glutamina, glicina, prolina, serina y tirosina. La tirosina se produce a partir de la fenilalanina. Si la dieta es deficiente en fenilalanina debe ser adquirida también. Las plantas son capaces de sintetizar todos los aminoácidos.

Los aminoácidos se suelen clasificar en grupos atendiendo a las *propiedades fisicoquímicas de las cadenas laterales*. Es importante señalar, que cualquier clasificación ordena y organiza los elementos de un conjunto según unas reglas, de forma que facilite y simplifique el estudio. Por tanto, podemos encontrarnos clasificaciones que difieren ligeramente en el criterio de clasificación y agrupan los aminoácidos de distintas maneras (ej. polaridad, pH, tipo de grupo en la cadena lateral, etc.). Lo importante, por tanto, es conocer la estructura y propiedades de cada uno de ellos.

Los 20 aminoácidos mayoritarios se pueden clasificar de acuerdo a la estructura y propiedades físicas-químicas de su cadena lateral en 4 grupos (Figura 7.6):

- **Neutros apolares:** cadenas con residuos poco polares que repelen a las moléculas de agua (hidrofóbicas), son: glicina, alanina, valina, leucina, isoleucina, metionina, prolina, fenilalanina y triptófano.
- **Neutros polares:** cadenas con residuos polares e hidrófilos que pueden formar interacciones débiles con moléculas de agua, son: serina, treonina, cisteína, tirosina, asparagina y glutamina.
- **Ácidos:** cadenas con residuos ácidos cargados negativamente a pH fisiológico, capaces de protonar moléculas de agua y disminuir el pH, son: ácido aspártico y ácido glutámico.
- **Básicos:** cadenas con residuos básicos cargados positivamente a pH fisiológico, capaces de ‘robar’ protones a las moléculas de agua y aumentar el pH, son: lisina, arginina e histidina.

También se puede distinguir otro grupo, los *aromáticos*, que son aminoácidos ya clasificados en las categorías anteriores que contienen en su cadena lateral un anillo aromático: fenilalanina, tirosina y triptófano.

Las propiedades físico-químicas de los aminoácidos no sólo determinan su actividad biológica sino también el *plegamiento de la proteína* en una estructura tridimensional y la *estabilidad de la estructura*.

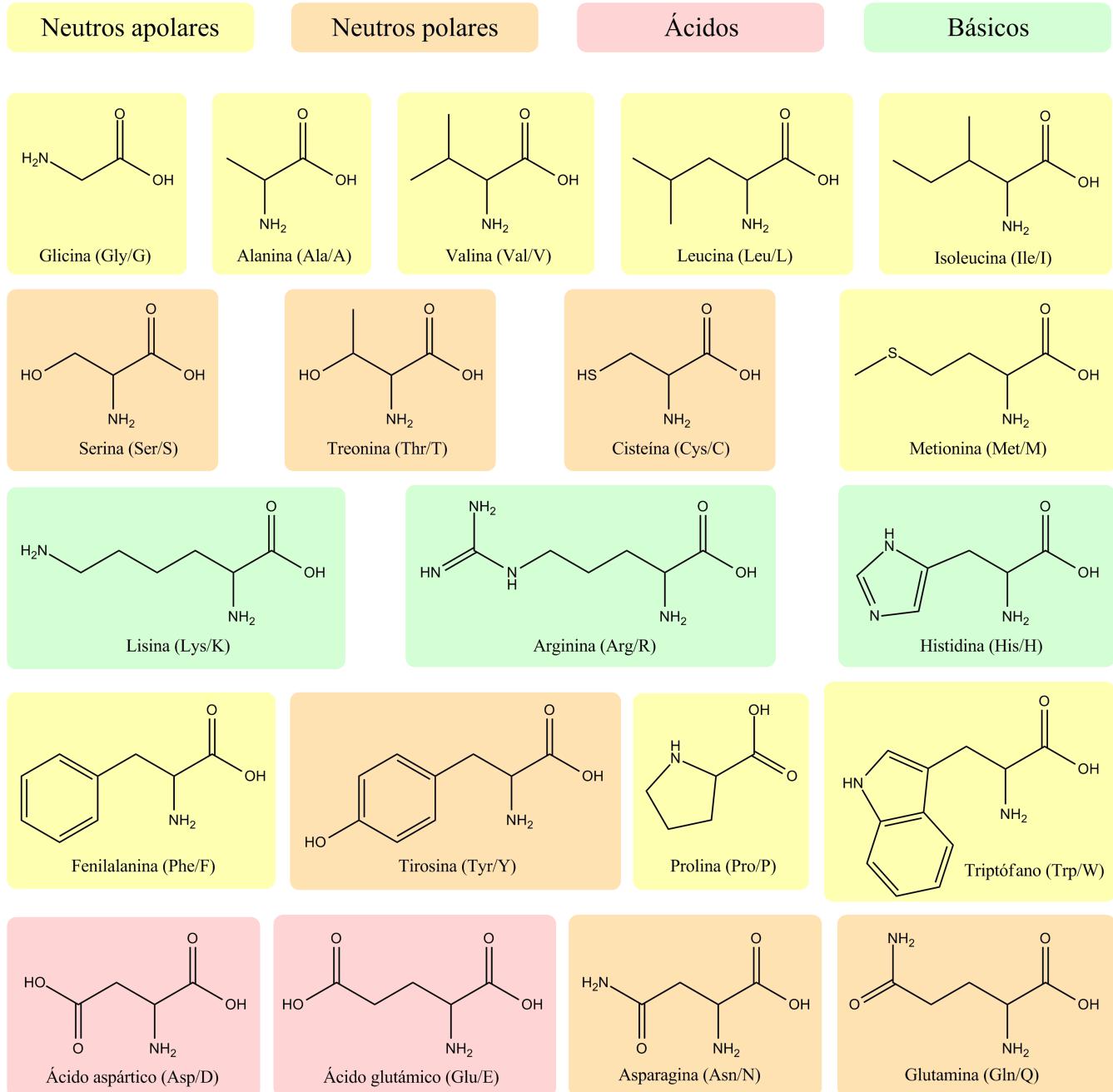
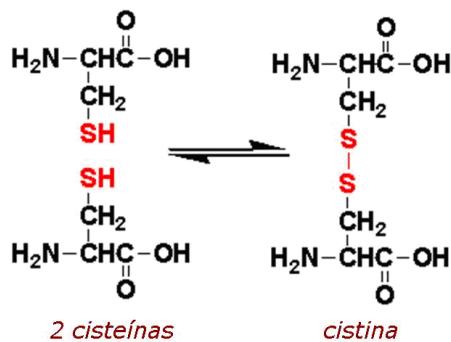


Figura 7.6: Fórmulas y nomenclatura de los 20 aminoácidos.

resultante. Algunas propiedades también ayudan a la *caracterización bioquímica* de las proteínas. Por ejemplo, los aminoácidos aromáticos son los responsables de la *absorbancia a 280 nm*, típica de las proteínas. Con la medida de la absorbancia a esta longitud de onda obtenida en un espectrofotómetro se puede determinar, de forma no destructiva, la concentración de una disolución proteíca. A continuación haremos un repaso de las propiedades particulares de cada uno de los 20 aminoácidos esenciales ordenados alfabéticamente.

- *Alanina* (Ala, A): es una molécula hidrofóbica ambivalente, es decir, puede encontrarse tanto en el interior como en el exterior de una proteína. El C $\alpha$  de la alanina es ópticamente activo, pero en proteínas sólo se encuentra el isómero L. Tiene como cadena lateral un grupo metilo (-CH<sub>3</sub>) y se utiliza como referencia para determinar las propiedades de los demás aminoácidos, que pueden considerarse derivados de él. Al ser apolar no participa en ningún mecanismo catalítico por lo que su función es meramente estructural. La alanina es un  $\alpha$ -aminoácido análogo al  $\alpha$ -cetopiruvato, un intermediario en el metabolismo de los azúcares, por lo que alanina y piruvato son intercambiables por una reacción de transaminación.
- *Arginina* (Arg, R): tiene un grupo guanidinio que capta protones y está cargado positivamente a pH neutro. Puede unir anión fosfato y a menudo se encuentra en los centros activos de las proteínas que unen sustratos fosforilados. Como catión, la arginina, así como la lisina, tiene un papel importante en el mantenimiento del balance de cargas de una proteína. La arginina tiene además un papel importante en el metabolismo del nitrógeno. En el código genético hay 6 codones para arginina. Sin embargo, aunque este elevado número de codones generalmente se asocia con una alta frecuencia del aminoácido en la secuencia de proteínas, la arginina es uno de los aminoácidos menos frecuentes. Esta discrepancia entre su frecuencia y el número de codones es mayor para arginina que para otros aminoácidos.
- *Asparagina* (Arg, N): es la amida del ácido aspártico. El grupo amida no soporta una carga formal bajo ninguna condición de pH biológicamente relevante, por lo que se mantiene neutro a pHs fisiológicos. El grupo amida se hidroliza fácilmente convirtiéndose la asparagina en ácido aspártico. Esta reacción se considera que está relacionada con las bases del envejecimiento. La asparagina suele formar puentes de hidrógeno dado que el grupo amida puede aceptar y donar dos hidrógenos. Se suele encontrar tanto en la superficie como en el interior de las proteínas. La asparagina es un sitio frecuente de unión de carbohidratos en glicoproteínas.
- *Ácido aspártico* (o aspartato, Asp, D): es uno de los dos ácidos, junto al ácido glutámico (o glutamato), que tienen una función importante en los centros activos de los enzimas, además de mantener la solubilidad y el carácter iónico de las proteínas. El pK<sub>a</sub> del grupo  $\beta$ -carboxílico del ácido aspártico es  $\approx 4,0$ . Participa en mecanismos de catálisis ácido/base. El ácido aspártico es homólogo al oxaloacetato, de igual modo que el piruvato lo es de la alanina. Así, el ácido aspártico y el oxaloacetato son interconvertibles por una simple reacción de transaminación.
- *Cisteína* (Cys, C): es uno de los dos aminoácidos que contienen átomos de azufre; el otro es la metionina (Met, M). La cisteína difiere de la serina en un átomo de azufre (S) en el grupo tiol que sustituye al oxígeno (O) del alcohol. Si no está ionizada no es tan polar como la serina o la treonina. Es de resaltar que la cisteína y la serina son mucho más diferentes en sus propiedades fisico-químicas que lo que sugiere su similitud molecular. Por ejemplo, el protón del grupo tiol (-SH) de la cisteína es mucho más ácido que el protón del grupo hidroxilo (-OH) de la serina, lo que hace que el primero sea más reactivo que el segundo. La cisteína se puede oxidar y condensarse con otra cisteína formando un *punte disulfuro* (-S-S-) (Figura 7.7) que tiene carácter covalente y sirve para entrecruzar dos regiones de una proteína o dos subunidades de un complejo proteíco y mantener una conformación estable. El ambiente dentro de la célula es demasiado reductor para

formar puentes disulfuros pero en el espacio extracelular los puentes disulfuro se pueden formar y estabilizar muchas proteínas.



**Figura 7.7:** Puente disulfuro entre dos residuos de cisteína. La oxidación de dos cisteínas próximas y con orientación adecuada ocurre de forma espontánea dando lugar a un enlace covalente entre dos átomos de azufre (S). En ocasiones dos cisteínas unidas por un puente disulfuro se denomina cistina.

- *Ácido glutámico* (o glutamato, Glu, E): tiene un grupo metileno adicional en su cadena lateral comparado con el ácido aspártico, y se denomina grupo carboxílico  $\gamma$ . Su  $pK_a$  es 4,3, significativamente más alto que el del ácido aspártico. Esto es debido al efecto inductivo del grupo metileno adicional. El ácido glutámico y el  $\alpha$ -cetoglutarato, un intermediario en el *ciclo de Krebs*, son interconvertibles por transaminación. Por lo tanto, el ácido glutámico puede entrar en el ciclo de Krebs y ser convertido por la enzima glutamina sintetasa en glutamina, que juega un papel importante en el metabolismo del nitrógeno. También es de resaltar que el ácido glutámico puede fácilmente convertirse en prolina. En un primer paso, el grupo carboxílico  $\gamma$  se reduce a aldehído, produciendo glutamato semialdehído que posteriormente reacciona con el grupo  $\alpha$ -amino, eliminando agua y formando una *base de Schiff*. En un segundo paso de reducción, la base de Schiff se reduce dando lugar a la prolina.
- *Glutamina* (Gln, Q): es la amida del ácido glutámico y se encuentra no cargada en todas las condiciones biológicas. El grupo metileno adicional en su cadena lateral, comparado con la asparagina le permite en su forma libre o como N-terminal de la cadena proteíca espontáneamente ciclar y formar una estructura de anillo de pirrolidona que se encuentra en el N-terminal de muchas inmunoglobulinas. Este detalle dificulta la determinación de la secuencia de aminoácidos en experimentos de secuenciación.
- *Glicina* (Gly, G): es el más pequeño de los aminoácidos. Su cadena lateral R es un átomo de hidrógeno, por lo que dos de los cuatro sustituyentes del  $C\alpha$  son iguales y no presenta isomería óptica. Es el aminoácido que más flexibilidad proporciona a las proteínas, pues su pequeño tamaño no obstaculiza el movimiento de los aminoácidos que lo flanquean. Es ambivalente, es decir, se puede encontrar tanto en la superficie como en el interior de las proteínas.
- *Histidina* (His, H): tiene un grupo imidazol en su cadena lateral R que es capaz de captar protones y tener carga positiva a pH neutro. Participa generalmente en reacciones catalizadas por enzimas. La forma desprotonada del grupo imidazol tiene un carácter nucleofílico y actúa como base, mientras que la forma protonada actúa como ácido. La histidina tiene un papel en la estabilidad del plegamiento de las proteínas.
- *Isoleucina* (Ile, I): como su nombre indica es un isómero de la leucina, y junto a ésta y la valina es uno de los tres aminoácidos que tienen cadenas laterales R ramificadas. Generalmente se

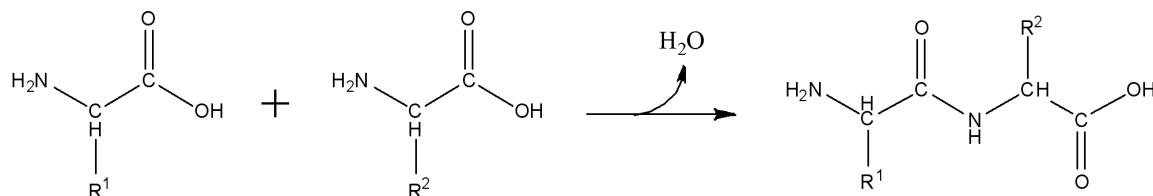
intercambia con leucina y ocasionalmente con la valina. Las cadenas laterales de estos aminoácidos no son reactivas y por lo tanto no participan en uniones covalentes y en la actividad catalítica de los centros activos de los enzimas. Sin embargo, estos aminoácidos confieren cierta rigidez a las proteínas, debido a la presencia de un segundo carbono asimétrico en la cadena lateral, que tiene siempre configuración S. Estos aminoácidos también tienen relevancia en la unión de ligandos en las proteínas y juegan un papel importante en la estabilidad de las mismas. El C $\beta$  de la isoleucina, al igual que el de la treonina, es ópticamente activo y por tanto ambos son centros quirales.

- *Leucina* (Leu, L): al igual que la isoleucina y la valina, tiene una cadena lateral ramificada y un grupo metileno (-CH<sub>2</sub>) adicional comparado con la valina. Tiene un carácter hidrofóbico y generalmente se encuentra en el interior de las proteínas.
- *Lisina* (Lys, K): tiene un grupo  $\epsilon$ -amino positivamente cargado. Básicamente la lisina es una alanina con un sustituyente propilamino en el carbono- $\beta$ . El grupo  $\epsilon$ -amino tiene un pK<sub>a</sub> significativamente más alto (ca. 10,5) que aquellos en grupos  $\alpha$ -amino. El grupo  $\epsilon$ -amino de la lisina es muy reactivo y a menudo participa en las reacciones catalíticas de los centros activos de los enzimas. La lisina a menudo se encuentra en el interior de las proteínas con sólo el grupo  $\epsilon$ -amino expuesto al solvente.
- *Metionina* (Met, M): es un tioéter que incluye un átomo de azufre (S) en la cadena lateral R. La cadena lateral es muy hidrofóbica y generalmente se encuentra en el interior de las proteínas. Sus propiedades fisico-químicas están ligadas al átomo de azufre; no tiene un carácter muy nucleófilo aunque a veces reacciona con centros electrofílicos. De esta manera, aunque la metionina se considera un aminoácido apolar el átomo de azufre (S) aparece ocasionalmente implicado en reacciones bioquímicas en algunas enzimas. El átomo de azufre en la metionina, al igual que en la cisteína tiende a oxidarse. El primer paso da lugar a metionina sulfóxido y en un segundo paso irreversible se produce metionina sulfato. Esta reacción se piensa que es la causante del enfisema pulmonar en fumadores.
- *Fenilalanina* (Phe, F): es un derivado de la alanina con un sustituyente fenilo en el C $\beta$ . Es muy hidrofóbico, incluso la fenilalanina libre no es soluble en agua. Como dato curioso es de mencionar que Marshall Nirenberg y Phil Leder en 1964, cuando realizaban sus experimentos para determinar el triplete del código genético que estaba asociado a cada aminoácido, encontraron que el producto del mensajero sintético polyU, era insoluble. En aquellos momentos no sabían que 'UUU' codifica a fenilalanina, pero poco después de estos experimentos precipitaron el producto formado en su mezcla; ya estaban en el camino de desentrañar el código genético y las bases de la síntesis de proteínas, y conseguir el Premio Nobel de Fisiología y Medicina en 1968.
- *Prolina* (Pro, P): comparte muchas propiedades con los aminoácidos alifáticos. En sentido estricto no es un aminoácido sino un iminoácido, pues la cadena lateral termina uniéndose por su extremo con el grupo  $\alpha$ -amino. Por ello la prolina es particularmente rígida y, además, su grupo amino, cuando forma parte de un enlace peptídico no puede actuar como donador en puentes de hidrógeno para estabilizar hélices- $\alpha$  o láminas- $\beta$ . A menudo se dice que la prolina no puede encontrarse en una hélice- $\alpha$ . Cuando ello ocurre la hélice tendrá una leve curva debido a la falta del enlace del hidrógeno.
- *Serina* (Ser, S): difiere de la alanina en que uno de los hidrógenos del grupo metilo (-CH<sub>3</sub>) es sustituido por un grupo hidroxilo (-OH). Es una molécula hidrofílica y suele encontrarse en el centro catalítico de muchos enzimas por la capacidad del grupo -OH de actuar como nucleófilo y atacar a grupos deficientes en electrones de otras moléculas, promoviendo cambios químicos como la hidrólisis.

- *Treonina* (Thr, T): es una molécula hidrofílica y al igual que la serina contiene un grupo hidroxilo (-OH), pero difiere de ella en que tiene un sustituyente metilo (-CH<sub>3</sub>) en vez de un hidrógeno en el C<sub>β</sub>. Tanto el C<sub>α</sub> como el C<sub>β</sub> de la treonina son ópticamente activos.
- *Triptófano* (Trp, W): es el aminoácido de mayor tamaño. Es un derivado de la alanina, pues tiene un sustituyente indol en el C<sub>β</sub>. El grupo indol absorbe en el ultravioleta (UV) lejano. La nube  $\pi$  del anillo aromático puede actuar como aceptor de puentes de hidrógeno y también puede formar interacciones con grupos cargados positivamente (ej. cadenas laterales de aminoácidos básicos o ligandos catiónicos).
- *Tirosina* (Tyr, Y): es también un aminoácido aromático derivado de la fenilalanina por una hidroxilación en la posición *para*. Aunque la tirosina tiene carácter hidrofóbico debido a su anillo aromático, es significativamente más soluble que la fenilalanina y se considera polar y protonable. El hidroxilo del grupo fenólico es más ácido que el hidroxilo alifático de la serina o la treonina, teniendo un pK<sub>a</sub> de ca. 9,8. Al igual que todos los grupos ionizables el pK<sub>a</sub> preciso depende del ambiente donde se encuentre la proteína. Las tirosinas que se encuentran en la superficie de las proteínas generalmente tienen un pK<sub>a</sub> más bajo que las que se encuentran en el interior. La ionización de la tirosina da lugar al anión fenolato que puede ser inestable en el interior hidrofóbico de una proteína. La tirosina absorbe la radiación UV y contribuye a la absorbancia espectral de las proteínas al igual que lo hacen los demás aminoácidos aromáticos (triptófano y fenilalanina). Sin embargo, su contribución es menor, el coeficiente de extinción de la tirosina es ca. 1/5 que el del triptófano a 280 nm. El triptófano es el que más contribuye a la absorbancia en el UV en las proteínas (dependiendo del número de residuos que haya en la secuencia).
- *Valina* (Val, V): es una molécula hidrofóbica y se encuentra en el interior de las proteínas. Al igual que isoleucina confiere cierta rigidez a las proteínas al presentar una ramificación de la cadena lateral del C<sub>β</sub>. Los electrones  $\pi$  del anillo de fenilo favorecen el apilamiento con otras moléculas aromáticas y a menudo lo hacen en regiones enterradas de las proteínas, añadiendo estabilidad a la estructura. Es de señalar que la valina y la treonina tienen una estructura y volumen similar por lo que es difícil, incluso a alta resolución distinguirlas.

### 7.5.2. Enlace peptídico

El *enlace peptídico* une dos aminoácidos mediante la condensación del grupo amino y el grupo carboxilo para formar una amida ( $R_A-NH-CO-R_B$ ) (Figura 7.8). De esta forma, en los extremos de los péptidos quedan un residuo amino y otro carboxilo que serán los únicos que no formen enlaces peptídicos. Por convenio los péptidos se escriben y representan con el extremo N-terminal a la izquierda y el extremo C-terminal a la derecha.



**Figura 7.8:** Formación de un enlace peptídico entre dos aminoácidos.

Todos los aminoácidos tienen un grupo ácido y un grupo amino ionizables. Sin embargo, para comprender el comportamiento en solución de las proteínas hay que considerar que ambos grupos desparecen

cuando se forma el enlace peptídico entre el grupo carboxílico (-COOH) de un aminoácido y el grupo amino (-NH<sub>2</sub>) del aminoácido adyacente, y se polimerizan para formar las proteínas en el ribosoma. Sólo los grupos carboxílico y amino que ocupan los extremos de la cadena proteica quedan libres. Pero a pesar de ello, *las proteínas presentan numerosos grupos cargados en las cadenas laterales R que son ionizables*. Estos aminoácidos hacen que sea posible la *solubilidad en agua* de muchas proteínas. Las curvas de titulación de los aminoácidos ayudan a comprender cómo varía su carga en función del pH. Para los aminoácidos ionizables, el *punto isoeléctrico* es la media aritmética, aproximadamente de los pK<sub>a</sub> que delimitan el intervalo de pH en que predomina la forma del aminoácido sin carga neta.

El *enlace peptídico* es un *enlace covalente de tipo amida* resultado de una reacción de condensación con la pérdida de una molécula de H<sub>2</sub>O. La condensación no es espontánea, al tener lugar en un disolvente acuoso, por lo que requiere aporte de energía. El enlace peptídico puede romperse por *hidrólisis* (adición de una molécula de agua) liberando 8–16 kJ/mol (2–4 kcal/mol) de energía libre. Este proceso es muy lento y es facilitado por los enzimas. La longitud de onda de absorbancia del enlace peptídico es 190–230 nm lo que hace que sea particularmente sensible a la radiación UV. El enlace peptídico presenta cierto carácter de enlace doble que determina gran parte de las propiedades conformacionales de las proteínas. La mayoría de los enlaces peptídicos de las proteínas son de *isomería trans*. La forma *trans* es la de menor energía, sitúa a los sustituyentes voluminosos (C<sub>α</sub> con las cadenas laterales) más alejados y por tanto con menor conflicto estérico.

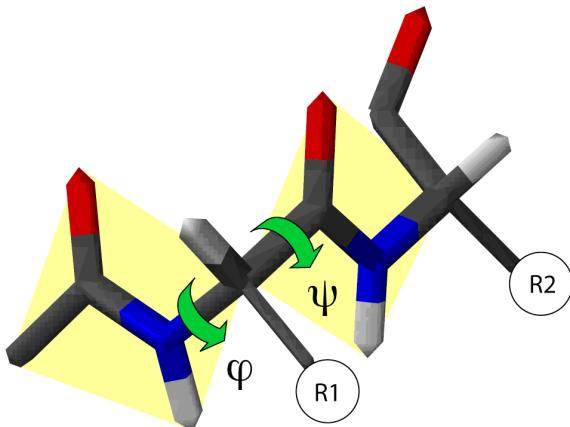
### 7.5.3. Ángulos de torsión y diagrama de Ramachandran

El enlace peptídico introduce restricciones que limitan las conformaciones que puede adoptar un polipéptido o proteína. La flexibilidad de estos depende del número de sus enlaces que permiten rotaciones de una parte de la molécula respecto al resto. Debido a su carácter de doble enlace la rotación completa no está permitida por lo que la rotación en torno al enlace peptídico ( $\omega$ ) sólo puede tener los valores 0°(conformación *cis*) y 180°(conformación *trans*). Cualquier otro valor sería desestabilizante. Los *ángulos de torsión*, también conocidos como *ángulos de Ramachandran*, describen las rotaciones del esqueleto de la cadena polipeptídica alrededor de los enlaces entre N y C<sub>α</sub> (ver Figura 7.9). El *ángulo Phi* ( $\varphi$ ) está definido por los átomos C-N-C<sub>α</sub>-C (C es el carbono carbonilo) y C<sub>α</sub>-C, y el *ángulo Psi* ( $\psi$ ) está definido por los átomos N-C<sub>α</sub>-C-N. Los ángulos  $\varphi$  y  $\psi$  de un aminoácido están restringidos por repulsiones estéricas.

El *diagrama de Ramachandran* es una sencilla representación de la distribución de los ángulos de torsión en la estructura de una proteína. Existen mayoritariamente dos zonas de ángulos  $\varphi$  y  $\psi$  permitidos, ambas con valores negativos del ángulo  $\varphi$ , y que se diferencian por el valor positivo o negativo del ángulo  $\psi$ . Cada punto ( $\varphi$ ,  $\psi$ ) en el diagrama representa a un aminoácido de la cadena polipeptídica (ver ejemplo en Figura 7.10).

El diagrama de Ramachandran permite tener una visión de las regiones de torsión que están permitidas y no permitidas, y por tanto de la flexibilidad de una región para adoptar un determinado plegamiento. No todas las parejas de ángulos son posibles dentro de las estructuras de los péptidos y proteínas, debido a los *efectos estéricos* entre los residuos (cadenas laterales) de los aminoácidos. Por tanto, el conocer los valores de los ángulos de torsión nos sirve como *indicador de la calidad de la estructura 3D de una proteína*. Los ángulos de torsión son uno de los parámetros estructurales más importantes que controlan

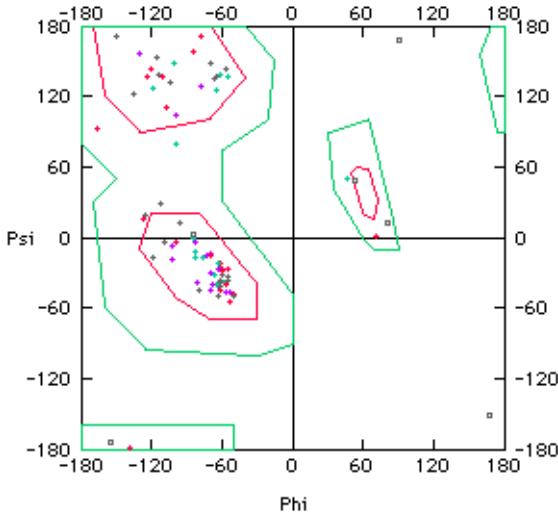
<sup>4</sup>Gráfico de Ramachandran en Wikipedia. [http://es.wikipedia.org/wiki/Gr\penalty\@M\hskip\z@\skip\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\accent19a\egroup\spacefactor\accent@spacefactor\penalty\@M\hskip\z@skip\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\spacefactor\accent@spacefactorfico\\_de\\_Ramachandran](http://es.wikipedia.org/wiki/Gr\penalty\@M\hskip\z@\skip\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\accent19a\egroup\spacefactor\accent@spacefactor\penalty\@M\hskip\z@skip\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\spacefactor\accent@spacefactorfico_de_Ramachandran)



**Figura 7.9:** Ángulos de torsión en una cadena polipeptídica. Los ángulos de torsión son ángulos dihedros definidos por cuatro puntos en el espacio. Los ángulos Phi ( $\varphi$ ) y Psi ( $\psi$ ) describen la rotación de la cadena alrededor de los enlaces a ambos lados del átomo  $C\alpha$ <sup>4</sup>.

el plegamiento de una proteína. Si somos capaces de predecir los ángulos de Ramachandran de una proteína seríamos capaces de predecir su plegamiento y su estructura secundaria. Esto es así porque el tercer posible ángulo de torsión dentro de una proteína ( $\omega$ ) está prácticamente fijo a  $180^\circ$ , debido al carácter parcial de doble enlace del enlace peptídico, que restringe la rotación alrededor del enlace C–N y coloca dos  $C\alpha$  contiguos y C, O, N e H entre ellos en un plano. Cuando la estructura de rayos X de una proteína no está adecuadamente refinada podemos encontrar ángulos de torsión en regiones no permitidas del diagrama de Ramachandran. En estos casos este tipo de desviaciones nos indica problemas con la estructura resultante de nuestro modelo o nuestra determinación.

El diagrama de Ramachandran también nos puede dar información sobre los diferentes elementos de *estructura secundaria* que componen la estructura 3D. En una estructura proteica se pueden distinguir regiones en las que los  $C\alpha$  de varios residuos consecutivos adoptan ángulos  $\varphi$  y  $\psi$  similares, hecho que hace que en estas regiones se encuentren disposiciones periódicas de sus unidades peptídicas. Estos elementos se pueden distinguir por el rango de valores de los ángulos  $\varphi$  and  $\psi$ . Las dos disposiciones periódicas tienen forma de hélice (hélice  $\alpha$ ) o de cadena extendida (lámina  $\beta$ ). Las hélices  $\alpha$  presentan valores promedio en el rango de  $\varphi -57^\circ$  y  $\psi -47^\circ$ , y las láminas  $\beta$  en el rango de  $\varphi -130^\circ$  y  $\psi +140^\circ$ . Cuando los ángulos de varios aminoácidos consecutivos se reparten entre las dos regiones no se forma una estructura secundaria. En estos casos se forma un bucle el que el polipéptido puede cambiar de dirección. Para adquirir estabilidad y generar complejas superficies de interacción las proteínas tienen regiones que carecen de la repetición de valores  $\varphi$  y  $\psi$ , característicos de las hélices  $\alpha$  y láminas  $\beta$ , estas regiones se denominan *bucles*. Los bucles más cortos se denominan *giros reversos* y están formados por sólo cuatro aminoácidos. En los giros abundan las glicinas.



**Figura 7.10:** Diagrama de Ramachandran del citocromo b5 (PDB 3b5c). En esta estructura refinada sólo los residuos de glicina se encuentran fuera de los regiones permitidas (polígonos con línea verde).

## 7.6. Niveles estructurales en proteínas

Para describir, analizar y comprender la estructura de las proteínas se han definido varios niveles de complejidad ordenados jerárquicamente. De esta manera las proteínas tienen cuatro niveles estructurales: i) *estructura primaria*, que corresponde a la secuencia de aminoácidos; ii) *estructura secundaria*, que se refiere a las disposiciones particulares estables de los aminoácidos, que dan lugar a patrones estructurales repetidos; iii) *estructura terciaria*, que describe la disposición tridimensional global de todos los átomos de una proteína; iv) *estructura cuaternaria*, que describe la disposición espacial de las subunidades polipeptídicas que componen una proteína.

### 7.6.1. Estructura primaria

Se define como *estructura primaria* de una proteína a las secuencias ordenadas de aminoácidos de los péptidos que la constituyen. A los aminoácidos dentro de un péptido o proteína también se les denomina *residuos*. Al igual que en el caso del DNA y el RNA, la secuencia proteica se representa con letras, en este caso con 20, una para cada aminoácido (Figura 7.6). Para facilitar la lectura también es muy común representar los residuos con secuencias de tres letras. El primer aminoácido por la izquierda representará el residuo N-terminal y el último por la derecha el C-terminal. El péptido representado se corresponderá con el péptido codificado por una secuencia de mRNA o cDNA en sentido  $5' \rightarrow 3'$  (provenientes de una hebra molde de DNA genómico transcrita en sentido  $3' \rightarrow 5'$ ), ver ejemplo en la Figura 7.11.

Además de la estructura primaria, se definen otros tres niveles estructurales en las proteínas que serán explicados más adelante en el capítulo: estructura secundaria, terciaria y cuaternaria. Los péptidos constituyentes de las proteínas se pliegan formando estructuras tridimensionales y se asocian entre sí para formar proteínas totalmente funcionales. Una proteína no es activa en su forma de cadena peptídica lineal, ésta requiere un determinado plegamiento tridimensional inducido por las propiedades físico-químicas de sus residuos y/o ayudado por proteínas y factores externos para poder ejercer su función celular.

*NH<sub>2</sub>-MALWMRLPLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRRE  
EAEDLQVGQVELGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN-COOH*

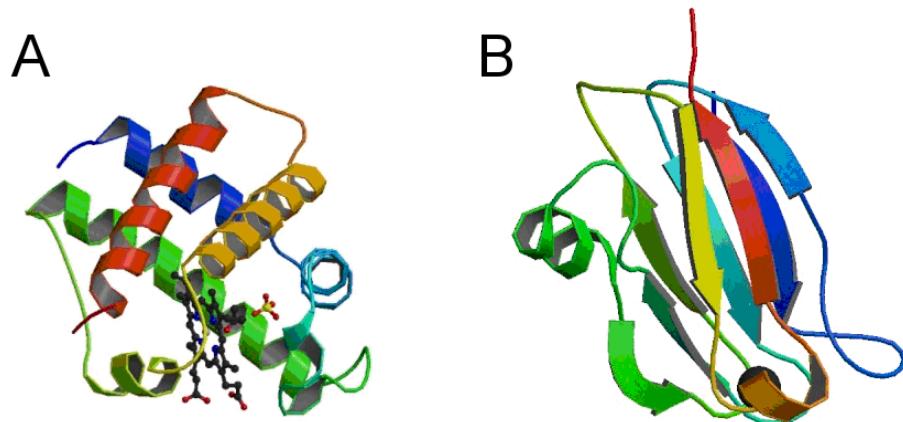
**Figura 7.11:** Ejemplo de secuencia de la pre-insulina humana (INS) tal y como se traduce del cDNA mostrado en la Figura 7.3, antes de sufrir modificaciones post-traduccionales para formar la insulina activa.

### 7.6.2. Estructura secundaria

Las proteínas naturales en disolución no son estables en conformaciones extendidas. En un solvente acuoso un polipéptido tiene muchos grupos capaces de formar puentes de hidrógeno y por tanto de manera espontánea se forman elementos de estructura secundaria que se conectan entre sí por lazos (*loops*) o por regiones intrínsecamente desordenadas o dúctiles. De esta manera la *estructura secundaria* se refiere a la conformación local de ciertas regiones de un polipéptido, es decir a los *patrones de plegamiento regulares que adopta la cadena polipeptídica*. Estos patrones son debidos a los *puentes de hidrógeno intramoleculares e intermoleculares* que establecen los grupos amida. La geometría que adopta la proteína está directamente relacionada con la geometría molecular del grupo amida. Sólo unas estructuras secundarias son estables y están ampliamente distribuidas en las proteínas: hélices  $\alpha$  y láminas  $\beta$ . La estructura de hélice es la disposición más sencilla que puede adoptar una cadena polipeptídica si tenemos en cuenta la rigidez de sus enlaces peptídicos y la libertad de rotación de los demás enlaces. Pauling y Corey (1951) predijeron esta estructura en base a imágenes de rayos X llamándola *hélice  $\alpha$* . En esta estructura el esqueleto polipeptídico se encuentra compactamente enrollado a lo largo del eje imaginario longitudinal de la molécula y de los grupos R de los aminoácidos que sobresalen del esqueleto helicoidal. La hélice está estabilizada por puentes de hidrógeno entre el grupo -NH de un aminoácido y el grupo -C=O del cuarto aminoácido consecutivo. La unidad repetida es el giro de la hélice. En cada giro se disponen 3,6 aminoácidos, esto significa que hay un aminoácido cada 100 grados de rotación ( $360^\circ/3,6$ ) y que cada residuo se traslada 1,5 Å a lo largo del eje longitudinal de la hélice, dando lugar a una distancia vertical de 5.4 Å entre átomos estructuralmente equivalentes en un giro. En general, en las proteínas globulares se observa que una cuarta parte de los aminoácidos se encuentran formando hélices  $\alpha$  aunque la proporción exacta varía en cada proteína. Ver ejemplo de hélice  $\alpha$  en la Figura 7.12A

Sin embargo, no todos los polipéptidos pueden formar una hélice  $\alpha$  estable. Las interacciones entre las cadenas laterales de los aminoácidos pueden estabilizar o desestabilizar la estructura (ej. una cadena con muchos Glu no podrá formar una hélice  $\alpha$  a pH 7,0 porque los grupos -COOH cargados negativamente de los Glu adyacentes se repelen con mayor intensidad que la atracción estabilizadora de los puentes de hidrógeno). La presencia de Pro o Gly también restringe la formación de hélices  $\alpha$ . Existen cinco tipos de *restricciones que afectan a la estabilidad de una hélice  $\alpha$* : i) la repulsión o atracción electrostática entre aminoácidos consecutivos con grupos R cargados; ii) el volumen de los grupos R adyacentes; iii) las interacciones de las cadenas laterales de aminoácidos separadas 3 o 4 aminoácidos; iv) la presencia de Pro y Gly; v) la interacción entre aminoácidos en los extremos de la hélice y el dipolo eléctrico de la estructura.

La conformación de *lámina  $\beta$*  en la cadena polipeptídica se encuentra extendida en zigzag en lugar de plegarse como una hélice. Los segmentos individuales se mantienen unidos por puentes de hidrógeno intermoleculares entre los grupos amida de dos segmentos separados. En este caso, los puentes de hidrógeno no se forman entre aminoácidos adyacentes como en las hélices  $\alpha$  sino entre aminoácidos de segmentos cercanos en la cadena polipeptídica. En algunos casos también pueden estar alejados. Los segmentos adyacentes de una lámina  $\beta$  pueden tener una orientación paralela o antiparalela. En



**Figura 7.12:** A. Estructura 3D de la leghemoglobina de *Glycine max* (soja, pdb 1BIN) a 2.2 Å de resolución. Esta hemo-proteína globular está formada mayoritariamente por hélices  $\alpha$ . En la estructura se observa la presencia del grupo hemo como cofactor. B. Estructura 3D de la plastocianina de *Anabaena variabilis* (pdb 2CJ3) a 1.7 Å de resolución. Esta proteína está formada mayoritariamente por láminas  $\beta$ . En la estructura se observa la presencia de un átomo de Cu como cofactor.

la orientación paralela, la cadena polipeptídica interacciona con otra en la misma disposición y en las que la dirección N-C es la misma. En este caso la interacción de varias cadenas da lugar a lo que se llama *hoja plegada paralela*. En la orientación antiparalela, las cadenas interaccionan de manera que su polaridad es opuesta. En este caso la estructura que resulta es la *hoja plegada antiparalela*. En el caso de la orientación paralela el periodo de repetición es más corto (6,5 Å), comparado con la antiparalela (7 Å), y los patrones de formación de puentes de hidrógeno son diferentes. En algunas estructuras densamente empaquetadas la presencia de ciertos aminoácidos está limitada. Por ejemplo, los grupos R de los aminoácidos de las superficies de contacto deben ser relativamente pequeños. Ver ejemplo de lámina  $\beta$  en la Figura 7.12B.

Otro elemento de estructura secundaria frecuente son los *giros  $\beta$* . Estos son elementos de conexión que unen segmentos sucesivos de hélices  $\alpha$  o láminas  $\beta$ . A menudo en los giros  $\beta$  son frecuentes los aminoácidos Gly y Pro. Esto es debido a que la Gly es pequeña y flexible, y la Pro adopta una configuración *cis* poco frecuente. Existen dos tipos de giros  $\beta$ , tipo I y tipo II.

### Motivos estructurales

Los *motivos estructurales* o también llamados *estructuras suprasecundarias* o *plegamientos* constituyen un nivel superior a la estructura secundaria y se definen como la disposición estable y la conectividad entre distintos elementos de estructura secundaria. En una proteína las hélices  $\alpha$  y láminas  $\beta$  pueden estar conectadas entre sí y combinadas de diferentes maneras, aunque en la naturaleza estas posibilidades son limitadas. Los motivos conocidos van de simple a complejo y aparecen a menudo como elementos repetidos o combinaciones. En general podemos distinguir cuatro tipos de estructuras suprasecundarias: i) todo hélices  $\alpha$  (ej. cuatro hélices  $\alpha$  empaquetadas); ii) hélice  $\alpha$ /lámina  $\beta$  donde los segmentos están alternados o entremezclados (ej. barril  $\alpha/\beta$ ); iii) hélice  $\alpha$  + lámina  $\beta$  donde los segmentos están algo segregados; iv) todo láminas  $\beta$  (ej. barril  $\beta$ ).

El análisis del plegamiento de una proteína nos puede revelar relaciones evolutivas que son difíciles de detectar al nivel de la secuencia polipeptídica (Capítulo 12). Esto puede ayudar a entender mejor la función, la actividad biológica o el papel en un determinado organismo. A la unidad funcional y evolutiva se le denomina *dominio*, y en general se puede decir que a cada dominio le corresponde

una función molecular. Existen varias bases de datos de referencia para definir dominios de proteínas. Por ejemplo, la base de datos Structural Classification Of Proteins<sup>5</sup> (SCOP) organiza las proteínas con estructura conocida y depositadas en la base de datos Protein Data Bank (ver Subsección 7.8.1) en base a criterios estructurales y evolutivos. Otra base de datos es Pfam<sup>6</sup>, que define familias de secuencias o dominios en base a alineamientos múltiples para facilitar su localización en otras proteínas. La base de datos SUPERFAMILY<sup>7</sup> contiene la anotación estructural y funcional de todas las proteínas y genomas. SUPERFAMILY asigna dominios en base a las definiciones de SCOP. Para más detalle consultar la Subsección 12.3.1.

### 7.6.3. Estructura terciaria

La *estructura terciaria* de una proteína es la disposición espacial de todos sus átomos y está determinada por la secuencia de aminoácidos que la compone. Las interacciones de las cadenas laterales de los residuos de la proteína guían al polipéptido para formar una estructura compacta. Existen cuatro tipos de *interacciones que cooperan para la estabilización de la estructura terciaria de las proteínas*: i) *puentes disulfuro*; ii) *interacciones hidrofóbicas* (*fuerzas de van der Waals*); iii) *puentes de hidrógeno*; iv) *interacciones iónicas*.

Los aminoácidos con cadenas laterales no polares tienden a localizarse en el interior de la proteína, en donde se asocian con otros aminoácidos con cadenas laterales no polares para alcanzar la máxima estabilidad posible. En general los aminoácidos polares tienden a encontrarse en la superficie de las proteínas. Esta organización en general está invertida en algunas proteínas de membrana que forman poros o canales, en donde los aminoácidos con cadenas laterales no polares están en contacto con los lípidos componentes de la bicapa lipídica de la membrana y los aminoácidos polares están en el centro de la molécula formando el poro hidrofílico o canal. Dentro de estas interacciones se encuentran las fuerzas de van der Waals. Los aminoácidos con cadenas laterales que contienen átomos de hidrógeno unidos a átomos de oxígeno o nitrógeno, como los grupos alcohol de serina y treonina, pueden formar puentes de hidrógeno con átomos ricos en electrones, como el oxígeno del grupo carboxilo o el oxígeno del grupo carbonilo del enlace peptídico. La formación de los puentes de hidrógeno entre los grupos polares en la superficie de la proteína y el solvente acuoso que la contiene, incrementa su estabilidad. Los aminoácidos con cadenas laterales que contienen átomos de hidrógeno unidos a átomos de oxígeno o nitrógeno, como los grupos alcohol de la serina y treonina, pueden formar puentes de hidrógeno con átomos ricos en electrones, como el oxígeno del grupo carboxilo o bien el oxígeno del grupo carbonilo del enlace peptídico. La formación de los puentes de hidrógeno entre los grupos polares en la superficie de la proteína y el solvente acuoso que la contiene, aumenta la estabilidad. Por otro lado, los grupos cargados negativamente que se encuentran en las cadenas laterales de algunos aminoácidos como el grupo carboxilo en la cadena lateral del aspartato o glutamato, pueden interaccionar con cadenas laterales cargadas positivamente como el grupo  $\epsilon$  amino de la lisina.

### 7.6.4. Estructura cuaternaria

Las proteínas también pueden contener múltiples subunidades polipeptídicas. Las *estructuras cuaternarias* de las proteínas comprenden desde dímeros sencillos hasta grandes complejos. Una proteína con varias subunidades se conoce como *multímero* o *proteína multimérica*. Cuando un multímero tiene

<sup>5</sup>Structural Classification Of Proteins. <http://scop2.mrc-lmb.cam.ac.uk>

<sup>6</sup>Pfam Database. <http://pfam.xfam.org>

<sup>7</sup>SUPERFAMILY Database. <http://supfam.org/SUPERFAMILY>

sólo unas pocas subunidades se denomina *oligómero*. Si un multímero está formado por varias subunidades diferentes, la estructura global puede ser asimétrica y bastante compleja. En otros casos los multímeros pueden tener subunidades idénticas o grupos repetidos de subunidades no idénticas dispuestos simétricamente. Muchas proteínas multiméricas tienen funciones reguladoras. En otros casos, subunidades diferentes pueden realizar funciones separadas aunque relacionadas, tales como la catálisis y la regulación.

## 7.7. Métodos empíricos para el estudio de macromoléculas

Existen diversos métodos experimentales para estudiar la estructura de las biomoléculas a nivel atómico. Fundamentalmente se utilizan las técnicas de difracción de rayos X, resonancia magnética nuclear y criomicroscopía electrónica de transmisión. La *difracción de rayos X* precisa de la preparación de cristales de proteínas. A pesar de que la obtención de cristales de proteína suele ser un proceso laborioso, ésta es la técnica preferentemente utilizada para resolver estructuras de proteínas, sobre todo las de gran tamaño. La *resonancia magnética nuclear* (RMN) permite la resolución de proteínas en disolución y de pequeño tamaño, ya que las medidas experimentales son difíciles o imposibles para moléculas grandes. La *crio-microscopía electrónica*, como cualquier método óptico, no tiene suficiente resolución para resolver a nivel atómico una estructura, pero puede proporcionarnos una buena imagen de la disposición subcelular de las moléculas. La *microscopía de electrones* es eficaz para estructuras muy grandes, como supercomplejos de proteínas.

### 7.7.1. Cristalización

Para iniciar un proyecto de determinación estructural mediante cristalográfia y difracción de rayos X primero necesitamos *cristalizar la proteína*. Pero para ello antes debemos purificar la proteína en una relativa gran cantidad (miligramos) y con una *alta pureza y homogeneidad*. La presencia de agregados, diferentes oligómeros o cierta inestabilidad en la proteína pueden dificultar su cristalización. Estos detalles son críticos para que la cristalización sea un éxito. Para evaluar estos posibles problemas podemos realizar medidas de *Dynamic Light Scattering* (DLS) para detectar posibles formas oligoméricas o agregados en la muestra a cristalizar o bien medidas de *Differential Scanning Fluorometry* (DSF) que permiten caracterizar la estabilidad de la muestra en diferentes tampones y la presencia de diferentes ligandos o cofactores. Otros métodos espectroscópicos como el *dicroísmo circular* (CD) también pueden dar información a este respecto. Por otro lado, métodos de purificación adicionales mediante *cromatografía en columna* o *ultracentrifugación* pueden ser útiles para una mejor caracterización de la proteína si fuera necesario. Es importante tener en cuenta que la preparación de la muestra para un experimento de cristalografía es mucho más crítica que para un experimento de caracterización bioquímica. En este segundo caso generalmente se exige como criterio de calidad una pureza parcial y que la proteína mantenga su actividad.

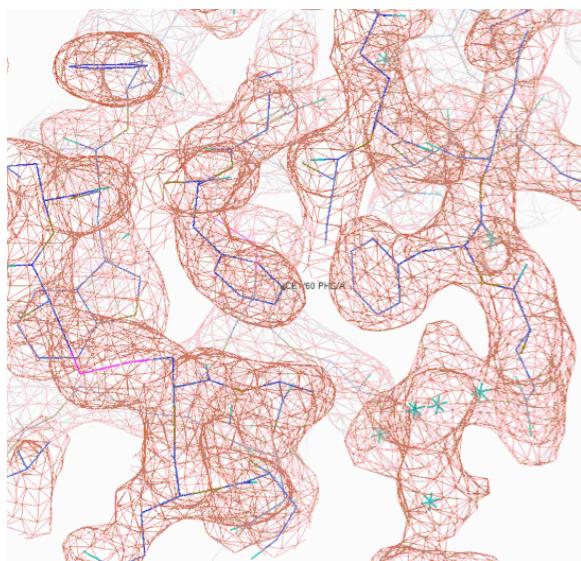
La cristalización de una proteína está controlada por las leyes de la termodinámica, por lo tanto es un proceso semejante al de cualquier sal o mineral. En ambos casos se necesita llevar la solución a un estado de *sobresaturación* después del cual la proteína (o la sal) comenzará a cristalizar. Sin embargo, los métodos empleados en uno y otro caso son diferentes. Por ejemplo, en el caso de una sal como CaSO<sub>4</sub> podríamos conseguir una solución sobresaturada calentando a 40-45 °C y luego después de dejarla enfriar a temperatura ambiente durante un tiempo precipitarían los cristales en el fondo del recipiente. En el caso de las proteínas el calentamiento no es un método adecuado porque las proteínas rápidamente se desnaturalizan a altas temperaturas (con la excepción de proteínas procedentes de

organismos extremófilos). La solubilidad de una proteína no sólo depende de la temperatura, además depende de la concentración, tipo de sal presente en el tampón, pH del tampón, presencia de posibles cofactores. El número de técnicas descritas para cristalizar una proteína es elevado pero sólo unas pocas se usan habitualmente. La más común se basa en alcanzar la sobresaturación mediante *difusión de vapor*. Esto se puede conseguir bien mediante la modalidad de “*gota colgante*”, o mediante “*gota posada*”. Otra estrategia menos usada es la de *diálisis*.

### 7.7.2. Difracción de rayos X

La *difracción de rayos X* es una de las técnicas más potentes para la determinación estructural de proteínas, pues proporciona una fotografía tridimensional a escala atómica del material cristalizado, que incluye su plegamiento, distancias y ángulos de enlace, empaquetamiento cristalino, etc. En definitiva proporciona datos para entender la función biológica de las proteínas.

Un requisito indispensable para abordar un estudio estructural mediante esta técnica es el de *disponer de cristales únicos, homogéneos y de buena calidad*. Cuando un haz de rayos X incide sobre un cristal interacciona con los electrones de los átomos que lo componen, haciéndoles vibrar acopladamente con las variaciones periódicas de su campo eléctrico. De esta manera, los electrones al vibrar se convierten en focos de una nueva radiación X que se emite de forma esférica. Este fenómeno se denomina *dispersión*. La desviación de los rayos X dispersados es mayor cuanto mayor es la densidad electrónica de la materia sobre la que inciden. Los rayos X dispersados por los electrones pueden interferir entre sí dando lugar a cancelaciones, o bien pueden combinarse y reforzarse en determinadas direcciones si están en fase, dando lugar a una dispersión cooperativa conocida como *difracción*. Para que se produzcan todas las difracciones posibles los cristales deben girarse de tal modo que todos los planos virtuales se coloquen en una disposición adecuada para cumplir la *ley de Bragg*. Sólo los haces difractados pueden ser detectados en la película fotográfica o en el detector. Durante la difracción de rayos X el haz incidente produce calor y radicales libres, por lo que es importante enfriar el cristal durante el proceso de difracción, con el fin de evitar daños irreversibles en el cristal y por tanto en la proteína.



**Figura 7.13:** Zona del mapa de densidad electrónica de un cristal de proteína<sup>8</sup>.

<sup>8</sup>Crystallographic X-ray Facility at the Department of Biochemistry, University of Cambridge. <http://www.xray>.

Los rayos X difractados por el cristal suministran una imagen en la que existe una disposición regular de manchas punteadas. El *patrón global de reflexiones y sus intensidades permite la construcción de un mapa de densidad electrónica de la proteína*, que tiene un aspecto similar al de un mapa topográfico. A partir del mapa de densidad electrónica se puede representar el trazado de las uniones entre átomos mediante distintos métodos. Sin embargo, para poder calcular la función de densidad electrónica, y por lo tanto poder saber la localización de los átomos en el interior de la celdilla, necesitamos conocer también el desfase entre las ondas, pero esta información se pierde durante el proceso de medida experimental, ya que no existen técnicas experimentales para medir esos desfases. Para resolver el “*problema de las fases*” Arthur Lindo Patterson (1934) propuso una fórmula, basada en métodos de Transformada de Fourier, conocida como la *función de Patterson*, que permite una simplificación de la información contenida en la función de densidad electrónica, pues suprime la información de las fases. La información que proporciona la función de Patterson es un mapa de vectores de posición entre átomos (posiciones relativas). Posteriormente, a lo largo del siglo XX varios autores, entre ellos Jerome Karle y Herbert Hauptmann (Premio Nobel de Química, 1985), propusieron diferentes métodos, métodos directos, para resolver el problema en cristales formados por moléculas de tamaño pequeño y medio. En cristales que contienen moléculas grandes, proteínas o enzimas, el problema de la fase puede resolverse mediante tres métodos, dependiendo del caso: i) *método de Reemplazo Isomorfo Múltiple* (MIR, del inglés, *Multiple Isomorphous Replacement*) basado en el método de Patterson; ii) *método de Difracción Anómala Múltiple* (MAD, del inglés *Multi-wavelength Anomalous Diffraction*); iii) *método de Reemplazo Molecular* (MR, del inglés *Molecular Replacement*), haciendo uso de un modelo estructural de una proteína homóloga, previamente determinada. Por último, el modelo estructural obtenido debe ser validado, es decir, debe ser consistente con los criterios químicos, no presentar impedimentos estéricos, coherencia en las longitudes y ángulos de enlaces, etc. Ver ejemplo de mapa de densidad electrónica en la Figura 7.13.

La cristalográfia-difracción de rayos X es un excelente método para determinar las estructuras de las proteínas rígidas que forman cristales ordenados. Las proteínas flexibles o con regiones desordenadas o dúctiles son difíciles o imposibles de estudiar por este método, pues la cristalográfia requiere tener muchas moléculas alineadas exactamente en la misma orientación. Las regiones desordenadas o dúctiles son invisibles en los mapas de densidad electrónica pues presentan diversas orientaciones y su densidad electrónica se expande por un espacio grande. La exactitud de la estructura atómica determinada depende de la calidad de los cristales. En cristales perfectos, tenemos mucha más confianza que la estructura atómica resuelta refleje correctamente la estructura de la proteína. Dos medidas importantes de la exactitud de una estructura son su *resolución*, que mide la cantidad de detalle que puede ser visto en los datos experimentales, y el *valor de R*, que mide cómo el modelo atómico se ajusta a los datos experimentales.

### 7.7.3. Resonancia magnética nuclear

Al igual que la difracción de rayos X, la *resonancia magnética nuclear* (NMR del inglés *nuclear magnetic resonance*) puede determinar las posiciones de cada uno de los átomos que constituyen una molécula de proteína. La RMN requiere que la proteína se encuentre en disolución, normalmente acuosa. Además, la NMR aporta no solo información estructural, sino también dinámica, porque permite observar ligeros cambios de conformación. Es una técnica muy adecuada para el *estudio de proteínas con regiones flexibles* o para *proteínas intrínsecamente desordenadas o dúctiles* donde la difracción de rayos X tiene una gran limitación. Pero no todo es posible con NMR ya que tiene una limitación según el tamaño de la proteína. En la actualidad el *tamaño máximo de las proteínas analizables por NMR está en torno a los*

60-80 kDa, aunque este límite va aumentando a medida que se producen avances en la técnica. Como en el caso de la difracción de rayos X, hay que preparar la muestra en unas condiciones óptimas para un su estudio. Es decir, hay que tener en cuenta la concentración de proteína, pureza, características del disolvente, el pH, la temperatura, etc. Las condiciones deben ser las que mejor conserven la estructura nativa de la proteína y que a su vez permitan la adquisición de espectros de NMR.

La técnica de RNM permite detectar la energía que absorben los núcleos de los átomos que entran en resonancia al ser irradiados con radiofrecuencias en el seno de un campo magnético. Cada tipo de núcleo resonante que se puede encontrar en una proteína (<sup>1</sup>H, <sup>13</sup>C, <sup>15</sup>N, <sup>31</sup>P) absorbe energía en una región de radiofrecuencias específica. Sin embargo, no todos los átomos del mismo tipo (por ejemplo, <sup>1</sup>H) que hay en una molécula absorben energía exactamente a la misma frecuencia de radiación; existen pequeñas variaciones que dependen de la densidad electrónica del entorno químico que lo rodea. Por tanto, cada núcleo <sup>1</sup>H contenido en una proteína resuena a un valor intrínseco de radiofrecuencia, según el entorno en que se encuentra. Teniendo en cuenta este principio se ha desarrollado la técnica de *NMR bidimensional*, idónea para resolver la estructura tridimensional de proteínas. Esta técnica permite identificar núcleos <sup>1</sup>H muy próximos (a menos de 5 Å), debido a un fenómeno denominado *efecto Overhauser nuclear* (NOE). El NOE consiste en que el acoplamiento de los campos magnéticos entre núcleos próximos conlleva a una variación de la radiofrecuencia con las que resuenan. La NMR bidimensional proporciona lo que se denomina un espectro bidimensional, gráficamente una nube de puntos entre dos ejes. Los puntos que están fuera de la diagonal permiten identificar pares de protones separados por menos de 5 Å. La interpretación de un espectro bidimensional NMR en una estructura tridimensional es un proceso complicado que precisa de equipos informáticos con adecuados programas gráficos. El programa de ordenador precisa de información sobre los parámetros químicos, tales como la estructura primaria de la proteína, longitudes y ángulos de enlaces, radios de van der Waals, etc. Entonces, el ordenador genera una familia de estructuras que representan una gama de conformaciones consistentes con las proximidades atómicas detectadas por NOE.

## 7.8. Herramientas bioinformáticas básicas para el estudio de macromoléculas

### 7.8.1. Protein Data Bank

El repositorio “*Protein Data Bank*” (PDB)<sup>9</sup> contiene los ficheros de las coordenadas atómicas e información relevante adicional que describe a las proteínas cuya estructura 3D se ha determinado por cristalográfia de rayos X, espectroscopia NMR o crio-microscopía electrónica de transmisión. Estos ficheros contienen un listado de los átomos de cada proteína y su localización en el espacio. El formato típico de un *fichero PDB* incluye un texto al inicio, que resume la descripción de la proteína y los detalles de la resolución de su estructura, seguido por la secuencia y una larga lista de los *átomos* y sus *coordenadas*. El archivo también contiene información sobre las condiciones experimentales que han sido usadas para determinar estas coordenadas atómicas.

### 7.8.2. Visualización de estructuras en 3D

Las estructuras 3D de biomoléculas pueden ser visualizadas con diferentes programas informáticos disponibles, tanto libres como de licencia comercial. Entre ellos podemos destacar PyMOL<sup>10</sup>, Visual

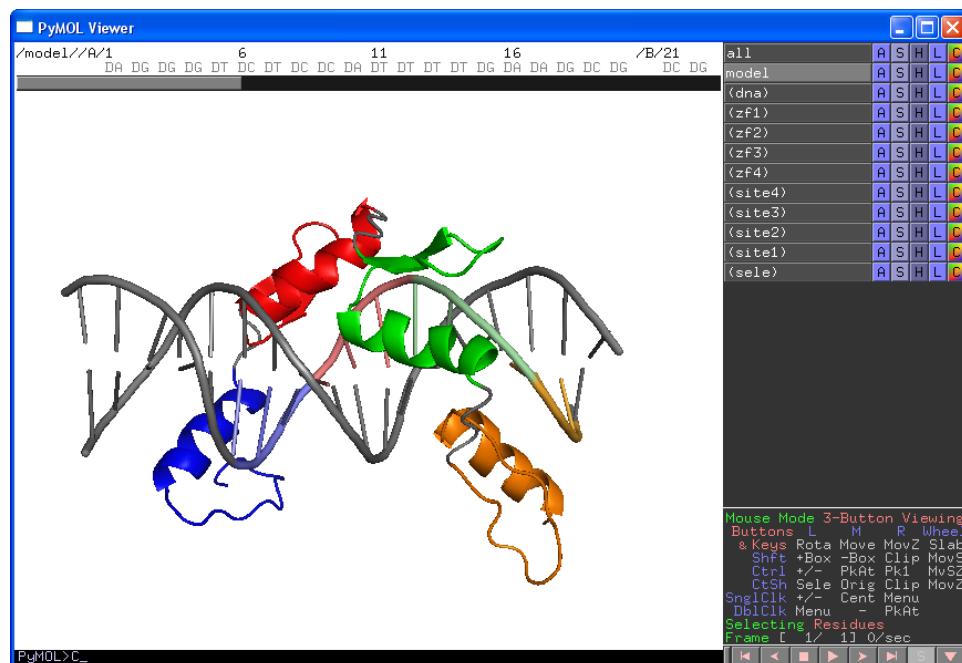
<sup>9</sup>Protein Data Bank. <http://www.rcsb.org>

<sup>10</sup>PyMOL. <http://www.pymol.org>

Molecular Dynamics<sup>11</sup> (VMD), UCSF Chimera<sup>12</sup>, DeepView<sup>13</sup> (old Swiss-PdbViewer), Jmol<sup>14</sup>, RasMol<sup>15</sup> o Protein Explorer<sup>16</sup> entre otros. Estos programas permiten visualizar una estructura molecular si conocemos sus coordenadas atómicas (generalmente en un archivo de formato texto). Existen dos tipos de programas de visualización molecular por ordenador, los que funcionan independientemente de las páginas web y los que funcionan como *plugins* adaptados a las páginas web y que siempre requieren páginas web para funcionar. A continuación comentaremos las aplicaciones de dos programas: PyMol y Swiss-Pdb Viewer, una lista más exhaustiva puede consultarse en Wikipedia<sup>17</sup>.

## PyMol

*PyMOL*<sup>10</sup> se puede instalar en nuestro ordenador con licencia o en su versión académica libre. Puede trabajar con una amplia variedad de formatos diferentes desde ficheros PDB a ficheros multi-SDF o mapas de densidad electrónica. PyMOL es fácil de usar y permite representar las estructuras en una gran variedad de formas (ej. esferas, superficie, vistas volumétricas, líneas, enlaces poniendo énfasis en la conectividad, elementos de estructura secundaria, entre otros). También permite la superposición de estructuras. Ejemplo en Figura 7.14.



**Figura 7.14:** Ejemplo de visualización de un complejo proteína-DNA con PyMOL.

<sup>11</sup>Visual Molecular Dynamics. <http://www.ks.uiuc.edu/Research/vmd>

<sup>12</sup>UCSF Chimera. <http://www.cgl.ucsf.edu/chimera>

<sup>13</sup>DeepView. <http://spdbv.vital-it.ch>

<sup>14</sup>Jmol. <http://jmol.sourceforge.net>

<sup>15</sup>RasMol. <http://rasmol.org/>

<sup>16</sup>Protein Explorer. <http://proteinexplorer.org>

<sup>17</sup>Software for protein structure visualization. [http://en.wikipedia.org/wiki/Software\\_for\\_protein\\_structure\\_visualization](http://en.wikipedia.org/wiki/Software_for_protein_structure_visualization)

## **DeepView**

*DeepView*<sup>13</sup> se puede instalar en el ordenador con licencia y es una aplicación que proporciona una interfaz de fácil manejo para visualizar y analizar proteínas al mismo tiempo. La proteínas se pueden superponer con el fin de deducir alineamientos estructurales y comparar sus sitios activos u otras regiones relevantes. Mutaciones de aminoácidos, puentes de hidrógeno, ángulos y distancias entre átomos son fáciles de determinar mediante su interfaz gráfica. DeepView está conectado con SWISS-MODEL<sup>18</sup>, un servidor de modelado estructural automático. El uso combinado de estos dos programas facilita el trabajo para generar modelos estructurales. DeepView también lee mapas de densidad electrónica y proporciona varias herramientas para construir mapas de densidad. Además integra varias herramientas de modelado y mutación de aminoácidos.

---

<sup>18</sup>SWISS-MODEL. <http://swissmodel.expasy.org>

## 7.9. Bibliografía

- [1] G. W. Beadle and E. L. Tatum. Genetic control of biochemical reactions in *neurospora*. *Proc Natl Acad Sci U S A*, 27(11):499–506, 1941.
- [2] I. H. G. S. Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–45, 2004.
- [3] F. H. Crick, L. Barnett, S. Brenner, and R. J. Watts-Tobin. General nature of the genetic code for proteins. *Nature*, 192:1227–32, 1961.
- [4] M. Grunberg-Manago, P. J. Ortiz, and S. Ochoa. Enzymatic synthesis of nucleic acidlike polynucleotides. *Science*, 122(3176):907–10, 1955.
- [5] H. G. Khorana. Polynucleotide synthesis and the genetic code. *Fed Proc*, 24(6):1473–87, 1965.
- [6] P. Leder and M. W. Nirenberg. Rna codewords and protein synthesis, 3. on the nucleotide sequence of a cysteine and a leucine rna codeword. *Proc Natl Acad Sci U S A*, 52:1521–9, 1964.
- [7] J. H. Matthaei, O. W. Jones, R. G. Martin, and M. W. Nirenberg. Characteristics and composition of rna coding units. *Proc Natl Acad Sci U S A*, 48:666–77, 1962.
- [8] F. Sanger and A. R. Coulson. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *J Mol Biol*, 94(3):441–8, 1975.
- [9] F. Sanger and H. Tuppy. The amino-acid sequence in the phenylalanyl chain of insulin. i. the identification of lower peptides from partial hydrolysates. *Biochem J*, 49(4):463–81, 1951.
- [10] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–8, 1953.



# Capítulo 8

## Análisis de secuencias biológicas

*Álvaro Sebastián*

### 8.1. Introducción

Las secuencias son a la biología como las palabras al lenguaje. Normalmente representamos complejos polímeros orgánicos como el DNA o las proteínas usando largas concatenaciones de letras que simplifican su estructura química y nos permiten visualizarlos de forma similar a como leemos este libro. Al igual que las palabras, las secuencias pueden tener una raíz común y compartir ‘significado’ con otras de la misma familia. Por ello es muy importante conocer las herramientas disponibles para comparar nuevas secuencias con otras ya conocidas y encontrar similitudes que nos ayuden a comprender su función, estructura o evolución, en resumen, el significado biológico de la secuencia.

#### 8.1.1. Homología de secuencias

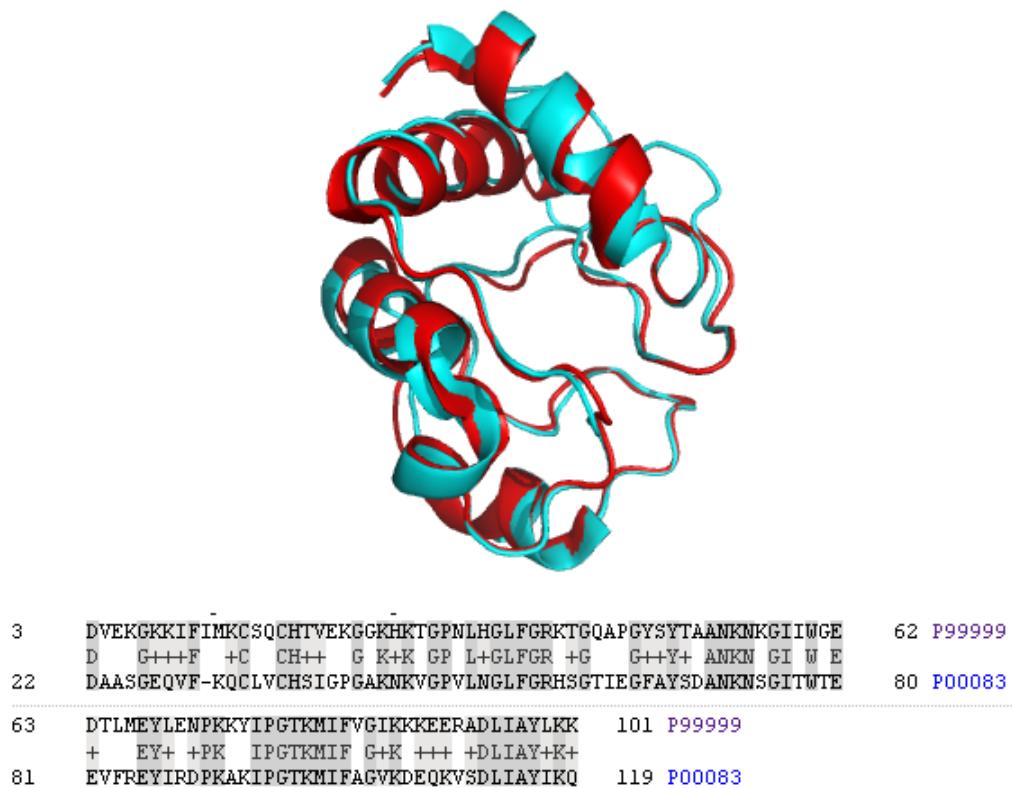
Dos secuencias que comparten un ancestro común se denominan *secuencias homólogas*. Dos proteínas homólogas provienen de dos genes homólogos y comparten la misma estructura tridimensional y la misma función biológica (salvo raras excepciones). A veces se confunde homología con similitud, pero la *homología es cualitativa* y sólo admite dos estados: o ser homólogo o no serlo, y la similitud en cambio es cuantificable. Sin embargo, el grado de similitud entre dos secuencias nos puede servir para inferir la existencia o no de homología entre ambas. Actualmente la comparación de secuencias es la herramienta más poderosa para entender la función biológica de una secuencia desconocida mediante la búsqueda en bases de datos de secuencias homólogas previamente caracterizadas. Para muchas proteínas se pueden encontrar homólogos que tuvieron un ancestro común hace 1000-2000 millones de años y que siguen compartiendo características en común como su estructura tridimensional o sitios activos como por ejemplo las hemoglobinas de plantas y animales o el citocromo c de procariotas y eucariotas [14]. En la Figura 8.1A se pueden comparar las estructuras superpuestas del citocromo c humano y el citocromo c2 de la bacteria *Rhodopseudomonas viridis*, así como sus secuencias proteicas alineadas (Figura 8.1B) observando una clara homología.

La predicción de homología se realiza extrayendo la información conservada durante la evolución de las secuencias a comparar. La forma más sencilla de realizar la comparación sería escribiendo cada secuencia en una línea y anotando los nucleótidos o aminoácidos comunes. Sin embargo, las secuencias sufren

cambios y mutaciones que impiden su comparación directa con otras secuencias homólogas. Como veremos más adelante en el capítulo, las secuencias a comparar deberán ser alineadas para posteriormente poder puntuar los residuos conservados en posiciones equivalentes. Finalmente las puntuaciones obtenidas deberán ser validadas estadísticamente para poder garantizar con una máxima fiabilidad que existe o no homología.

*¿Por qué comparar secuencias y no estructuras de proteínas?* Porque la similaridad de estructuras no garantiza que exista homología. El parecido estructural puede ser debido a restricciones químicas y físicas que favorecen un determinado plegamiento proteico. Existen numerosas excepciones de proteínas con estructuras similares que no son homólogas y de hecho sus secuencias son muy diferentes [10]. Además están descritas muchas menos estructuras que secuencias en las bases de datos y la comparación de estructuras es más compleja y con mayor costo computacional como se verá en el Capítulo 10. Sin embargo, dos proteínas homólogas poseerán estructuras similares.

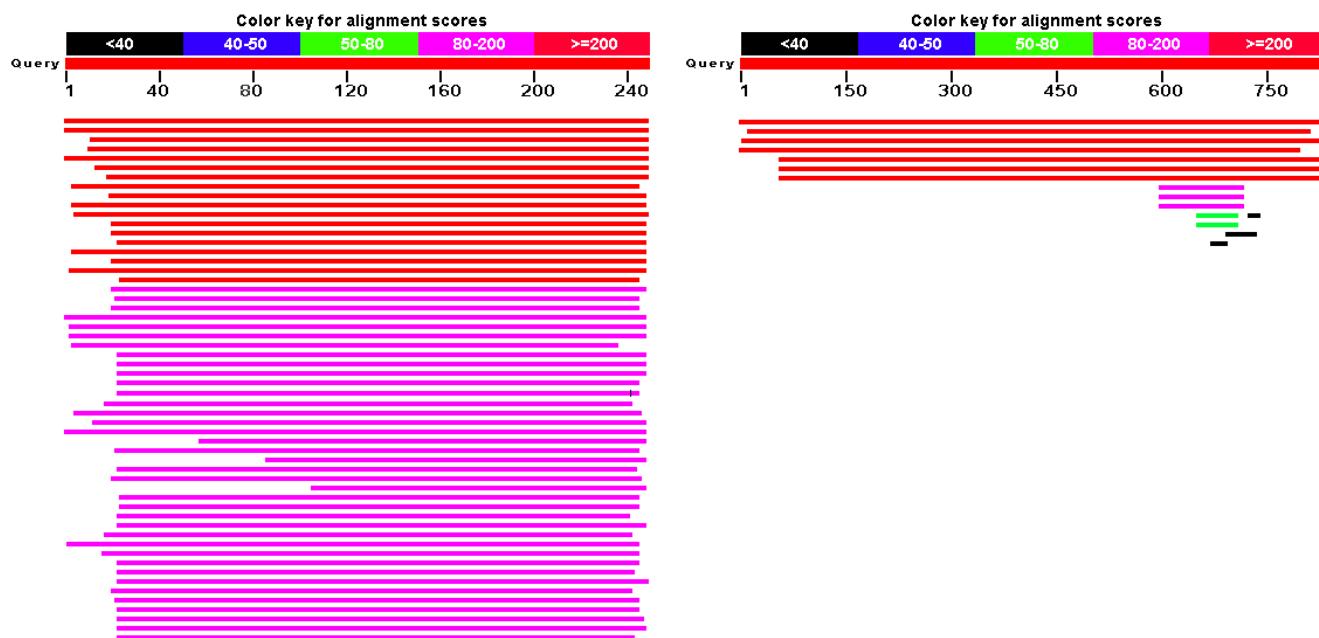
Finalmente, añadir que otro gran indicador de homología es la función de una proteína, incluso mejor que la secuencia. Sin embargo, *¿por qué no se usa la función para predecir homología?* Porque no se conoce la función real de la mayoría de proteínas anotadas en las bases de datos. Aunque muchas proteínas tienen una función anotada, ésta ha sido predicha comparando su secuencia con otras de función conocida o también predicha, por ello no es un dato fiable.



**Figura 8.1:** A: Estructuras tridimensionales del citocromo c humano y el citocromo c2 de la bacteria *Rhodopseudomonas viridis* superpuestas. B: Secuencias del citocromo c humano y el citocromo c2 de la bacteria *Rhodopseudomonas viridis* alineadas.

### 8.1.2. Diferencias entre el alineamiento de DNA y de proteínas

La mayoría de algoritmos y técnicas de alineamiento pueden aplicarse tanto a proteínas como DNA o RNA. Sin embargo *las comparaciones de secuencias de nucleótidos suelen ser menos precisas e informativas*. Veamos como ejemplo el resultado de alinear las secuencias de proteína (NP\_002760.1) y de mRNA (NM\_002769.4) de la tripsina humana con el programa BLAST contra las bases de datos de referencia del NCBI con los parámetros por defecto y buscando únicamente secuencias humanas. En la Figura 8.2A se observa como la búsqueda de secuencias de proteínas homólogas devuelve 108 resultados, todos ellos de calidad buena o muy buena (colores rosa y rojo respectivamente), por contra la búsqueda de la secuencia de cDNA obtiene tan sólo 15 resultados, de los cuales sólo 7 son de una calidad aceptable como para considerar su homología. La explicación a este fenómeno la podemos encontrar en la degeneración del código genético que permite la sustitución de nucleótidos en el DNA sin cambiar el aminoácido que codifican, lo cual hace que si no se conoce el marco de lectura, no se consigan encontrar secuencias estadísticamente significativas en las búsquedas. Por otro lado, las sustituciones de aminoácidos en las proteínas suelen ser por otros con similares propiedades físico-químicas. La probabilidad de dichas sustituciones está tabulada, lo que permite asignar puntuaciones a los alineamientos y con ello una mayor precisión (ver Subsección 8.3.2). En general se puede decir que la búsqueda de secuencias de nucleótidos es únicamente válida para buscar secuencias homólogas muy conservadas, que divergieron hace menos de 600000 años. La búsqueda de homólogos proteicos es mucho más sensible, pudiendo encontrar proteínas con ancestros comunes de más de 1000 millones de años [13]. Una posible solución a este problema es traducir las secuencias de DNA o RNA a secuencias peptídicas siempre que se conozca el marco de lectura, tal como se explicó en la Sección 7.4. Sin embargo, la búsqueda de secuencias de nucleótidos sigue siendo válida en muchas otras ocasiones como ensamblajes de secuencias, búsquedas de polimorfismos, diagnósticos genéticos o búsquedas de sitios de unión de factores de transcripción.



**Figura 8.2:** Resultados estadísticamente significativos tras alinear las secuencias de proteína (A) y de mRNA (B) de la tripsina humana con el programa BLAST contra las bases de datos de referencia del NCBI.

## 8.2. Obtención de secuencias y formatos

Elegir la base de datos adecuada donde buscar una secuencia problema es tan importante como escoger una óptima técnica de alineamiento. Por ejemplo, si queremos buscar proteínas homólogas a una humana en ratón, será inútil que consultemos una base de datos de secuencias de plantas. Por ello, aunque ya se han presentado las bases de datos de secuencias más importantes en el Capítulo 1, volveremos a recordarlas brevemente.

### 8.2.1. Bases de datos de secuencias de DNA y proteínas

Cuando se trata de buscar *secuencias de proteínas*, la base de datos mejor anotada actualmente es *UniProt*<sup>1</sup> [3]. Esta base de datos recoge actualizaciones de proteínas de la antigua base de datos anotada manualmente llamada Swiss-Prot y también anotaciones automáticas (TrEMBL). A su vez permite descargarse proteomas completos de diversos organismos, así como buscar secuencias de proteínas automáticamente desde su interfaz web. Las anotaciones de las proteínas revisadas suelen ser muy precisas, incluyendo datos como nomenclatura, función, dominios proteicos, sitios catalíticos, variantes proteicas, bibliografía, secuencia, así como enlaces a otras bases de datos con el gen que las codifica, el mRNA o la estructura tridimensional.

En el caso de buscar secuencias de nucleótidos, quizás la más completa es *GenBank*<sup>2</sup> [2] mantenida por el organismo americano NCBI. Su mayor problema es su gran tamaño, en enero de 2011 contenía más de 135 millones de secuencias sin incluir secuencias de proyectos de secuenciación incompletos. Al igual que UniProt permite descargar archivos con secuencias e integra la herramienta BLAST para realizar búsquedas. Si nos interesa una mejor anotación y menor número de secuencias deberíamos usar *RefSeqGene*<sup>3</sup>, donde podemos encontrar información anotada sobre los genes, sus variantes de mRNA y las proteínas que codifican, además de su posición en el genoma, fenotipos, función, interacciones, etc. La característica más interesante de RefSeqGene es que permite obtener las secuencias tanto de nucleótidos como de péptidos de las diferentes variantes de *splicing* de un gen (ver Sección 7.2). Sin embargo, en RefSeqGene existen muchas menos secuencias anotadas que en GenBank.

### 8.2.2. Formatos de archivos de secuencias

El formato más usado en el campo del análisis de secuencias es el formato *FASTA*<sup>4</sup>. Consiste en archivos de texto donde la primera línea antes de cada secuencia, también llamada cabecera, comienza por el carácter ‘>’ seguido del identificador de la secuencia. Tras dicha línea siguen otras con la secuencia escrita con código de una letra por cada nucleótido o aminoácido (Ver Sección 7.3 y Sección 7.6) usándose guiones ‘-’ para representar huecos en la secuencia. La longitud de cada línea de secuencia se recomienda que no sea superior a 80 caracteres para facilitar su lectura. En la Figura 8.3A se puede ver un ejemplo de secuencia en formato FASTA.

Otro formato con una anotación más completa es el formato *GenBank*<sup>5</sup>. En este formato el tipo de contenido almacenado en cada línea está indicado por una palabra identificadora en letras mayúsculas colocada al inicio de la línea y seguida de un número fijo de espacios hasta el comienzo de la información. Los campos de cabecera son: LOCUS, DEFINITION, ACCESSION, VERSION, GI, KEYWORDS,

<sup>1</sup>UniProt. <http://www.uniprot.org/>

<sup>2</sup>GenBank. <http://www.ncbi.nlm.nih.gov/genbank/>

<sup>3</sup>RefSeqGene. <http://www.ncbi.nlm.nih.gov/refseq/rsg/>

<sup>4</sup>Descripción del formato FASTA. [http://en.wikipedia.org/wiki/FASTA\\_format](http://en.wikipedia.org/wiki/FASTA_format)

<sup>5</sup>Descripción del formato GenBank. <http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord>

SOURCE y REFERENCE, a su vez cada uno de ellos puede tener contenidos otros, por ej. AUTHORS, TITLE y JOURNAL dentro del campo REFERENCE. El campo FEATURES y sus subcampos ‘source’, ‘genes’ y ‘CDS’ contienen la información sobre el organismo, la localización cromosómica, así como del inicio y final de las regiones génicas y codificantes, incluyendo en su caso el nombre, identificadores y otros datos del gen o de la proteína y la secuencia codificada. Finalmente el campo ORIGIN incluye la secuencia a la que pertenecen las anotaciones anteriores, escrita en código de una letra y numerada. En la Figura 8.3B se puede ver un ejemplo de secuencia en formato GenBank.

A

```
>gi|109148525|ref|NM_000207.2| Homo sapiens insulin (INS), transcript variant 1, mRNA
AGCCCTCAGGACAGGCTGCATCAGAAAGAGGCCATCAAGCAGATCACTGTCCITCTGCCATGCCCTGTG
GATGCCCTCCTGCCCTGCTGGCGCTGCTGGCCTCTGGGGACCTGACCCAGCCGACGCCCTTGTAAC
CAACACCTGTGGCCTCACACCTGGTGAAGCTCTACCTAGTGTGCGGGAAACGAGGCTTCTTCTACA
CACCCAAGACCCGCCGGAGGCAGAGGACCTGCAGGTGGGCAGGTGGAGCTGGCGGGCCCTGGTGC
AGGCAGCCTGCAGCCCTGGCCCTGGAGGGTCCCTGCAGAAGCGTGGCATTTGGAACAATGCTGTACC
AGCATGTGCTCCCTACACAGCTGGAGAACTACTGCAACTAGACGCAGCCCGCAGGCAGCCCCAACACCG
CCGCCTCTGCACCGAGAGAGATGGAATAAGCCCTGAAACCAGCAAA
```

B

LOCUS	NM_000207	469 bp	mRNA	linear	PRI	17-JUN-2012
DEFINITION	Homo sapiens insulin (INS), transcript variant 1, mRNA.					
ACCESSION	NM_000207					
SOURCE	Homo sapiens (human)					
ORGANISM	Homo sapiens					
REFERENCE	1 (bases 1 to 469)					
AUTHORS	Ley, S.H., Hanley, A.J., Sermer, M., Zinman, B. and O'Connor, D.L.					
TITLE	Associations of prenatal metabolic abnormalities with insulin and adiponectin concentrations in human milk					
JOURNAL	Am. J. Clin. Nutr. 95 (4), 867-874 (2012)					
FEATURES	Location/Qualifiers					
source	1..469 /organism="Homo sapiens" /mol_type="mRNA" /chromosome="11" /map="11p15.5"					
gene	1..469 /gene="INS"					
CDS	60..392 /gene="INS" /gene_synonym="IDDM2; ILPR; IRDN; MODY10" /note="proinsulin; preproinsulin" /codon_start=1 /product="insulin preproprotein" /protein_id="NP_000198.1" /translation="MALWMRLPLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCG ERGFIFYTPKTRREAEDLQVGQVELGGPGAGSLQPLALEGSLLQKRGIIVEQQCTSICSL YQLENYCN"					
ORIGIN	<pre> 1 agccctccag gacaggctgc atcagaagag gccatcaagc agatcaactgt ctttctgcca 61 tggccctgtg gatgcgcctc ctgcccctgc tggcgctgtc ggccctctgg ggacctgtacc 121 cagccgcagc ctttgtaac caacacctgt gcggttcaca cctgggtggaa gctctctacc 181 tagtgtgcgg ggaacgaggc ttcttctaca cacccaaagac ccgcggggag gcagaggacc 241 tgcagggtgg gcagggtggag ctggcggggg gcccgtgtc aggccgttg cagcccttgg 301 ccctggaggg gtccctgcag aacgcgtggca ttgtggaaaca atgtgtacc agcatctgt 361 ccctctacca gctggagaac tactgcaact agacgcggcc cgcaggcagc cccacacccg 421 ccgcctccctg caccggagaga gatggataaa agcccttgaa ccagcaaaa //</pre>					

**Figura 8.3:** Secuencia del mRNA del gen de la insulina. A: Formato FASTA. B: Formato GenBank.

### 8.2.3. Ejemplo de código en Perl para leer un archivo FASTA

Para finalizar el apartado, explicaré un pequeño ejemplo escrito en Perl para leer secuencias en formato FASTA y almacenarlas en una tabla hash para poder acceder a ellas fácilmente más tarde en el programa. El código comienza definiendo la variable `$identificador` donde se almacenará la cabecera de la secuencia que se esté leyendo en cada momento y la tabla hash `%secuencias` que almacenará las secuencias con el identificador como índice. Al leer el archivo FASTA línea por línea, se identificarán las cabeceras con una sencilla expresión regular ‘`/>(.*)/`’ y se almacenará en la variable `$identificador` el contenido que sigue al símbolo ‘`>`’. Posteriormente se leerá la secuencia que se irá almacenando en la tabla hash `%secuencias` según se vaya leyendo. Tras leer todo el archivo, se podrá acceder a las secuencias invocando a la tabla con un identificador como índice ‘`$secuencias{“Identificador”}`’.

Código 8.1: Ejemplo de lectura de un archivo FASTA con Perl.

```
1 # Código para leer secuencias FASTA y almacenarlas en una tabla hash
2
3 my (%secuencias, $identificador);
4
5 # Abrir el archivo FASTA de secuencias
6 open(FASTAFILE,"archivo.fasta");
7
8 # Leer linea por linea el archivo
9 while(<FASTA>){
10
11     # Detectar las cabeceras y guardar las secuencias en la tabla hash
12     if(/^>(.*)/){
13         $identificador = $1;
14     } else {
15         $secuencias{$identificador} .= $_;
16     }
17 }
18
19 # Cerrar el archivo FASTA
20 close(INFILE);
21
22 # Imprimir en pantalla una secuencia almacenada en la tabla hash
23 print $secuencias{"Identificador"};
```

## 8.3. Alineamiento de secuencias

Los grandes avances en las tecnologías de secuenciación de DNA, la disponibilidad de grandes bases de datos de secuencias, junto con el desarrollo de eficientes algoritmos de comparación de secuencias han motivado un cambio sustancial en la biología molecular y celular modernas. Actualmente cuando se descubre un nuevo gen no se va directamente al laboratorio a realizar experimentos para conocer su función celular, simplemente se introduce su secuencia en un ordenador esperando encontrar genes homólogos de otros organismos ya estudiados. Tal y como se ha ido explicando a lo largo del capítulo, la forma de encontrar secuencias homólogas en una base de datos es alineando nuestra secuencia problema con las otras secuencias y evaluando el parecido entre ellas. Este proceso no es tan sencillo como poner una secuencia encima de otra, las secuencias tienen diferentes longitudes, inserciones, delecciones, sustituciones... Si volvemos al ejemplo del citocromo c humano y el citocromo c2 de la bacteria *Rhodopseudomonas viridis* (Figura 8.1), probablemente no seamos capaces de alinear manualmente sus secuencias, a pesar de su clara homología. De ahí la importancia de conocer los entresijos de los

algoritmos de alineamiento que nos permiten realizar la tarea de comparación de secuencias de una forma rápida y fiable, siempre que se sepan elegir los parámetros adecuados.

### 8.3.1. Definición de similitud e identidad

Dos definiciones muy importantes a la hora de evaluar la calidad de un alineamiento son la similitud y la identidad. La *identidad* es la suma de residuos idénticos en posiciones equivalentes en dos secuencias alineadas. La *similitud* es la suma de puntuaciones correspondientes a residuos en posiciones equivalentes en dos secuencias alineadas, dichas puntuaciones suelen estar tabuladas e incluir penalizaciones para las inserciones y delecciones (también llamados *gaps*, porque insertan huecos en el alineamiento). Las tablas de puntuaciones de sustitución de un residuo por otro se denominan *Matrices de sustitución*. En la Figura 8.4 podemos ver un ejemplo de alineamiento de 2 secuencias de DNA con sus correspondientes valores de identidad y similitud.

<b>TGAAGTA-ACT</b>	C→C, G→G +2 A→A, T→T +1 A→T, T→A -1 C→G, G→C -2 OTROS -2 GAP -4
<b>TCATGTACACT</b>	
<b>Identidad:</b> 1+0+1+0+1+1+1+0+1+1+1 = <b>8</b>	
<b>Similitud:</b> 1-2+1-1+2+1+1-4+1+2+1 = <b>3</b>	

**Figura 8.4:** Ejemplo de valores de identidad y similitud para un alineamiento de 10 nucleótidos. Las puntuaciones de cada sustitución para calcular la similitud se muestran a la derecha.

### 8.3.2. Matrices de sustitución

La historia de las matrices de sustitución se remonta a los años 70, cuando la investigadora Margaret Oakley Dayhoff se afanaba en recopilar todas las secuencias de proteína existentes en su libro ‘Atlas of Protein Sequence and Structure’ [4]. Dayhoff y colaboradores estudiaron el modelo evolutivo de los cambios en los aminoácidos de las proteínas, para ello estudiaron 1572 cambios en 71 grupos de proteínas, dentro de cada grupo las secuencias compartían más del 85 % de identidad. De esta forma anotaron el número de cambios para todas las combinaciones posibles de 2 aminoácidos, observando que 35 de las posibles mutaciones nunca ocurrían, estas se correspondían con aminoácidos poco frecuentes. También observaron que las mutaciones más frecuentes se daban entre aminoácidos con similares propiedades físico-químicas, como por ej. Asp y Glu. Muchos de los cambios de aminoácido esperados por modificación de un sólo nucleótido en los codones codificantes no se daban o eran infrecuentes, lo que demostró una mayor presión evolutiva a nivel de secuencia proteica que a nivel de DNA.

El cambio de un aminoácido por otro se denominó ‘mutación puntual aceptada’ (*PAM*). Normalizando los datos de las PAMs de acuerdo a la probabilidad de mutación de cada aminoácido en los datos estudiados (*mutabilidad*) se obtuvo la famosa matriz PAM1 en la que cada elemento de la matriz  $M_{ij}$  cuantifica la probabilidad de que un aminoácido  $i$  sea remplazado por otro aminoácido  $j$  en el intervalo evolutivo de 1 PAM. 1 PAM se define como el intervalo evolutivo en que cambia un 1 % de los aminoácidos en el alineamiento de 2 secuencias (1 cambio o PAM por cada 100 aminoácidos).

La matriz PAM1 sirve para simular cambios evolutivos en secuencias de proteínas. Para ello basta tomar un número aleatorio (entre 0 y 1) para cada aminoácido de una secuencia dada y asignarle un cambio si la probabilidad es menor que la anotada en la matriz para conservar el aminoácido. El proceso se puede repetir múltiples veces hasta alcanzar la distancia PAM deseada. Las matrices PAM también tienen unas propiedades muy interesantes: *i*) la matriz PAM0 sólo posee unos en la diagonal y el resto son ceros; *ii*) la matriz se puede multiplicar por sí misma para calcular matrices de N PAMs; *iii*) si la matriz se multiplica infinitas veces por sí misma obtendremos la frecuencia del aminoácido  $j$  para todas las columnas de  $i$ .

Los intervalos evolutivos medidos en PAMs los podemos relacionar con porcentajes de residuos conservados idénticos por medio de la fórmula:

$$\text{Identidad} (\%) = 100 \sum f_i M_{ii} \quad (8.1)$$

Siendo  $f_i$  la frecuencia normalizada de aparición de un aminoácido y  $M_{ii}$  el valor en la diagonal de la matriz PAM. Algunas equivalencias calculadas entre identidad y PAMs se pueden consultar en la Tabla 8.1.

PAMs	1	5	11	23	38	56	80	112	159	195	246	328
Identidad (%)	99	95	90	80	70	60	50	40	30	25	20	15

**Tabla 8.1:** Tabla de equivalencias entre PAMs y porcentaje de identidad entre secuencias proteicas.

Toda la anterior explicación teórica de las matrices PAM está muy bien, pero volviendo al tema de alinear y comparar secuencias, ¿para qué nos sirven las matrices PAM? Las matrices PAM no nos son útiles directamente, pero sí el *odd-ratio* ( $R_{ij}$ ) calculado dividiendo un elemento de la matriz  $M_{ij}$  entre la frecuencia normalizada de  $j$  ( $f_j$ ):

$$R_{ij} = \frac{M_{ij}}{f_j} \quad (8.2)$$

$M_{ij}$  nos da la probabilidad de que un aminoácido  $i$  sea sustituido por otro  $j$  en una distancia evolutiva definida por la matriz PAM y  $f_j$  es la probabilidad de encontrar el aminoácido  $j$  en una posición de la secuencia por casualidad. El *odd-ratio*  $R_{ij}$  cuantifica la probabilidad de que una sustitución se dé en una posición dada. Un *odd-ratio* de valor 10 significaría que la sustitución es 10 veces más frecuente que la probabilidad de encontrar alineados ambos aminoácidos. Por el contrario, un *odd-ratio* de valor 0,5 significaría que la probabilidad de encontrar alineados ambos aminoácidos es el doble de probable que la mutación.

Podríamos puntuar un alineamiento de dos secuencias multiplicando los *odd-ratios* calculados para cada posición. Sin embargo, en informática las multiplicaciones son costosas y se prefieren las sumas, así que se calcula el *log-odd* multiplicado por 10 de  $R_{ij}$ , estos números son más intuitivos y sencillos de sumar y serán la base de las *puntuaciones* de los Alineamientos:

$$S_{ij} = 10 \log_{10} \frac{M_{ij}}{f_j} = 10 \log_{10} R_{ij} \quad (8.3)$$

Las matrices de *log-odds* calculados con la Ecuación 8.3 son las que habitualmente denominamos PAM y usamos para calcular valores de similitud en alineamiento de secuencias (puntuaciones). En la Tabla 8.2 se puede consultar la matriz PAM250, una de las más usadas para puntuar alineamientos.

Si queremos encontrar un significado probabilístico de los valores *log-odd* de una matriz, bastaría con volver a calcular el *odd-ratio* ( $R_{ij}$ ):

$$R_{ij} = 10^{\frac{S_{ij}}{10}} \quad (8.4)$$

**Tabla 8.2:** Matriz de log-odds PAM250.

Otras nuevas versiones de las matrices PAM han sido calculadas con un número mayor de grupos de secuencias homólogas alineadas, sin embargo no han conseguido mejorar sustancialmente las matrices originales de Dayhoff [5, 7].

Otro tipo de matrices de sustitución que sí han conseguido mejorar a las PAM son las *matrices BLOSUM* (BLOcks of Amino Acid SUbstitution Matrix), creadas por Henikoff [6]. Las matrices BLOSUM fueron creadas a partir de datos de más de 500 grupos de alineamientos de secuencias de proteínas y con el objetivo de mejorar los alineamientos de secuencias divergentes donde las matrices PAM fallaban. Para definir diferentes matrices BLOSUM se marcaron diferentes umbrales de identidad de secuencias, de forma que las secuencias con mayor o igual identidad que el umbral se agruparon para disminuir su contribución en la matriz. Por ejemplo, para calcular la matriz BLOSUM62 se agruparon las proteínas con identidad mayor o igual que 62 %. Con los bloques de secuencias alineadas se calcula una tabla de frecuencias de cada pareja de aminoácidos alineados, obteniendo 210 parejas posibles con sus respectivas frecuencias de aparición que permitirán calcular los *odd-ratios* ( $R_{ij}$ ) entre las frecuencias observadas ( $q_{ij}$ ) y las frecuencias esperadas por casualidad ( $e_{ij}$ ) (Ecuación 8.5). Henikoff decidió calcular los *log-odds* ( $R_{ij}$ ) de una manera ligeramente diferente a Dayhoff, usando logaritmos en base 2 (Ecuación 8.6). En la Tabla 8.3 se representa la matriz BLOSUM62, ésta es la matriz preferida para usar por defecto por algoritmos tan famosos como BLASTP.

$$R_{ij} = \frac{q_{ij}}{e_{ij}} \quad (8.5)$$

$$S_{ij} = \log_2 \frac{q_{ij}}{e_{ij}} \quad (8.6)$$

**Tabla 8.3:** Matriz de log-odds BLOSUM62.

Las matrices BLOSUM demostraron ser más sensibles a la hora de identificar alineamientos de proteínas homólogas [6]. Las principales diferencias entre ambos tipos de matrices es que las PAM son generadas por extrapolación de datos de alineamientos de secuencias muy conservadas y las BLOSUM, por contra, son derivadas de datos reales de alineamientos de secuencias menos conservadas. En la Tabla 8.4 se muestra la equivalencia entre diferentes matrices PAM y BLOSUM, a menor distancia evolutiva PAM, mayor porcentaje de identidad BLOSUM y al contrario. Como norma general se prefiere el uso de matrices BLOSUM, sin embargo, cuando se realizan comparaciones de secuencias muy conservadas, las matrices PAM pueden conseguir mejores resultados.

PAM120 ↔ BLOSUM80  
PAM160 ↔ BLOSUM62  
PAM250 ↔ BLOSUM45

**Tabla 8.4:** Equivalencia de matrices PAM y BLOSUM.

Todo lo explicado hasta ahora sobre matrices de sustitución ha sido en el contexto de alineamientos proteicos. ¿Qué sucede en el caso de alineamientos de secuencias de DNA o RNA? Para los nucleótidos también se han calculado matrices PAM de forma similar a la explicada para proteínas [22], teniendo en cuenta las diferentes probabilidades de mutaciones por transición ( $A \leftrightarrow G$ ,  $C \leftrightarrow T/U$ ) o transversión ( $A/G \leftrightarrow C/T/U$ ). Sin embargo, programas como BLAST emplean por defecto puntuaciones de 1 y -2 para evaluar coincidencia/no coincidencia de nucleótidos respectivamente. Aunque el uso de matrices

PAM puede mejorar alineamientos de nucleótidos con identidades < 70 %, normalmente su mayor sensibilidad no compensa el mayor tiempo necesario para realizar los alineamientos, especialmente cuando estamos trabajando con genomas. Como ya se explicó, cuando se requiere alinear secuencias de DNA o RNA divergentes se prefiere traducirlas a secuencias proteicas antes de realizar su alineamiento.

### 8.3.3. Significación estadística y *E-value*

Tenemos dos secuencias alineadas y hemos puntuado su alineamiento mediante las matrices de sustitución, ahora bien *¿cómo sabemos si ambas secuencias son homólogas o su alineamiento es fruto del azar?* si su similitud o identidad son altos podremos suponer que son homólogas (ver Subsección 8.3.1), aunque nunca lo sabremos con el 100 % de certeza. Ahora surge otra pregunta, *¿qué se considera una alta identidad o similitud?* Valores de identidad o similitud para encontrar homólogos en una familia proteica concreta pueden no ser suficientemente altos para otra familia diferente. Según vamos bajando el umbral de similitud/identidad para la búsqueda de homólogos, llega un momento en que no podemos diferenciar entre los que realmente lo son y los que no. El porcentaje de identidad del 25-30 % en proteínas marca el límite donde coexisten alineamientos de verdaderos y falsos homólogos. Estos casos de alineamientos dudosos se denominan *twilight zone* y serán analizados en la siguiente Subsección 8.3.4.

Para intentar evitar la inexactitud de las medidas de similitud e identidad, se usa la *significación estadística*. La significación estadística mide la probabilidad de que un alineamiento no sea debido al azar. Si el alineamiento es improbable que sea fruto del azar, entonces ambas secuencias son probablemente homólogas. Como en el caso de la identidad/similitud, la significación estadística tampoco garantiza con un 100 % de certeza la homología, pero permite discriminar mucho mejor alineamientos dudosos.

La primera medida de significación estadística fue introducida por Lipman y Pearson [9, 15]. Para ello generaron secuencias al azar tomando bloques de aminoácidos de la secuencia alineada y calcularon sus valores de similitud. Tras ello calcularon la desviación de la medida de similitud del alineamiento original respecto a los alineamientos de secuencias aleatorias. Si esta desviación era superior a 6 veces la desviación estándar ambas proteínas eran probablemente homólogas. Esta medida se llama *Z-score* (*Z*), se calcula restando el valor de la similitud calculada mediante una matriz de sustitución (*S*) al valor medio de los alineamientos de secuencias al azar ( $\mu$ ), todo ello dividido entre la desviación estándar ( $\sigma$ ):

$$Z(S) = \frac{S - \mu}{\sigma} \quad (8.7)$$

Sin embargo, el *Z-score* es únicamente válido para distribuciones normales (gausianas) y los valores de alineamientos de secuencias al azar siguen un patrón de distribución de valores extremos.

Otra medida estadística muy usada en bioinformática son los *P-values* (ver Capítulo 3 “Estadística y R”). El *P-value* (*P*) del alineamiento de dos secuencias es la probabilidad de que dos secuencias aleatorias (de la misma longitud y composición) tengan una similitud mayor o igual al alineamiento original. A pesar de la validez de este parámetro estadístico, cuando alineamos secuencias se prefiere usar el *E-value*. El *E-value* (*E*) nos da el número esperado de secuencias aleatorias cuyo alineamiento da valores de similitud mayores o iguales que el valor del alineamiento original. La ventaja de usar *E-values* es su mayor manejabilidad, pues su rango de valores ( $-\infty - +\infty$ ) es más amplio e intuitivo que los *P-values* (0-1). No obstante, ambos valores pueden interconvertirse con la siguiente fórmula:

$$P(S) = 1 - e^{-E(S)} \quad (8.8)$$

Donde  $E(S)$  es el *E-value* para obtener un valor de similitud mayor o igual que  $S$ . La fórmula para calcular analíticamente *E-values* fue publicada por Karlin y Altschul [8]:

$$E(S) = Kmne^{-\lambda S} \quad (8.9)$$

$E$  es proporcional al espacio de búsqueda ( $mn$ ),  $m$  es la longitud de la secuencia problema,  $n$  es la longitud de todo el conjunto de secuencias a alinear con la secuencia problema,  $\lambda$  es un parámetro para normalizar los valores de similitud y hacerlos independientes de la matriz de sustitución empleada, y  $K$  es un factor de escalado para el espacio de búsqueda. En la ecuación podemos observar como  $E$  disminuye exponencialmente según aumenta el valor de similitud exigido ( $S$ ). A su vez la relación lineal de  $E$  con el espacio de búsqueda sugiere la importancia estadística del espacio muestral respecto a considerar únicamente valores de similitud como estabamos haciendo hasta el momento. Un valor de  $E = 0,1$  cuando estamos alineando una secuencia contra una base de datos de 1000 secuencias puede ser equivalente a  $E = 0,001$  si únicamente alineamos contra 10 secuencias.

Hasta ahora habíamos usado los valores de identidad y similitud para evaluar los alineamientos de secuencias, a partir de este momento pasaremos a usar los valores de *E-value*. Tampoco habíamos tenido en cuenta que los valores de similitud dependían de la matriz de sustitución empleada al calcularlos, lo cual supone algo similar a definir una distancia sin nombrar las unidades. Podemos calcular valores de similitud normalizados en unidades de similitud arbitrarias llamados *bit-scores* ( $S'$ ) con la siguiente fórmula:

$$S' = \frac{\lambda S - \ln K}{\ln 2} \quad (8.10)$$

Ahora el cálculo de *E-values* es mucho más sencillo, puesto que los parámetros  $K$  y  $\lambda$  están implícitos en  $S'$ :

$$E(S') = mn2^{-S'} \quad (8.11)$$

Finalmente podemos concluir este apartado con la idea de que los *E-values* son los mejores indicadores que tenemos para conocer la validez de un alineamiento en el reconocimiento de secuencias homólogas. Valores de  $E(S) \leq 0,001$  son habitualmente suficientes para considerar que el alineamiento de dos secuencias no es fruto del azar sino de la homología, incluso valores superiores de  $E(S)$  pueden darse en homólogos remotos.

### 8.3.4. El *twilight* u ‘ocaso’ de los alineamientos

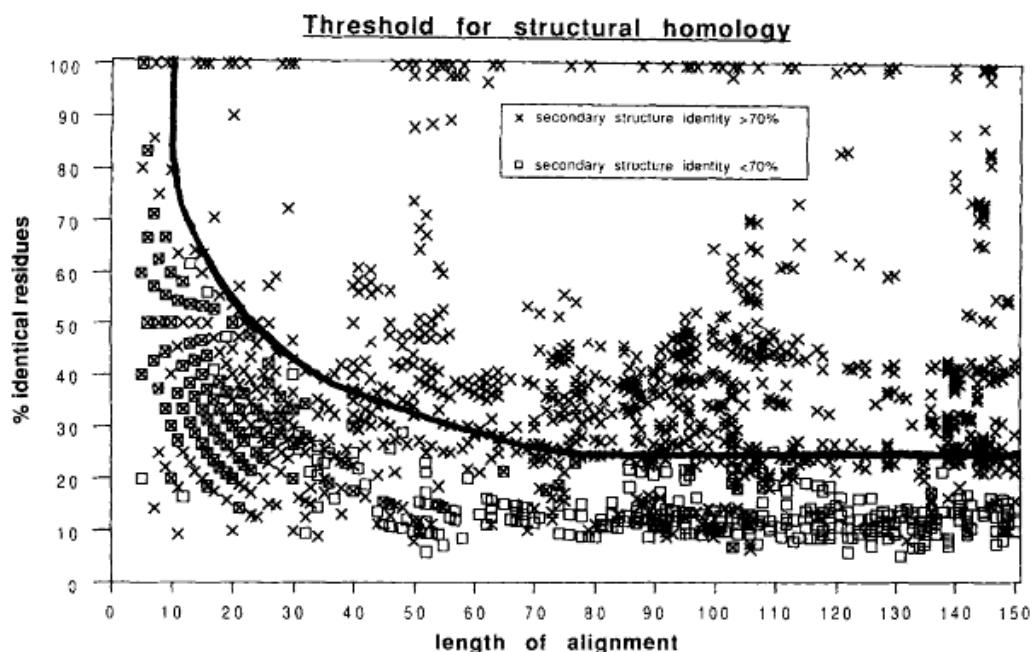
La *zona de twilight (ocaso) de los alineamientos* de proteínas es el rango de pares de valores de longitudes de secuencia y valores de identidad para los cuales existe una alta probabilidad de que sean erróneos [18, 19]. Si observamos la Figura 8.5 podemos concluir que alineamientos con menos del 25 % de residuos idénticos pertenecen la mayoría a proteínas no homólogas. Además, si la longitud de la secuencia proteica alineada es menor de 80 aminoácidos, se requiere de mayor % de identidad para encontrar homología. Llegando a requerirse una identidad del 100 % para secuencias más cortas que 10 residuos.

En el caso de secuencias de nucleótidos la zona de twilight es más difusa, la identidad entre las secuencias de cDNA o mRNA de dos genes homólogos cuyas proteínas comparten un 25 % de identidad puede ser mucho menor.

A la hora de evaluar alineamientos se utilizará siempre que sea posible el *E-value* como ya se ha explicado. El *E-value* nos dará la probabilidad de que existan falsos alineamientos y por ello estemos dentro de la zona de *twilitght* o no. Sin embargo, en casos dudosos podremos considerar la identidad entre secuencias y valorar su pertenencia o no al grupo de alineamientos dudosos.

Un buen consejo para alineamientos dentro de la zona de *twilitght*, es alinear ambas secuencias contra bases de datos anotadas. Si encontramos para alguna de las secuencias resultados de genes o proteínas de la misma familia con buenos alineamientos, podremos esclarecer la homología de las secuencias iniciales.

Programas de alineamiento como BLAST recomiendan secuencias de longitud mayor a 22 nucleótidos o 6 aminoácidos, esto se debe en parte a la necesidad de una mínima longitud de secuencia para salir de la zona de *twilitght* y poder realizar alineamientos con un mínimo de probabilidad de certeza.



**Figura 8.5:** Identidad de secuencia vs. longitud de alineamiento. Gráfica original de Sander y Schneider [19] donde las proteínas homólogas están representadas por X y el resto por cuadrados. La curva marca la separación entre alineamientos de proteínas homólogas y otras no homólogas o dudosas.

### 8.3.5. Técnicas y programas de alineamiento

Ahora que ya hemos visto las bases teóricas del alineamiento de secuencias vamos a ver cómo se pueden emplear diferentes herramientas para ello. Pero primero deberemos diferenciar entre dos tipos posibles de Alineamientos:

**Global:** alineamiento de la secuencia completa. Es útil cuando se comparan secuencias muy similares en tamaño y composición, por ejemplo de dos genes muy conservados.

**Local:** cuando sólo nos interesa alinear regiones similares entre secuencias. Se utiliza cuando las secuencias a comparar son diferentes en tamaño o poseen regiones no conservadas. Un ejemplo

podría ser el alineamiento de un dominios proteico entre dos proteínas con diferente número total de dominios.

Para realizar de una forma eficiente y rápida estos alineamientos se utilizan algoritmos computacionales de *programación dinámica*. No vamos a entrar en detalle, pero sí que merece la pena conocer los dos más importantes: *Needleman–Wunsch* [12] es el empleado para realizar alineamientos globales y *Smith–Waterman* [21] sirve para optimizar alineamientos locales entre secuencias. Jonathan Pevsner realiza una muy buena y detallada descripción de ambos métodos en su libro *Bioinformatics and Functional Genomics* [17].

A su vez hay que distinguir entre el alineamiento de pares de secuencias y alineamientos múltiples:

**Pares de secuencias:** mide la similitud entre dos secuencias, por ejemplo la secuencia problema y cada una de las secuencias de una base de datos, realizando comparaciones individuales entre pares.

**Alineamiento múltiple:** compara más de dos secuencias al mismo tiempo. Es especialmente importante cuando queremos interrelacionar varias secuencias entre sí, por ejemplo para calcular árboles filogenéticos, buscar patrones o regiones conservadas. Es un problema bastante más complejo que el de alinear sólo 2 secuencias y existen diferentes técnicas que ofrecen diferentes resultados.

En ambos casos el alineamiento puede ser local o global

### 8.3.6. Alineamiento local de pares de secuencias

A continuación se explicará el uso de dos de los programas más populares para el alineamiento local de pares de secuencias: BLAST y FASTA. Ambos usan el algoritmo de Smith–Waterman junto a técnicas heurísticas que los hacen extremadamente rápidos y útiles para búsquedas en grandes bases de datos.

#### BLAST

*BLAST*<sup>6</sup> (Basic Local Alignment Search Tool) es la herramienta más popular de búsqueda y alineamiento de secuencias. De hecho, el artículo original que la describe [1] es uno de los más citados en la historia de la ciencia. Como su propio nombre indica, realiza alineamientos locales tanto de nucleótidos como de amino ácidos, normalmente las secuencias problema se alinean contra secuencias de bases de datos. El algoritmo de BLAST tiene el siguiente funcionamiento:

1. *Algoritmo heurístico:* divide la secuencia a alinear en subsecuencias (*k-meros*) de longitud más corta (3 amino ácidos o 28 nucleótidos por defecto) y busca éstas entre las secuencias de la base de datos.
2. *Programación dinámica:* cuando encuentra varias subsecuencias en una misma entrada de la base de datos, extiende el alineamiento hacia ambos lados mediante el algoritmo de programación dinámica de Smith–Waterman [21] y una matriz de sustitución (por defecto Blosum62 para amino ácidos).
3. *Significación estadística:* finalmente calcula el *bit-score* y *E-value* del alineamiento local extendido que nos dará la probabilidad de que dicho alineamiento sea fruto del azar en comparación con el tamaño de la base de datos (ver Subsección 8.3.3).

Existe una familia de programas que usan el algoritmo BLAST de diferentes formas:

---

<sup>6</sup>BLAST. <http://blast.ncbi.nlm.nih.gov>

**blastn:** busca una secuencia de nucleótidos en una base de datos del mismo tipo.

**blastp:** busca una secuencia proteica en una base de datos de proteínas.

**blastx:** traduce a amino ácidos una secuencia de nucleótidos y la busca en una base de datos de proteínas.

**tblastn:** busca una secuencia proteica en una base de datos de nucleótidos previamente traducidos a proteínas.

**tblastx:** traduce a amino ácidos una secuencia de nucleótidos y la busca en una base de datos de nucleótidos previamente traducidos a proteínas.

**bl2seq:** compara dos secuencias entre sí, sin usar base de datos.

**blastpgp:** realiza la búsqueda de una proteína en una base de datos varias veces, de forma que en cada nueva búsqueda se utiliza una nueva secuencia o perfil que es fruto de la combinación de los resultados de la búsqueda anterior. De esta forma es posible encontrar secuencias homólogas evolutivamente más remotas que con una búsqueda clásica, pero también existe el peligro de encontrar secuencias sin ningún tipo de relación.

**megablast:** es una versión más rápida de *blastn* utilizada para buscar un gran número de secuencias de DNA en bases de datos. Para acelerar la búsqueda, concatena varias secuencias en una única y tras la búsqueda separa los resultados.

Finalmente apuntar que todos los programas enumerados se pueden usar en su versión *online*<sup>6</sup> o descargarlos para usarlos en nuestro ordenador<sup>7</sup>.

## FASTA

El paquete de programas FASTA<sup>8</sup> es más popular por el formato de archivo que lleva su nombre (ver ??) que por sus herramientas. Sin embargo FASTA es un conjunto de programas de alineamiento muy similar a BLAST [9]. El esquema del algoritmo de alineamiento de FASTA es casi idéntico al de BLAST por lo que no se volverá a explicar. Únicamente destacar que BLAST es más rápido que FASTA, aunque FASTA puede ser más sensible para alinear secuencias muy divergentes.

### 8.3.7. Alineamiento global de pares de secuencias

El alineamiento global de secuencias tiene más limitaciones de uso que el local. No sirve para detectar similitud entre proteínas de diferente longitud o con múltiples dominios funcionales que no son de una misma familia o que poseen largas duplicaciones o delecciones en la secuencia. Su utilidad es más bien limitada al alineamiento de proteínas homólogas para generar árboles filogenéticos, ya que los valores de similitud de estos alineamientos pueden ser transformados fácilmente en distancias evolutivas [16].

## Needle y Stretcher

*Needle* es un programa que implementa rigurosamente el algoritmo de Needleman-Wunsch [12]. Forma parte de *EMBOSS*<sup>9</sup> (The European Molecular Biology Open Software Suite). Cuando las secuencias

<sup>7</sup>Download NCBI Software. <http://www.ncbi.nlm.nih.gov/guide/howto/dwn-software/>

<sup>8</sup>FASTA Sequence Comparison at the University of Virginia. <http://fasta.bioch.virginia.edu/>

<sup>9</sup>The European Molecular Biology Open Software Suite. <http://emboss.sourceforge.net>

a alinear son largas, el alineamiento puede fallar o ser muy lento debido a las altas necesidades de memoria del algoritmo (proporcional al producto de las longitudes de ambas secuencias).

*Stretcher* es una modificación del algoritmo de Needleman-Wunsch [11] que requiere únicamente una cantidad de memoria proporcional a la secuencia más corta, con lo cual es válido para todo tipo de secuencias. También forma parte del paquete de programas EMBOSS<sup>9</sup>.

### 8.3.8. Alineamiento múltiple de secuencias

El *alineamiento múltiple* de secuencias sirve, como su nombre indica, para alinear más de dos secuencias al mismo tiempo. Normalmente las herramientas de alineamiento múltiple incluyen algoritmos para realizar tanto alineamientos de tipo local como global. La principal utilidad del alineamiento múltiple es la detección de homología entre grupos de secuencias que presentan baja similitud entre sí, pero que al compararlas en su conjunto se detectan posiciones o regiones muy conservadas que indican su origen evolutivo común. Por ello este tipo de alineamiento permite fácilmente la detección de regiones o dominios conservados entre varias secuencias proteicas, como pueden ser sitios catalíticos, de trasducción de señal o dominios de unión a DNA o entre proteínas. Los resultados de este tipo de alineamientos también son muy valiosos para el análisis filogenético y la construcción de árboles (ver Capítulo 9 “Filogenia y evolución molecular”).

El cálculo del mejor alineamiento múltiple alcanza gran complejidad según se aumenta el número de secuencias, lo que también incrementa las posibles combinaciones entre sus alineamientos y diferentes posibilidades de incluir *gaps* entre ellas. Para resolver el problema se emplean diferentes estrategias heurísticas para obtener buenos alineamientos en un tiempo razonable, aunque no sean los óptimos. La *técnica progresiva* (también conocida como método jerárquico o de árbol) es la más popular. Consiste en realizar previamente todas las posibles combinaciones de alineamientos entre pares de secuencias para construir un árbol de distancias por similitud. El alineamiento comienza tomando como referencia el alineamiento de las dos secuencias más similares y va añadiendo una por una y en el orden establecido por el árbol el resto de secuencias a alinear. El principal problema del método progresivo es su fuerte dependencia del alineamiento de las dos secuencias inicialmente alineadas que servirán de referencia al resto, si ambas secuencias son muy diferentes darán un mal alineamiento que irá empeorando según se añadan más secuencias.

#### ClustalW/ClustalO

*Clustal Omega*<sup>10</sup> es el programa de alineamiento múltiple más popular (en su versión antigua se conoce como ClustalW). Utiliza una técnica progresiva mejorada que realinea de forma iterativa las secuencias iniciales y que utiliza modelos ocultos de Markov para mejorar la eficiencia de los alineamientos. Permite obtener alineamientos múltiples de buena calidad incluso con cientos de miles de secuencias en un tiempo razonable [20].

#### Tcoffee

*T-Coffee*<sup>11</sup> es otro programa de alineamiento múltiple de método progresivo. Su principal característica es que permite integrar en el alineamiento información estructural, de estructura secundaria o combinar diferentes métodos de alineamiento múltiple en un único resultado.

---

<sup>10</sup>Clustal: Multiple Sequence Alignment. <http://www.clustal.org>

<sup>11</sup>T-Coffee. <http://www.tcoffee.org>

### 8.3.9. Edición y visualización de alineamientos

Para terminar el capítulo hablaremos de cómo visualizar y modificar alineamientos de una forma gráfica a partir de los archivos generados por los programas de alineamiento explicados en los apartados anteriores.

Un programa de edición y visualización de alineamientos muy completo es *Jalview*<sup>12</sup>. Es un programa gratuito que además permite hacer nuevos alineamientos, representar árboles filogenéticos o visualizar estructuras moleculares. Jalview se puede usar tanto en su versión online como descargarlo y usarlo en nuestro ordenador.

---

<sup>12</sup>Jalview. <http://www.jalview.org>



## 8.4. Bibliografía

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, 1990.
- [2] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. Genbank. *Nucleic Acids Res*, 39(Database issue):D32–7, 2011.
- [3] U. Consortium. Reorganizing the protein space at the universal protein resource (uniprot). *Nucleic Acids Res*, 40(Database issue):D71–5, 2012.
- [4] M. O. Dayhoff and R. M. Schwartz. *A model of evolutionary change in proteins*, volume 5, chapter 22, pages 345–352. National Biomedical Research Foundation, 1978.
- [5] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256(5062):1443–5, 1992.
- [6] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–9, 1992.
- [7] D. T. Jones, W. R. Taylor, and J. M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci*, 8(3):275–82, 1992.
- [8] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci U S A*, 87(6):2264–8, 1990.
- [9] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–41, 1985.
- [10] A. G. Murzin. Can homologous proteins evolve different enzymatic activities? *Trends Biochem Sci*, 18(11):403–5, 1993.
- [11] E. W. Myers and W. Miller. Optimal alignments in linear space. *Comput Appl Biosci*, 4(1):11–7, 1988.
- [12] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–53, 1970.
- [13] W. R. Pearson. Effective protein sequence comparison. *Methods Enzymol*, 266:227–58, 1996.
- [14] W. R. Pearson. *Protein sequence comparison and Protein evolution*. PhD thesis, University of Virginia, 2001.
- [15] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85(8):2444–8, 1988.
- [16] W. R. Pearson and T. C. Wood. *Statistical significance in biological sequence comparison*. PhD thesis, University of Virginia, 2000.
- [17] J. Pevsner. *Bioinformatics and Functional Genomics*. Wiley, 2009.
- [18] B. Rost. Twilight zone of protein sequence alignments. *Protein Eng*, 12(2):85–94, 1999.
- [19] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 9(1):56–68, 1991.
- [20] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol Syst Biol*, 7:539, 2011.
- [21] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–7, 1981.
- [22] D. States, W. Gish, and S. Altschul. Improved sensitivity of nucleic acid database searches using application-specific scoring matrices. *Methods*, 3:66–70., 1991.



# Capítulo 9

## Filogenia y evolución molecular

*Federico Abascal, Iker Irisarri y Rafael Zardoya*

### 9.1. Introducción

Hace aproximadamente 3.500 millones de años surgió la vida en la Tierra. Desde entonces, la vida ha explorado múltiples caminos, unos breves y otros muy extensos. Sólo algunos de ellos han perdurado hasta la actualidad. Es una historia de cambios y adaptación. Una adaptación que se refleja a muchos niveles, desde la maquinaria celular y los genes que gobiernan el desarrollo, hasta modificaciones morfológicas o de comportamiento. Esta historia ha quedado registrada en el genoma de las especies actuales, de modo que el estudio de la información genética nos permite conocer los vericuetos de la historia de la vida, y además nos abre una ventana al laboratorio evolutivo, donde infinidad de opciones han sido probadas, pero sólo una parte de ellas han tenido éxito. Una pregunta clave en Biología es *por qué los seres vivos son como son*, y no se puede responder sin considerar el componente histórico. Los estudios evolutivos intentan resolver ésta y otras cuestiones, y aportan información muy valiosa prácticamente en cualquier nivel de organización en biología. En palabras del célebre biólogo Theodosius Dobzhansky: “*Nada tiene sentido en biología sino es a la luz de la evolución*”.

Cuando hablamos de evolución nos referimos al proceso de descendencia con modificación a partir de un ancestro común [8]. El hecho de que los organismos comparten relaciones ancestro-descendiente nos permite utilizar árboles filogenéticos para representar gráficamente sus relaciones de parentesco. Los árboles filogenéticos tienen dos componentes principales: las relaciones de parentesco (que conocemos como la topología del árbol) y la cantidad de cambio acumulado en cada rama (que conocemos como longitudes de rama).

Los métodos de reconstrucción filogenética permiten hacer uso de la información contenida en el genoma de las especies actuales para *inferir* su historia evolutiva (el árbol filogenético). Hablamos de inferencia porque la reconstrucción filogenética se basa necesariamente en una información incompleta, puesto que sólo tenemos acceso a la información genética de organismos actuales [43]. Estamos, por tanto, ante un problema de encontrar la (o las) hipótesis más probable(s). Por eso, la estadística juega un papel muy importante en el análisis evolutivo.

En origen, los métodos de reconstrucción filogenética fueron concebidos para inferir las relaciones de parentesco entre organismos. Sin embargo, y dado que la evolución subyace a todo proceso biológico, en la actualidad los métodos filogenéticos se utilizan para responder a muchos otros interrogantes. Estos incluyen la reconstrucción de la historia evolutiva de genes, la estimación de tiempos de divergencia, la

reconstrucción de estados ancestrales, el estudio de la selección natural, la detección de recombinación, la dinámica poblacional, teniendo importantes aplicaciones prácticas en epidemiología, criminología y medicina [18].

En este capítulo introduciremos algunos principios importantes que rigen la evolución a nivel molecular y explicaremos cómo interpretar correctamente un árbol filogenético. A continuación, presentaremos los métodos de inferencia filogenética más frecuentemente utilizados, haciendo énfasis en los métodos probabilísticos por ser los que mejor extraen la información filogenética de las secuencias. Además, veremos cómo estimar la robustez estadística del árbol inferido y cómo el contraste de hipótesis puede utilizarse tanto para comparar árboles filogenéticos como para comprender cómo ha sido el proceso evolutivo subyacente.

### 9.1.1. Teoría de la Evolución

En 1859, veinte años después de regresar de su viaje a bordo del *Beagle*, Charles Darwin publicó *El origen de las especies*, donde formulaba detalladamente la teoría de la evolución. Este mérito debe ser compartido con Alfred Russell Wallace, que de forma independiente propuso la selección natural como el mecanismo generador de la diversidad biológica.

Darwin propuso que todos los organismos comparten un ancestro común, a partir del cual ha surgido toda la diversidad que observamos, y que los organismos similares se encuentran más emparentados entre sí, es decir, comparten un ancestro común más cercano. La clave del proceso evolutivo se encuentra en la selección natural. La teoría de la evolución propone que ciertos cambios espontáneos generan una variabilidad entre los organismos que puede heredarse. Las cualidades que resultan más favorables de cara a la adaptación al entorno (el factor determinante es el éxito reproductivo) serán las favorecidas por el mecanismo de selección natural. Hoy en día puede parecernos una idea sencilla, casi intuitiva, pero considerando que en aquel tiempo no se conocía el DNA ni las proteínas, es comprensible que se admire a Darwin y también que su teoría despertase tanta polémica.

El descubrimiento de los genes lo debemos a Gregor Mendel, cuyo trabajo hacia 1866 permitió conocer con mayor detalle cómo se produce la transmisión de información entre generaciones y cómo se genera la variabilidad sobre la que actuará la selección natural. Más tarde se determinó la naturaleza de estos genes como ácidos nucleicos (DNA) y de sus productos (proteínas), se descubrió la estructura de doble hélice y se descifró el código genético. Posteriormente fue posible determinar las secuencias de proteínas primero y DNA después, dando lugar a los primeros estudios de evolución molecular.

### 9.1.2. Evolución molecular

Gracias a los hallazgos anteriormente citados, hoy en día conocemos la base molecular de la evolución propuesta por Darwin. Las mutaciones del DNA son la fuerza productora de variabilidad sobre la que actúa la selección natural. La mutación se considera un proceso aleatorio, aunque está influenciado por una amplia variedad de factores, como lo son el modo de vida de los organismos, el tiempo de generación, o la tasa metabólica [2]. Además, genes distintos tienen tasas evolutivas distintas que dependen de la estructura y función de las proteínas que codifican. Distintas regiones del genoma presentan patrones y tasas de mutación diferentes, relacionados con el estrés replicativo (número veces que se copia por generación), la recombinación, la metilación del DNA o el posicionamiento de los nucleosomas. Debido a la naturaleza poblacional del proceso evolutivo, algunas de estas mutaciones se perderán con el tiempo, pero otras se fijarán en la población (lo que conocemos como sustituciones). En lo que respecta a la

inferencia filogenética, las sustituciones son una fuente de información muy valiosa que nos permite establecer relaciones de parentesco y estudiar la naturaleza del proceso evolutivo.

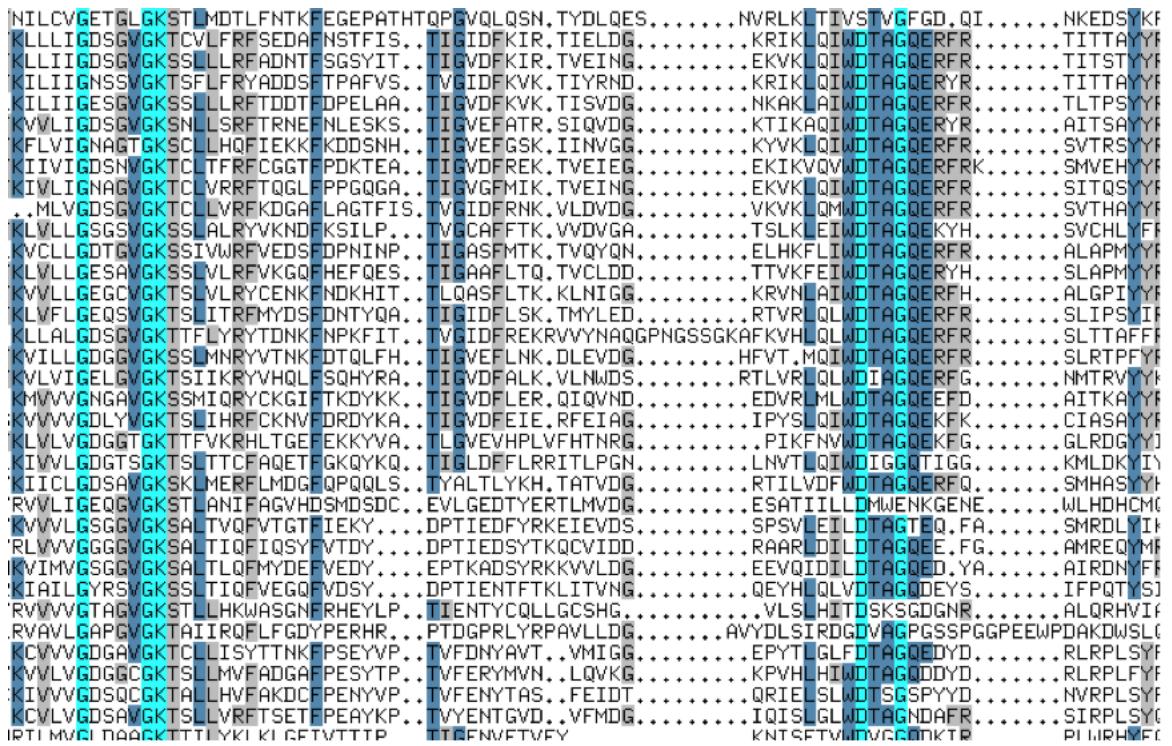
A lo largo de este capítulo nos centraremos en el análisis filogenético de secuencias de DNA o proteínas, puesto que presentan numerosas ventajas con respecto a otro tipo de datos y son (con diferencia) los más utilizados en la actualidad. Los primeros análisis comparativos de secuencias de proteínas, hace ya más de 50 años, supusieron un cambio importante en el enfoque del estudio de la evolución. En un principio se extrapolaron los principios evolutivos que se conocían del estudio de la morfología, aplicándolos al nivel molecular. A nivel morfológico todo cambio es objeto de *selección natural*, sea ésta positiva o negativa. La *selección positiva* (o evolución adaptativa) se refiere al proceso por el que los cambios que ofrecen cierta ventaja son favorecidos y tienden a fijarse en la población. Un ejemplo morfológico clásico es el de la evolución del cuello de la jirafa: un cuello más largo permite a los individuos que lo poseen comer las hojas de los árboles que otros no alcanzaban, aumentando sus posibilidades de dejar descendencia. La *selección negativa* (o purificadora) se refiere a la eliminación de aquellos cambios que perjudican la adaptación, y tienden a desaparecer de la población al reducir el éxito reproductivo.

Cuando Zuckerkandl y Pauling [48] compararon las primeras secuencias de globinas de distintas especies, observaron algo que no encajaba con esta visión: el número de cambios aminoácídicos entre linajes era aproximadamente proporcional al tiempo de divergencia estimado a partir del registro fósil. Este fenómeno se dio a conocer como reloj molecular. El reloj molecular parecía contradecir la visión existente de la evolución, según la cual todo cambio era objeto de selección. Como contraposición esta la visión seleccionista, Kimura (1983) [23] formuló la teoría neutral de la evolución molecular, según la cual una parte considerable de los cambios no afectan el éxito reproductor del individuo (o su efecto es muy pequeño), y al no estar sujetos a selección, estos cambios se acumulan a un ritmo aproximadamente constante a lo largo del tiempo. En la actualidad se asume que la evolución molecular está gobernada tanto por procesos neutrales como por la selección natural, si bien no siempre es sencillo conocer el peso específico de estas fuerzas evolutivas.

Los alineamientos múltiples son una forma de establecer relaciones de homología (origen evolutivo común) entre las posiciones (nucleótido o aminoácidos) de un conjunto de secuencias. La huella de algunos procesos evolutivos puede observarse fácilmente en los alineamientos múltiples. Así, la *selección negativa* se manifiesta como conservación en el alineamiento (todas las secuencias tienen el mismo nucleótido o amino ácido). Cuando una posición está conservada no es que no haya sufrido mutaciones, sino que cuando ha mutado la selección negativa (purificadora) ha actuado. La huella de la selección negativa es un buen indicio de relevancia funcional. La *evolución neutral*, en cambio, se manifiesta como variabilidad. Las posiciones que varían más son aquellas que tienen menor relevancia en el mantenimiento de la estructura y la función de la proteína. En realidad, entre estos dos extremos, conservación y variabilidad, existe todo un gradiente que depende de la relevancia de cada posición para la estructura y función del gen en cuestión (Figura 9.1). La huella de los procesos evolutivos se aprecia de forma diferente a nivel molecular y morfológico. A nivel molecular, por ejemplo, y a diferencia de lo que ocurre con la selección negativa y la evolución neutral, la huella de la *selección positiva* es difícil de detectar (volveremos a este problema más adelante). Por el contrario, a nivel morfológico la evolución positiva y negativa resultan obvias, mientras que la evolución neutral no se aprecia fácilmente.

### 9.1.3. Interpretación de un árbol

Un *árbol* consta de ramas y nodos que conectan estas ramas entre sí. Las *ramas terminales* (hojas) corresponden a las secuencias actuales (observadas), y las *ramas internas* representan sus ancestros hipotéticos. Su longitud refleja la cantidad de cambio acumulado. Los *nodos* relacionan estas ramas entre sí según su relación ancestro-descendiente. La rama más interna, el ancestro común, representa la

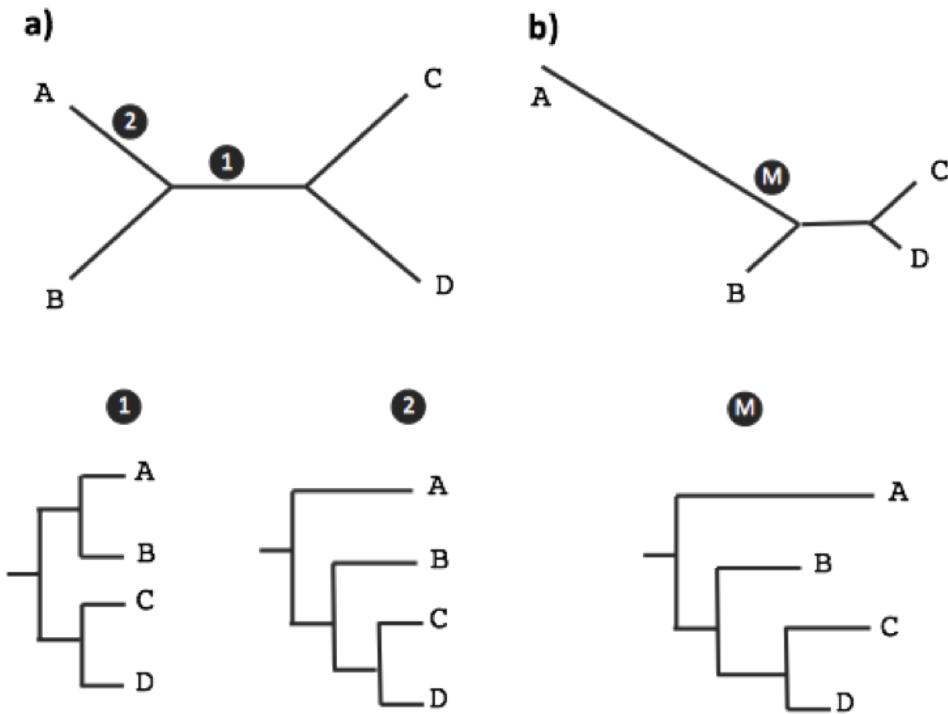


**Figura 9.1:** Alineamiento múltiple de proteínas. En él podemos observar regiones bajo distintas intensidades de selección que se manifiestan en posiciones más o menos conservadas en las distintas secuencias.

raíz del árbol y es esencial para dotar al árbol de un sentido histórico. Los métodos de reconstrucción filogenética a partir de secuencias (salvo alguna excepción) producen árboles sin raíz. En estos árboles, la raíz puede establecerse a posteriori en distintos puntos, lo cual determinará las relaciones de parentesco entre las secuencias. Por ejemplo, en la Figura 9.2, al enraizar el árbol en 1, la secuencia A será el grupo hermano de B, pero si lo enraizamos en 2, A será el grupo hermano de B, C, más D. Elegir un enraizamiento correcto es esencial para poder interpretar la historia evolutiva.

La forma más recomendable de enraizar un árbol es incluyendo una secuencia externa al grupo de interés que dote al árbol filogenético de sentido histórico (polaridad). La elección del *grupo externo* (o *outgroup*) requiere cierta información adicional, como la derivada de estudios morfológicos o taxonómicos. En algunos casos no disponemos de esta información o del grupo externo, en cuyo caso el árbol suele enraizarse en el *punto medio* (*midpoint rooting*), es decir, en la rama más larga (M en la Figura 9.2(b)). Cuando las secuencias evolucionan a velocidades muy distintas (ausencia de reloj molecular, ver más adelante) el enraizamiento en el punto medio debe interpretarse con precaución.

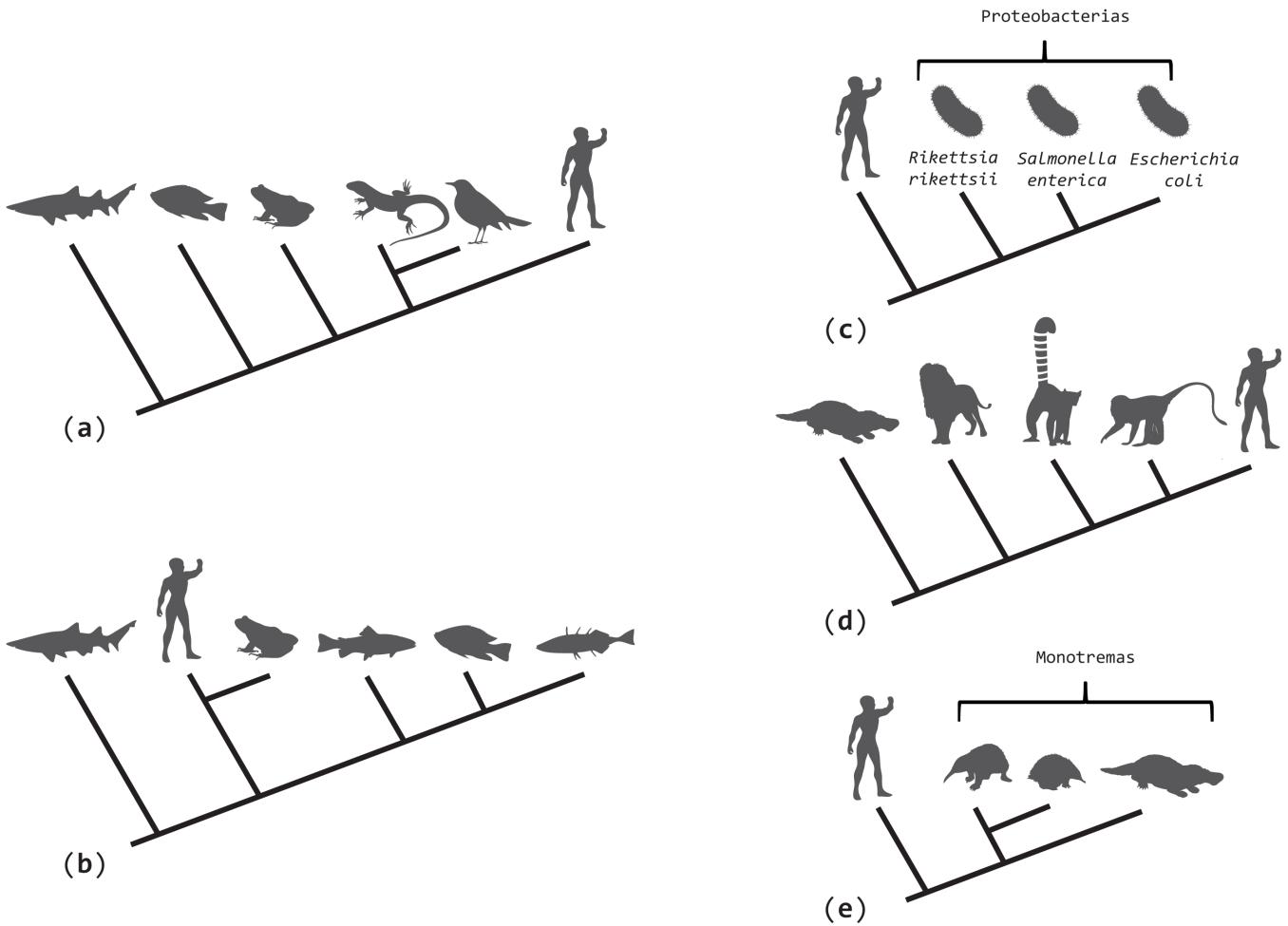
A pesar de que los árboles filogenéticos son representaciones bastante intuitivas de la historia evolutiva, ciertos errores de interpretación son relativamente frecuentes, incluso entre científicos experimentados [30]. El error más común es interpretar el árbol como una línea de progreso. Así, si observamos la Figura 9.3(a), podríamos caer en la tentación (antropocéntrica) de ver una direccionalidad en la evolución desde peces hasta humanos, pasando por anfibios, reptiles y aves. Sin embargo, una representación alternativa pero coherente con la anterior podría sugerirnos lo contrario (Figura 9.3(b)). Si 9.3(a) y 9.3(b) te sugieren relaciones diferentes es que estás interpretando erróneamente los árboles; posiblemente estás asumiendo líneas de progreso entre distintas especies actuales. Es un error pensar que entre las especies actuales (observables) hay unas más evolucionadas que otras; todas lo están por igual, simplemente representan distintas ramas del árbol de la vida. La línea de progreso existe, pero ésta



**Figura 9.2:** La raíz da sentido histórico (polariza) a un árbol filogenético. En (a) se muestran dos posibles enraizamientos (1, 2) para este árbol de cuatro secuencias (A, B, C, D) y cómo éstos se traducen en historias evolutivas diferentes. En (b) se muestra el enraizamiento en el punto medio (*midpoint rooting*).

en el tiempo: va desde la raíz del árbol hacia las hojas y no entre distintas hojas. Para evitar caer en este error, recuerda que las ramas de los árboles pueden rotarse de cualquier manera sin que ello altere las relaciones representadas en el árbol. Otro ejemplo clarificador lo encontramos en la Figura 9.3(c), donde el humano es el grupo externo utilizado para enraizar un árbol de bacterias. Ese árbol, aunque pueda sorprenderte, es absolutamente correcto; si te sorprende es porque asumes (consciente o inconscientemente) una direccionalidad evolutiva en el sentido equivocado. Este árbol simplemente nos dice que las bacterias están más cercanamente emparentadas entre sí de lo que lo están con los humanos, pero en ningún caso que los humanos hayan dado origen a las bacterias.

Hablando de errores comunes, es frecuente oír hablar de especies basales, primitivas o poco evolucionadas. La Figura 9.3(d) muestra un árbol de mamíferos placentarios donde el ornitorrinco (monotremo) es el grupo externo. Si al ver este árbol piensas que el ornitorrinco es un mamífero basal o primitivo, tal vez ver el árbol 9.3(e) te ayude a reconocer tu error. Para hablar con propiedad, debemos referirnos a grupos de divergencia temprana o tardía (*early or late branching*) [7] y referirnos a las relaciones representadas en el árbol como relaciones de grupos hermanos (*sister group relationships*).



**Figura 9.3:** Árboles filogenéticos que tratan de ilustrar ciertos errores comunes en su interpretación (ver texto).

## 9.2. Métodos de reconstrucción filogenética

En esta sección describiremos algunos de los métodos más utilizados para reconstruir la historia evolutiva a partir de datos de secuencia de DNA o proteínas, aunque estos métodos también pueden utilizarse con otro tipos de datos, como por ejemplo caracteres morfológicos.

Existen dos tipos principales de métodos de reconstrucción filogenética: (i) los que utilizan distancias genéticas, siendo los más conocidos el método *neighbor-joining* (NJ) y el de *mínima evolución* (ME); y (ii) los basados en caracteres, como los métodos de *máxima parsimonia* (MP), de *máxima verosimilitud* (MV) (en inglés, *maximum likelihood*) y de *inferencia Bayesiana* (IB) [18, 43]. Dentro de los métodos basados en caracteres, MV e IB (a diferencia de MP) son métodos probabilísticos que tratan la inferencia filogenética como un problema estadístico, y utilizan modelos explícitos de evolución molecular para el cálculo de probabilidades. Además de los anteriores, existen otros métodos, pero debido a la gran variedad y complejidad de éstos quedan fuera del alcance de este capítulo. Aquí introduciremos únicamente los métodos más comúnmente utilizados: un método que utiliza distancias genéticas (NJ) y los métodos probabilísticos (MV e IB). Los métodos probabilísticos son los que mejor aprovechan la información filogenética contenida en las secuencias, los más avanzados y generalmente los más fiables.

Para saber más acerca de métodos filogenéticos: Swofford et al. 1996 [43], Felsenstein 2004 [14] y Lemey et al. 2009 [27].

### 9.2.1. El alineamiento múltiple, la información de partida

Los *alineamientos múltiples* representan hipótesis de homología de posición entre caracteres (nucleótidos, codones o aminoácidos) de las distintas secuencias. Son el punto de partida para inferir filogenias y estudiar patrones y tasas de cambio evolutivo. Existen muchos métodos para el alineamiento de secuencias y han sido tratados en detalle en el Capítulo 8 “Análisis de secuencias biológicas”. Debemos tener en cuenta que partir de un buen alineamiento es clave para obtener estimas fiables de la filogenia. Pero *¿cómo saber si nuestro alineamiento es correcto?* Una cuestión particularmente importante es que además de la información filogenética, los alineamientos también contienen cierto ruido, por lo que es recomendable analizarlos cuidadosamente.

En ocasiones, como cuando alineamos regiones que evolucionan rápidamente, establecer la homología de posición resulta complejo. En las regiones más variables es común que hayan ocurrido múltiples cambios superpuestos (mutaciones sobre mutaciones), inserciones y delecciones. Estos fenómenos producen hipótesis de homología entre caracteres erróneas y el alineamiento no es fiable. Esto es el *ruido* al que nos referíamos, que puede incluso llegar a superar a la *señal filogenética*. Para aliviar estos problemas existen diversos métodos para *limpiar* alineamientos, los cuales básicamente eliminan las regiones más variables. GBlocks [5] retiene los bloques del alineamiento que contengan pocos gaps (inserciones y delecciones) y presenten poca variabilidad, pudiendo ajustarse los parámetros según el caso. TrimAl [4] es un método muy versátil que acepta múltiples opciones, como la eliminación de columnas del alineamiento con un porcentaje determinado de gaps o una similitud menor de 0,5 (Tabla 9.1). Estos métodos nos permiten filtrar los alineamientos y trabajar solo con aquellas hipótesis de homología entre caracteres más robustas, lo cual tiene un impacto muy positivo sobre cualquier análisis posterior, incluyendo la reconstrucción filogenética [5].

### 9.2.2. Modelos de evolución, el engranaje

Los *modelos evolutivos* son descripciones matemáticas de la evolución de las secuencias y constituyen el engranaje que nos permite conectar los datos (alineamientos) con los métodos de reconstrucción filogenética. En función de lo que consideremos la unidad de evolución, existen modelos para nucleótidos, aminoácidos o codones.

Generalmente asumimos que las posiciones de un alineamiento (nucleótidos, aminoácidos o codones) evolucionan de forma independiente, y aunque sabemos que ésta no es una asunción completamente realista, simplifica enormemente la complejidad computacional de inferir filogenias. La sustitución entre nucleótidos, aminoácidos o codones se modela como un evento aleatorio y los cambios entre los estados se describen mediante una cadena de Markov tiempo-continua [3]. La propiedad más importante de las cadenas de Markov es que no tienen memoria, es decir, que la probabilidad de cambio entre nucleótidos sólo depende del nucleótido actual y no de aquéllos nucleótidos que pudo haber en esa posición con anterioridad. Lo mismo se aplica también a los modelos de aminoácidos y codones. Además de estas asunciones, los modelos utilizados con mayor frecuencia (y que tratamos en este capítulo), asumen ciertas propiedades del proceso de evolución molecular, que se considera globalmente estacionario, reversible, y homogéneo (para mayor detalle: [20]).

Algunos de estos modelos son bastante sencillos (emplean pocos parámetros), mientras que otros son muy complejos. Los modelos tienen dos tipos parámetros principales: las probabilidades de cambio

entre estados (ya sean nucleótidos, aminoácidos, o codones) y las frecuencias de estos nucleótidos, aminoácidos, o codones en nuestro alineamiento. En los modelos de nucleótidos y codones, los valores de estos parámetros se estiman a partir de los datos observados.

El modelo de evolución de nucleótidos más sencillo es el conocido como *Jukes-Cantor* (JC) y asume que cualquier cambio entre nucleótidos tiene la misma probabilidad de ocurrir y que las frecuencias de los cuatro nucleótidos son iguales [21]. Bajo las asunciones de estacionariedad, reversibilidad y homogeneidad, el modelo más complejo es GTR (del inglés *general time reversible*), que utiliza un parámetro distinto para modelar cada una de las sustituciones entre nucleótidos y sus frecuencias [44]. Entre ambos extremos tenemos muchos otros modelos, como K80, F81, F84, HKY85 o TN93 [47].

En el caso de los modelos de reemplazamiento de aminoácidos, en cambio, los parámetros generalmente no se estiman a partir de los datos, sino que vienen previamente fijados. Decimos que son *modelos no paramétricos*. La razón para ello es sencilla: al ser el alfabeto de aminoácidos más extenso que el de nucleótidos (20 vs 4), el número de parámetros a estimar se incrementa mucho y su estimación a partir del alineamiento produciría estimas con varianzas elevadas, lo cual no es conveniente. Por esta razón se utilizan matrices de reemplazamiento precalculadas, que han sido estimadas a partir de grandes conjuntos de datos [47]. Algunas de estas matrices son Blosum, Dayhoff, JTT, LG, MtArt, MtREV o WAG. En algunos casos, se permite utilizar las frecuencias observadas de aminoácidos, lo que añade 19 parámetros al modelo (se indica como +F en la selección de modelos; ver más abajo).

Además de modelar las probabilidades de cambio entre nucleótidos (o aminoácidos) y sus frecuencias relativas, con frecuencia se utilizan parámetros adicionales para tener en cuenta el hecho de que distintas partes del alineamiento pueden evolucionar a tasas distintas. En muchos casos, la adición de estos parámetros incrementa notablemente la capacidad del modelo para explicar nuestros datos, mejorando así la inferencia filogenética [47]. La variación de tasas entre sitios puede tenerse en cuenta de formas diferentes. La primera es añadir un parámetro para modelar la variación de tasas entre las posiciones del alineamiento. Esta variación se describe mediante una distribución gamma que se especifica con un único parámetro (alfa) [47]. Este parámetro podemos encontrarlo en la forma +G, “*gamma shape*”, “*alpha shape*” o “*rate variation across sites*”. La segunda forma es asumir que una proporción de las posiciones en nuestro alineamiento son invariables (completamente conservadas) [35]. Este parámetro podemos encontrarlo como +I, “*p-inv*” o “*proportion of invariable sites*”. Ambos parámetros pueden también utilizarse conjuntamente (+I+G).

Ante la gran variedad de modelos posibles, la pregunta obvia es *¿qué modelo debemos elegir?* La respuesta es que depende del conjunto de datos que estemos analizando. Para cada conjunto de datos es necesario identificar el modelo que mejor se ajuste, pues cuanto más cercano sea éste a la realidad de los datos, más posibilidades tendremos de inferir una buena filogenia. Cuantos más factores queramos tener en cuenta, más parámetros tendremos que incluir en el modelo. En principio, uno podría estar tentado de elegir siempre el modelo más complejo, pero entonces nos topamos con un problema estadístico. A mayor número de parámetros, mayor será la varianza con la que se estimen, es decir, las estimas serán menos fiables. Por el contrario, un modelo sencillo (con pocos parámetros) permitirá hacer estimas muy estables, pero se ajustará peor a los datos. Por tanto, es necesario encontrar un balance entre el ajuste a nuestros datos y la complejidad del modelo.

Existen distintos criterios que pueden utilizarse para elegir el modelo que mejor se ajusta a nuestros datos, al mismo tiempo que sopesamos su complejidad (el número de parámetros). Tradicionalmente se han venido utilizando los contrastes basados en la razón de verosimilitudes (*likelihood ratio test*, LRT). Los LRT comparan modelos por pares y requieren que las hipótesis estén anidadas, es decir, que el modelo nulo (aquel con menos parámetros) sea un caso particular del modelo alternativo [19]. Estos contrastes son muy útiles para testar ciertos tipos de hipótesis filogenéticas (ver la Sección 9.3), aunque

se ha desaconsejado su uso para la selección de modelos evolutivos en favor de otras alternativas como el criterio de información de Akaike (*Akaike information criterion*, AIC) [34]. El AIC es una medida de la cantidad de la información que perdemos al explicar los datos observados utilizando un modelo en particular. Por tanto, el modelo que mejor se ajusta a los datos observados es aquél que tiene un valor de AIC menor. A diferencia del LRT, el AIC permite comparar más de dos modelos simultáneamente y no requiere que estén anidados. El AIC se calcula como:

$$AIC = 2k - 2 \ln L \quad (9.1)$$

donde  $L$  es el valor de verosimilitud y  $k$  el número de parámetros del modelo.

La selección del modelos en función de AIC (y otros criterios) puede hacerse mediante los programas jModelTest (para secuencias de nucleótidos) [32], ProtTest (para secuencias de aminoácidos) [1] o ModelGenerator (para ambos) [22]. Estos programas calculan la verosimilitud de todos los modelos para un alineamiento dado y utilizan el AIC (u otro criterio) para identificar el modelo que mejor se ajusta a los datos. Para mayor detalle acerca de la selección de modelos leer el texto de David Posada [33].

En la actualidad es frecuente utilizar conjuntos de datos que combinan varios genes, con el objeto de tener grupos de datos independientes y más caracteres totales con los que reconstruir la historia evolutiva. En estos casos, en lugar de utilizar un único modelo para el conjunto de datos global, se suelen utilizar modelos evolutivos independientes para cada gen, de modo que se puedan tener en cuenta sus distintas propiedades evolutivas. Este tipo de *particiones* se puede aplicar no sólo a genes, sino a regiones de éstos (por ejemplo hélices transmembrana) o a posiciones de los codón (por ejemplo usando un modelo diferente para las tercera posiciones, que evolucionan a mayores tasas). Frecuentemente, el número de particiones posibles es enorme, por lo que es preciso utilizar un criterio explícito para su elección. Al igual que en la selección de modelos evolutivos, podemos utilizar el valor de AIC para elegir el esquema de partición que mejor se ajusta a nuestros datos, para lo cual podemos utilizar el programa PartitionFinder [26].

Todos los modelos de evolución molecular realizan ciertas asunciones, pero gracias a que éstas son explícitas, pueden ser evaluadas estadísticamente (trabajamos en un marco probabilístico) y así podemos comprender mejor el proceso evolutivo. Por eso no sólo existen modelos para optimizar la reconstrucción filogenética, sino también otros que nos permiten caracterizar el proceso evolutivo en sí mismo. Así, estos otros modelos, que veremos en la Sección 9.3, nos permiten responder a preguntas tales como si ha habido selección positiva o si dos caracteres han coevolucionado.

### 9.2.3. Métodos de distancias: rápidos pero no tan fiables

La forma más sencilla de medir la distancia genética es contar el número de diferencias entre dos secuencias, lo que conocemos como *distancia observada* (o distancia  $p$ ), que se expresa como el número de diferencias por posición. Pero esta forma de calcular distancias tiene un problema: a medida que aumenta la divergencia entre las secuencias, la probabilidad de que ocurran cambios superpuestos es también mayor. Los *cambios superpuestos* enmascaran cambios previos ocurridos en esa posición, por lo que la distancia observada en estos casos es una subestimación de la distancia real. Además, por azar, puede ocurrir que cambios superpuestos hayan dado lugar al mismo nucleótido en dos secuencias distintas. En ese caso, la observación de un mismo nucleótido no se debería a descendencia común, sino que representaría una convergencia, dando lugar a lo que conocemos como *homoplásia*. La homoplásia introduce ruido en los datos, afectando seriamente a la estimación de las distancias genéticas y la

inferencia filogenética. A medida que aumenta la divergencia entre dos secuencias, aumenta también la subestimación del número real de cambios calculados como distancias observadas. Cuando esto ocurre, decimos que las secuencias están saturadas o que hay *saturación*. Los métodos basados en distancias pueden utilizar los modelos de evolución para intentar corregir las distancias observadas y aliviar el problema de la saturación.

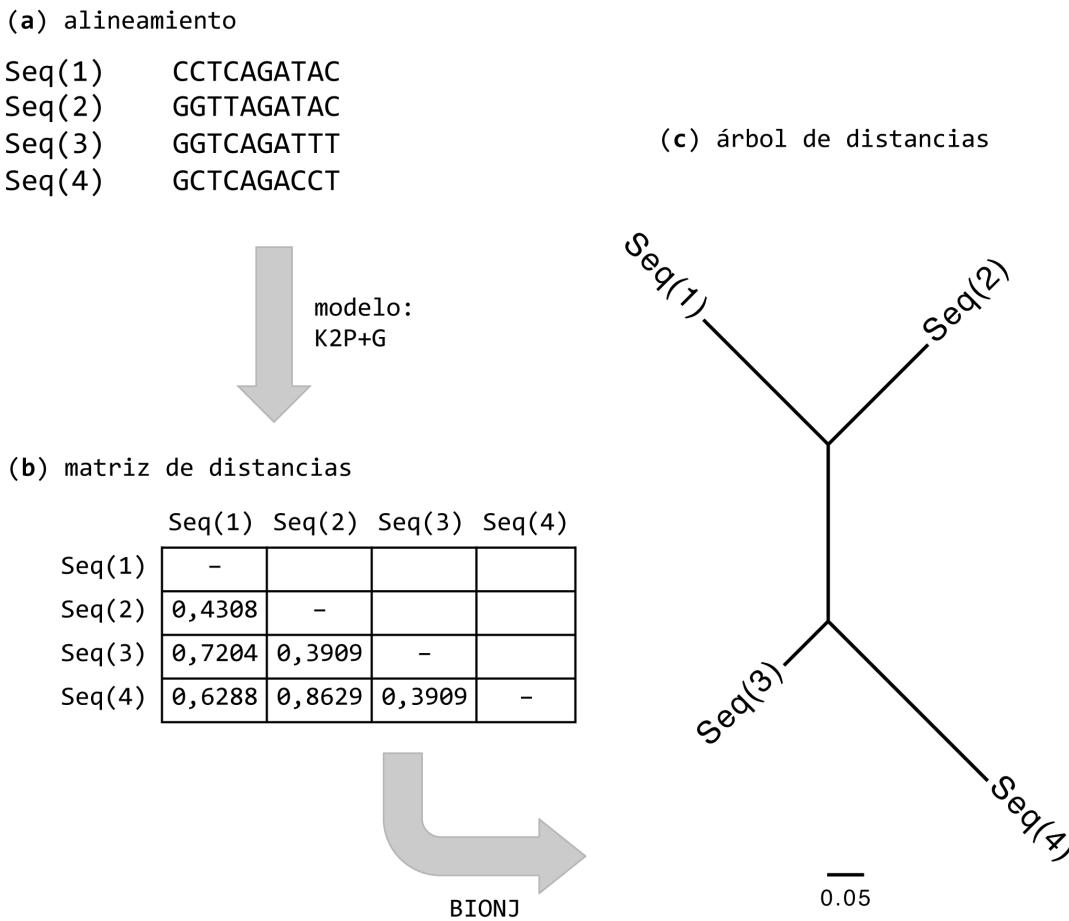
Los métodos que reconstruyen árboles filogenéticos a partir de distancias genéticas operan en dos pasos (Figura 9.4). En un primer lugar, se calculan las distancias genéticas entre todos los pares de secuencias, y se resume esta información en una matriz de distancias. Posteriormente, se utilizan los valores de esta matriz para reconstruir un árbol filogenético. Idealmente, las distancias genéticas han de reflejar las divergencias evolutivas reales.

Si asumiéramos que todas las secuencias han evolucionado con la misma tasa (lo que conocemos como la *hipótesis del reloj molecular*), las distancias genéticas serían directamente proporcionales al tiempo. Asumiendo además que las distancias genéticas reflejan divergencia real entre las secuencias, la reconstrucción filogenética se convertiría en un problema trivial (una simple regla de tres) y podríamos resolverlo con un método sencillo. Sin embargo, las distancias genéticas raramente son un buen estimador de la divergencia real y en muchos casos, la tasa de cambio en distintas secuencias es variable.

La reconstrucción de un árbol filogenético a partir de una matriz de distancias puede hacerse utilizando varios algoritmos. Uno de los métodos de distancias más conocido es *Neighbour-Joining* (NJ) [38] y una de sus variantes es BIONJ [15], que tiene en cuenta el hecho de que divergencias mayores acarrean varianzas mayores. Estos métodos no asumen la existencia de un reloj molecular para la reconstrucción filogenética.

Los métodos de distancias tienen la ventaja de ser muy rápidos, y resultan útiles como parte de un análisis preliminar. Los árboles estimados con métodos de distancias pueden ser similares o iguales al árbol estimado con métodos probabilísticos si partimos de unos datos limpios (con poco ruido), pero tienen un rendimiento bastante malo cuando las asunciones del modelo evolutivo son violadas por los datos [43]. Además, los métodos de distancias no estiman la confianza que tenemos sobre el árbol generado, es decir, no nos dicen si existen hipótesis alternativas (otros árboles) que pudieran explicar las relaciones entre las secuencias de modo similar. El mayor problema de los métodos de distancias es que pierden gran cantidad de información al resumir todas las diferencias entre las secuencias del alineamiento en un único número (la distancia genética), y también que son más sensibles al problema de atracción de ramas largas (ver más adelante). Una vez calculada la distancia genética, los métodos de distancias no nos permiten ir hacia atrás para ver en qué parte del árbol ocurrió un determinado cambio en las secuencias.

Los métodos como NJ producen siempre un único árbol que viene determinado por la matriz de distancias. Debido a los problemas asociados a las distancias genéticas que hemos mencionado, los métodos como NJ pueden generar árboles que difieran de la filogenia verdadera. Además, el algoritmo de NJ no garantiza encontrar el mejor árbol (según la matriz de distancias), pudiendo quedar atrapado en mínimos locales. En estos casos, es posible encontrar árboles similares mediante pequeñas reorganizaciones locales (conocidas como búsquedas heurísticas; ver más adelante) y utilizar un criterio de optimización para decidir qué árbol, dentro del conjunto de árboles similares, es el más adecuado. El método de mínima evolución, por ejemplo, prefiere el árbol más corto (el que con la suma de todas longitudes de rama menor) como explicación de la historia evolutiva [37].



**Figura 9.4:** Reconstrucción filogenética a partir de distancias genéticas. Partiendo de un alineamiento múltiple (a), se calcula una matriz de distancias (b) que es utilizada para estimar el árbol filogenético (c), en este caso mediante BIONJ.

#### 9.2.4. Métodos probabilísticos basados en caracteres

A diferencia de los métodos basados en distancias, los basados en caracteres tienen en cuenta los cambios que ocurren en cada posición del alineamiento, y hacen por tanto un uso más eficiente de la información [43]. Su funcionamiento se fundamenta en *criterios de optimización*: en un primer paso se generan árboles utilizando ciertos algoritmos heurísticos, y en un segundo paso estos árboles se evalúan utilizando una función objetiva que nos permite elegir el mejor árbol entre todos los generados en el paso anterior.

En principio, lo deseable sería poder comparar todos los árboles posibles utilizando nuestro criterio de optimización e identificar cuál de entre todos ellos es el mejor. Sin embargo, evaluar todos los posibles árboles es generalmente muy poco práctico, ya que este número crece de forma factorial con respecto al número de secuencias. Por ejemplo, existen tres árboles sin raíz posibles para explicar las relaciones entre cuatro secuencias, pero el número de árboles posibles para 10 secuencias es de más de dos millones, y para 80 secuencias aumenta hasta  $2,18 \times 10^{137}$  (un número mucho mayor que el número estimado de protones en el Universo,  $10^{89}$ !). En la práctica, no es factible comparar todos los árboles posibles si

no es para un número de secuencias menor de 6-7. De forma general, se utilizan *métodos heurísticos* que aunque no garantizan encontrar el mejor árbol globalmente, pero permiten evaluar un número significativo de árboles más probables. Las búsquedas heurísticas utilizan algoritmos que partiendo de un árbol (por ejemplo, aleatorio o reconstruido con NJ) realizan pequeñas reorganizaciones locales para encontrar árboles más probables. Durante las búsquedas heurísticas, las distintas reorganizaciones se aceptan o rechazan con ayuda del criterio de optimización, y la búsqueda continúa hasta que no se encuentren más mejoras significativas.

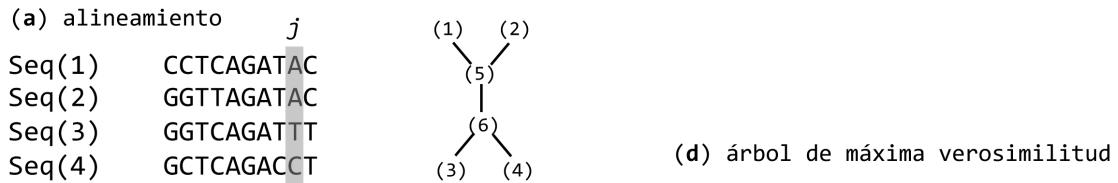
## Máxima verosimilitud

La aplicación del principio de *máxima verosimilitud* (MV) a la reconstrucción filogenética [10, 11] supuso un gran avance al permitir estudiar las relaciones evolutivas desde un punto de vista estadístico. Al trabajar en un *marco probabilístico*, es posible estimar patrones históricos e inferir parámetros intrínsecos a los procesos de evolución molecular, y el contraste de hipótesis filogenéticas se realiza de forma directa. El principio de MV se define como la probabilidad de que un modelo de evolución dado (de nucleótidos, codones o aminoácidos) y una historia evolutiva hipotética (el árbol) hayan dado lugar a los datos observados (alineamiento de secuencias) [11]. El árbol de MV es aquél que tiene la mayor probabilidad de haber generado los datos observados, y por tanto se prefiere como hipótesis filogenética. El método de MV es eficiente y consistente. Esto significa que la varianza de las estimas es menor que en otros métodos (están menos afectadas por el error de muestreo) y que las estimas convergen al valor correcto del parámetro conforme aumentamos el número de caracteres [14]. Estudios de simulación han demostrado que la reconstrucción filogenética por MV es bastante robusta, aunque no inmune, a posibles violaciones de las asunciones del modelo por parte de los datos [43].

La MV trata de inferir la *historia evolutiva más probable*. Como hemos visto anteriormente, para calcular esta probabilidad es necesario utilizar los *modelos evolutivos*, pues son el engranaje que describe la evolución de las secuencias en términos de probabilidad. A continuación veremos el fundamento del cálculo del valor de verosimilitud de forma intuitiva y su aplicación a la reconstrucción filogenética. Los modelos evolutivos generalmente asumen que el proceso de evolución molecular es reversible. Esto permite que el valor de verosimilitud de un árbol sea independiente de la posición de la raíz [40]. Además, hemos asumido que las posiciones evolucionan de forma independiente en el alineamiento. Esta asunción nos permite calcular la verosimilitud de forma independiente para cada posición del alineamiento y combinar posteriormente estas verosimilitudes en un valor final.

El cálculo de verosimilitud para cada posición se realiza teniendo en cuenta todos los posibles escenarios que han podido dar lugar a los datos observados (para esa posición) [43]. Esto implica sumar las probabilidades de cada combinación única de estados ancestrales (reconstrucciones hipotéticas en los nodos internos) y longitudes de rama (Figura 9.5). Obviamente, ciertos escenarios son mucho más probables que otros. Por ejemplo, en la Figura 9.5(b), la probabilidad de encontrar A en el ancestro de 1 y 2 (nodo 5) es mayor que la probabilidad de encontrar C, G, o T. Sin embargo, al calcular la verosimilitud se contemplan todas las posibilidades aunque su probabilidad sea muy baja. Puesto que la probabilidad de un escenario concreto (entre todos los posibles) es extremadamente pequeña, y que éstos valores de verosimilitud son a su vez el producto de muchas probabilidades, en la práctica trabajamos con los logaritmos naturales de las verosimilitudes ( $\ln L$ ). Una vez que hemos calculado la verosimilitud para cada posición en el alineamiento, la verosimilitud total del árbol se calcula como la suma de los logaritmos de las verosimilitudes para cada sitio (Figura 9.5).

La verosimilitud no debe confundirse con una probabilidad, a pesar de que se define en términos probabilísticos. La *verosimilitud* es la probabilidad del evento observado, que se calcula utilizando un modelo evolutivo que asumimos como correcto. Sin embargo, la verosimilitud no tiene nada que ver con



(d) árbol de máxima verosimilitud

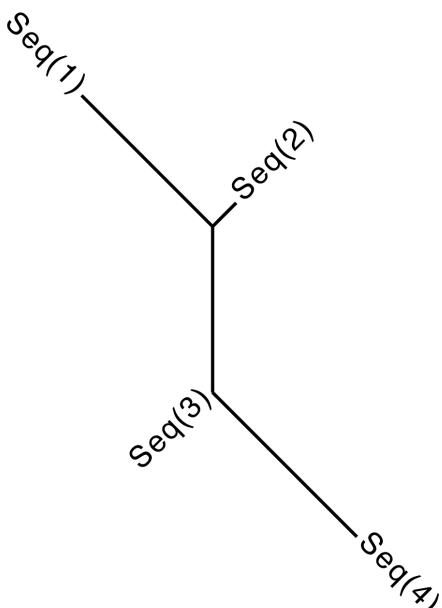
(b) verosimilitud para el sitio  $j$

$$L(j) = \text{prob} \begin{bmatrix} A & A \\ A & \\ \diagdown & \diagup \\ T & C \end{bmatrix} + \text{prob} \begin{bmatrix} A & A \\ A & \\ \diagdown & \diagup \\ T & T \end{bmatrix} + \text{prob} \begin{bmatrix} A & A \\ A & \\ \diagdown & \diagup \\ T & C \end{bmatrix} + \\ + \text{prob} \begin{bmatrix} A & A \\ A & \\ \diagdown & \diagup \\ T & G \end{bmatrix} + \text{prob} \begin{bmatrix} A & A \\ A & \\ \diagdown & \diagup \\ T & T \end{bmatrix} + \dots + \dots$$

(c) verosimilitud para el alineamiento

$$\ln L(\text{total}) = \ln L(1) + \ln L(2) + \dots + \ln L(N) = \sum_{j=1}^N \ln L(j)$$

0.08



**Figura 9.5:** Reconstrucción filogenética por máxima verosimilitud. Dados unos datos y un árbol (a), se calculan los valores de verosimilitud para cada sitio de forma independiente (b), y éstos se combinan en un valor final (c). El valor de verosimilitud nos permite evaluar los árboles generados durante la búsqueda heurística y elegir el árbol de máxima verosimilitud (d).

que el modelo que asumimos describa correctamente o no el proceso evolutivo real. De ahí la importancia de elegir el modelo que mejor se ajusta a nuestros datos. Es más, el hecho de que obtengamos distintos valores de verosimilitud al asumir modelos distintos es el fundamento de la selección de modelos y el contraste de hipótesis basado en verosimilitud ([19, 33]; volveremos a ello más adelante).

Como decíamos antes, los métodos de búsqueda heurística nos permiten “visitar” un cierto número de árboles entre todos los posibles, y valorar si un árbol es mejor o peor en función de un criterio de optimización, en este caso su verosimilitud. El árbol de MV será aquél que tiene el mayor valor de verosimilitud (menor  $\ln L$ ) entre todos los árboles evaluados durante la búsqueda heurística. Esperamos que la búsqueda heurística nos permita encontrar el mejor árbol entre todos los posibles (alcanzar el máximo global). Sin embargo, una búsqueda heurística nunca nos asegurará que hemos encontrado el mejor árbol, pues durante el proceso de optimización puede quedar atrapada en un máximo local, lo cual nos generará un árbol subóptimo. Para evitar este problema, podemos realizar varias búsquedas de MV (por ejemplo, 10 o 100) a partir de otros tantos árboles iniciales. Realizar múltiples búsquedas heurísticas nos permite explorar de forma más eficiente el conjunto de árboles posibles, y en última instancia, nos permite comparar los distintos árboles de MV encontrados en cada búsqueda heurística.

Si las diferentes búsquedas heurísticas (que parten de puntos independientes) coinciden en los árboles de MV obtenidos, tendremos mayor certeza de que hemos explorado bien el conjunto de posibles árboles, aumentando por tanto la confianza en la robustez de nuestra estima.

Existen multitud de programas con los que podemos realizar análisis de MV. Entre los más conocidos y utilizados están PhyML o Tree-puzzle, y otros especialmente útiles cuando utilizamos alineamientos grandes, como son RAxML o GARLI (Tabla 9.1).

## Métodos Bayesianos

La estadística Bayesiana, a diferencia de la estadística clásica, considera los parámetros de un modelo como variables aleatorias con una distribución estadística y no como constantes desconocidas. Puesto que desconocemos los valores de los parámetros del modelo, desde un punto de vista Bayesiano resulta más razonable especificar a priori unas distribuciones para describir los valores que éstos puede tomar. La *inferencia Bayesiana* (IB) de la filogenia se basa en el cálculo de la probabilidad posterior de un árbol, esto es, la probabilidad de que dicho árbol sea correcto dados unos datos y un modelo. La probabilidad posterior se calcula mediante el *teorema de Bayes*, según el cual la probabilidad posterior de un árbol  $P(T, \theta|D)$  es proporcional a su probabilidad previa  $P(T, \theta)$  multiplicada por su verosimilitud  $P(D|T, \theta)$  (o probabilidad de los datos dados un árbol y modelo evolutivo):

$$P(T, \theta|D) = \frac{P(T, \theta)P(D|T, \theta)}{P(D)} \quad (9.2)$$

donde el numerador  $P(D)$ , que corresponde a la probabilidad marginal de los datos, que es un factor de normalización. La probabilidad marginal de los datos es una suma sobre todas las posibles topologías, y para cada topología una integral sobre todas las longitudes de rama y parámetros del modelo, cuya función es que la suma de probabilidades posteriores de los árboles y sus parámetros sea uno.

Para entender mejor cómo funciona la IB, observemos el gráfico en la Figura 9.6(b), donde se representan en una única dimensión un conjunto de árboles posibles (eje x) en función de su probabilidad posterior (eje y). En la curva, cada punto representa una combinación única de topología y longitudes de rama. Obviamente, esta representación no es real, ya que el conjunto de árboles posibles es mucho más complejo y la distribución de las probabilidades posteriores es multimodal. La IB trata de determinar el área bajo esta curva, que corresponde a la probabilidad posterior de cada árbol concreto. En la Figura 9.6(b), podemos observar tres regiones de probabilidad posterior elevada, que corresponden a tres topologías distintas con su combinación particular de longitudes de rama.

Como hemos dicho anteriormente, el cálculo de la probabilidad posterior implica evaluar todos los posibles árboles y, para cada árbol, investigar todas las posibles combinaciones de longitudes de ramas y parámetros del modelo evolutivo. De nuevo, este cálculo resulta prohibitivo computacionalmente. En el caso de la IB, utilizamos algoritmos MCMC (o *Markov Chain Monte Carlo*) para obtener una aproximación de la distribución de la probabilidad posterior [16, 28]. Los MCMC son algoritmos de simulación estocástica que evitan el cálculo directo de las probabilidades posteriores y permiten obtener una muestra de la distribución posterior. De manera análoga a cómo opera la búsqueda heurística por MV, las cadenas MCMC de la IB parten de un árbol al azar (combinación al azar de topología, longitudes de rama y parámetros del modelo) y realiza un cierto número de visitas a árboles concretos. De este modo, los algoritmos MCMC modifican ligeramente una de las variables del árbol inicial (ya sea la posición o longitud de una rama en concreto, o el valor de un parámetro del modelo evolutivo) y evalúan el nuevo árbol. Esto es lo que se conoce como una generación. Tras cada generación, el cambio propuesto se acepta o rechaza en función del valor de la razón entre las probabilidades posteriores de

los estados actual y anterior [36]. Si esta razón es mayor de uno (el nuevo estado es más probable), el cambio se acepta y la cadena MCMC continúa desde este punto. Si por el contrario, la razón es menor de uno, el cambio puede aceptarse o rechazarse en función del valor que toma la razón entre probabilidades posteriores. Recordemos ahora lo que decíamos al hablar de los modelos evolutivos, sobre que los procesos de Markov no tenían memoria. Durante varias generaciones, la cadena MCMC continúa de esta manera y tras haber transcurrido el tiempo suficiente, se espera que la cadena MCMC muestre cada valor de topologías, longitudes de rama y parámetros del modelo un número de veces proporcional a su probabilidad posterior. Es decir, si las relaciones mostradas en la Figura 9.6 tienen una probabilidad posterior de 0.82, la cadena MCMC habrá muestreado esta topología en aproximadamente el 82 % de las generaciones.

Debemos tener en cuenta que la aproximación de la probabilidad posterior será más exacta cuantas más generaciones se evalúen en la cadena de MCMC. Durante las generaciones iniciales de las cadenas MCMC, los árboles suelen tener una probabilidad posterior baja como resultado de haber comenzado a partir de combinaciones aleatorias de topología, longitudes de rama y valores de parámetros. Esto es lo que conocemos como la *fase de burnin* o *calentamiento*. Tras esta fase inicial, las cadenas MCMC alcanzan una fase estacionaria donde los árboles muestreados tienen una probabilidad posterior elevada. Para la estimación del árbol final mediante IB es necesario descartar los árboles del *burnin* y todo el resto de árboles con probabilidad posterior elevada se resumen mediante un árbol consenso de mayoría [36].

Como ocurre con las búsquedas heurísticas de MV, las cadenas MCMC pueden quedar atrapadas en máximos locales. Para evitarlo, la IB utiliza varias cadenas MCMC simultáneamente y además se recomienda hacer cada análisis de IB por duplicado. Los programas de IB como MrBayes [36] generalmente implementan cuatro cadenas MCMC, una cadena “fría” y tres “calientes”. Las cadenas “calientes” elevan la probabilidad posterior de los árboles visitados a una potencia entre 0 y 1, favoreciendo así el salto entre picos locales de probabilidad posterior. Durante el proceso de búsqueda, las cadenas se intercambian, de modo que en cada momento la cadena con mayor probabilidad posterior se comporta como “fría” y el resto como “calientes”. Esto favorece una exploración más exhaustiva del conjunto de árboles posibles, puesto que se facilita el salto entre picos locales de probabilidad posterior elevada y con ello se reduce la posibilidad de que la cadena “fría” quede atrapada en máximos locales. Para el resultado final, sólo se tiene en cuenta los resultados de la cadena “fría”, pues incluye a los árboles con mayor probabilidad posterior entre todos los visitados por las cuatro cadenas.

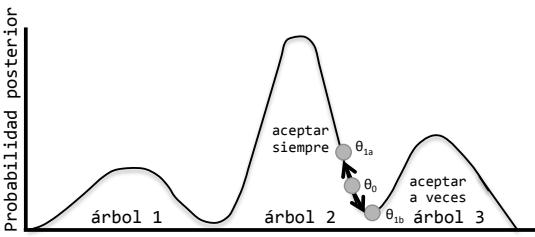
Para saber si las cadenas MCMC de nuestro análisis de IB han explorado de modo eficiente el conjunto de árboles es necesario estudiar la convergencia de las cadenas “frías” entre los dos análisis de IB. La forma más sencilla es observar sus diferencias medias, que en MrBayes conocemos como “*average standard deviation of split frequencies*”. Una diferencia entre las desviaciones estándar medias menor de 0.01 suele indicarnos que los dos análisis de IB han convergido. Sin embargo, es muy recomendable además estudiar la convergencia de las cadenas MCMC a posteriori con herramientas adecuadas desarrolladas a tal efecto, como AWTY (Tabla 9.1) [29].

Los métodos de IB tienen una conexión muy fuerte con la MV, ya que la hipótesis preferida es aquélla con la mayor probabilidad posterior, y éste es un valor que se calcula en función de la verosimilitud. Puesto que la IB se basa directamente en la verosimilitud, comparte sus propiedades de eficiencia y consistencia [18]. Sin embargo, existen diferencias importantes entre MV e IB: la MV trata de encontrar un único árbol (el mejor, el de mayor verosimilitud), mientras que la IB integra un gran número de árboles de probabilidad posterior elevada (con sus topologías y longitudes de rama) [18]. Los árboles con una probabilidad posterior alta generalmente incluyen al mejor árbol de MV, pero también incluyen a un número de árboles cuyo valor de verosimilitud es ligeramente menor. La ventaja de integrar varios árboles es que consideramos la incertidumbre existente en la estimación de la topología, las longitudes

(a) alineamiento

Seq_A	CCTCAGATAC
Seq_B	GGTTAGATAC
Seq_C	GGTCAGATT
Seq_D	GCTCAGACCT

(b) MCMC



(c) Análisis Bayesiano con cuatro cadenas MCMC (x2)

```

1 -- [-2955.775] (-3056.207) (-2997.883) (-2998.675) * [-2911.695] (-3013.663) (-2997.978) (-3016.143)
1000 -- [-2316.864] (-2314.046) (-2322.162) (-2357.051) * [-2377.011] (-2336.706) (-2352.991) [-2301.297]
2000 -- [-2286.754] (-2314.236) (-2298.651) (-2317.366) * [-2325.212] (-2337.705) (-2327.255) [-2285.621]
3000 -- [-2280.581] (-2285.757) (-2303.145) (-2319.858) * [-2305.778] (-2318.057) (-2302.670) [-2285.094]
4000 -- [-2283.396] (-2295.878) (-2312.352) (-2303.106) * [-2286.150] (-2330.199) (-2321.260) [-2298.520]

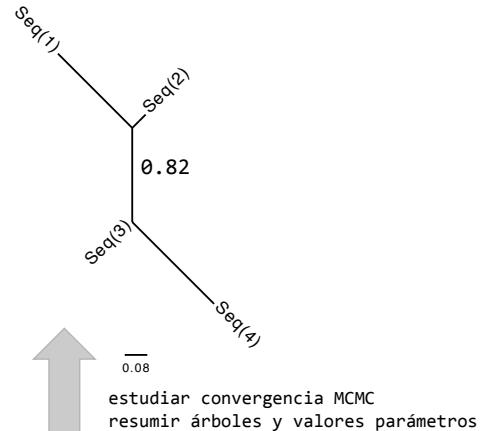
```

Average standard deviation of split frequencies: 0.132583

[...]

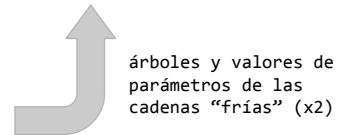
```
10000000 -- [-11441.322] (-11455.888) (-11439.830) (-11462.323) * [-11435.720] (-11458.960) (-11440.275) (-11440.950)
```

Average standard deviation of split frequencies: 0.000584



(c) Resultados de cada generación (x2)

GEN	árbol	par1	par2	parN
1000	(1,2),(3,4)	0,2760	0,1398	0,0005
2000	((1,2),(3,4))	0,2761	0,1398	0,0011
3000	((1,4),(2,4))	0,2779	0,1392	0,0012
4000	((1,2),3,4)	0,2777	0,1399	0,0010
...	...	...	...	...
10000	((1,2),(3,4))	0,2778	0,1390	0,0010



**Figura 9.6:** Inferencia Bayesiana de la filogenia. Las cadenas de *Markov Monte Carlo* (MCMC) progresan hacia árboles más probables gracias a las razones de probabilidad posterior (b). Un análisis de inferencia Bayesiana típico se realiza por duplicado y cada análisis consta de una cadena [fría] y tres (calientes) (c). Tras un número de generaciones determinado, se eliminan las generaciones de la fase burnin y los resultados de las cadenas frías se combinan en el árbol final (d). Con posterioridad, se recomienda revisar la convergencia de las cadenas MCMC.

de rama y los parámetros del modelo evolutivo. Otra diferencia importante entre la IB y el resto de métodos es que resulta necesario especificar la distribución previa de todos los parámetros [18]. Esto nos permite incorporar cierto conocimiento previo en el análisis (por ejemplo, que las transiciones son más probables que las transversiones), pero generalmente no existe tal conocimiento para la mayoría de los parámetros. Por ello, se suelen utilizar distribuciones previas equiprobables (no informativas o *flat priors*) de modo que las diferencias observables de probabilidad posterior dependen principalmente del valor de verosimilitud [36]. A pesar de la fuerte influencia de la verosimilitud en la IB, los árboles de MV e IB pueden ser distintos debido a diferencias durante el proceso de estimación, que es marginal en MV e integrada en IB [18].

## 9.3. Contrastes de hipótesis

### 9.3.1. Contrastes entre modelos evolutivos alternativos

Una de las ventajas de trabajar en un marco evolutivo probabilístico es que podemos recurrir a la estadística para realizar análisis *a posteriori* sobre las filogenias obtenidas. En la Subsección 9.2.2 ya comentamos que existen contrastes, tanto basados en la razón de verosimilitudes (LRT) como en criterios de información (como el AIC), que permiten determinar qué modelo se ajusta mejor a los datos. Además, como ya adelantamos entonces, los contrastes LRT también permiten comparar ciertos modelos que nos ayudan a comprender cómo ha sido el proceso evolutivo que ha dado lugar a las secuencias que observamos. En esta sección introduciremos tres de estos contrastes, que nos permitirán (i) testar la presencia de reloj molecular, (ii) detectar selección positiva y (iii) determinar si dos caracteres han evolucionado de forma correlacionada.

Pero antes explicaremos brevemente cómo realizar el LRT, un tipo de contraste muy recurrente. El LRT requiere que los dos modelos a comparar estén anidados, es decir, que uno de ellos sea un caso especial del otro (el modelo más complejo ha de incluir todos los parámetros del modelo más sencillo). Para realizar el contraste, es necesario calcular la verosimilitud del modelo nulo ( $L_0$ ) y del alternativo ( $L_1$ ), cuya diferencia se mide por medio del estadístico  $D$ , que no es sino el logaritmo de la razón de las verosimilitudes multiplicado por menos dos:

$$D = -2 \ln \frac{L_0}{L_1} = -2 \ln L_0 + 2 \ln L_1 \quad (9.3)$$

Cuando los modelos están anidados, el estadístico  $D$  sigue aproximadamente una distribución chi-cuadrado ( $\chi^2$ ) con grados de libertad igual a la diferencia en el número de parámetros entre ambos modelos. Si el LRT es significativo, se considera que el modelo alternativo (más complejo) se ajusta mejor a los datos; en otras palabras, se rechaza la hipótesis nula.

Para testar si unas secuencias determinadas han evolucionado de acuerdo a una tasa evolutiva constante, es decir si podemos asumir la presencia de un *reloj molecular estricto*, es posible utilizar un contraste de tipo LRT. En primer lugar, calcularemos la verosimilitud del modelo nulo, obtenida al asumir una tasa evolutiva constante (todas las hojas del árbol quedarán alineadas) y la verosimilitud del modelo alternativo donde se permite que las tasas sean variables. En este caso, el test chi-cuadrado tendrá  $s - 2$  grados de libertad, donde  $s$  representa el número de secuencias (el modelo de tasas variables tiene  $s - 1$  parámetros, mientras que el nulo tiene 1 parámetro). Los cálculos de verosimilitud de acuerdo a estos modelos pueden realizarse, por ejemplo, con los programas MEGA, PAML, TREE-PUZZLE e HYPHY (Tabla 9.1). Además de testar la existencia de un reloj molecular global, es posible testar la presencia de relojes moleculares locales, es decir, podemos testar si puede asumirse un reloj molecular en partes concretas de la filogenia.

La importancia de determinar si hay o no constancia de tasas no sólo es útil para comprender cómo ha sido el proceso evolutivo, sino que es importante a la hora de datar árboles filogenéticos. Así, la información contenida en las secuencias puede ser utilizada para estimar tiempos absolutos de divergencia entre linajes (en millones de años). Esto nos permite saber si una separación determinada entre dos linajes ha ocurrido antes o después que otro evento, o si esta separación guarda relación con cierta etapa geológica o climática del planeta. Es por tanto un asunto de gran interés. En el caso de que no podamos asumir la constancia de tasas para todas las secuencias, es posible utilizar lo que conocemos como reloj molecular relajado, que nos permite estimar tiempos de divergencia al mismo tiempo que

acomoda la variación de tasas evolutivas entre linajes [9, 39, 45]. Para conocer más acerca de la datación molecular consultar el texto de Lemey y Posada [27].

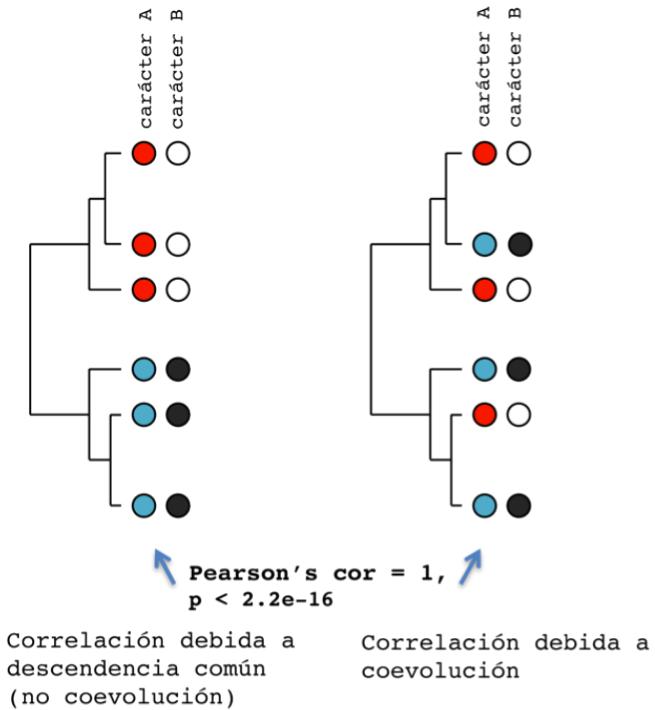
Cuando observamos la morfología de animales o plantas, la *evolución adaptativa* resulta evidente ya que las modificaciones fenotípicas seleccionadas por la evolución tienen un valor directo de adaptación al medio ambiente. A nivel molecular, sin embargo, la evolución adaptativa resulta más difícil de detectar. Aunque hay varias formas de distinguir su huella, la más común se basa en el análisis de secuencias codificantes de proteínas y la naturaleza redundante del código genético. En principio, se trata de comparar el número de cambios sinónimos (sustituciones nucleotídicas que no alteran el aminoácido codificado) por cada sitio sinónimo ( $dS$ ) y el número de cambios no sinónimos por cada sitio no sinónimo ( $dN$ ). Para ello utilizamos modelos de sustitución de codones, que tienen en cuenta factores como las frecuencias de cada codón, la frecuencia relativa de transiciones y transversiones o el sesgo en la composición nucleotídica [47].

La idea subyacente tras la medición de cambios sinónimos y no sinónimos radica en que éstos cambios se acumularán de forma distinta en función de qué fuerza selectiva opere sobre las secuencias. Cuando las secuencias evolucionan de forma neutral, los cambios se acumularán de forma aleatoria, por lo que la razón  $dN/dS$  (también conocida como coeficiente de selección u omega,  $\omega$ ) tenderá a ser 1. En cambio, bajo selección negativa (o purificadora) se espera una proporción mayor de cambios que no alteran el significado y la razón  $dN/dS$  será menor de uno. Si hay selección positiva (o evolución adaptativa), se favorecerán los cambios no sinónimos que aumenten la adaptación y por tanto  $dN/dS$  será mayor de uno. La evolución adaptativa puede observarse con relativa facilidad en genes relacionados con la respuesta inmune, pues requieren de una adaptación constante a los posibles patógenos que deben combatir y con los que coevolucionan (lo que se conoce como una *carrera armamentista* o *hipótesis de la Reina de Corazones* o *Red Queen*).

La huella de la selección positiva es con frecuencia sutil y desaparece rápidamente. En primer lugar, la evolución adaptativa suele actuar de forma episódica (en un momento y circunstancias determinadas) y después la selección se torna purificadora con el fin de mantener dicha adaptación. Por tanto, tras un episodio de evolución adaptativa, la razón  $dN/dS$  puede volver rápidamente a ser menor de uno. En segundo lugar, la selección positiva no actúa sobre toda la proteína, sino que generalmente afecta sólo a posiciones concretas que tienen importancia funcional. Por tanto, un valor de  $\omega$  puede ser menor de uno si lo promediamos para todas las posiciones, pero al mismo tiempo puede enmascarar ciertas posiciones donde omega es mayor de uno. Para tratar de resolver estas dificultades existen modelos sofisticados que nos permiten enfocar el cálculo de  $\omega$  en regiones concretas del árbol (se conocen como *modelos de ramas* o *branch models*), en las distintas posiciones de las proteínas (*modelos de sitios* o *site models*), o en ambos simultáneamente (*modelos de ramas y sitios* o *branch-site models*). Los programas más populares para realizar este tipo de análisis (y otros relacionados) son PAML y HYPHY (Tabla 9.1). El manual de usuario de PAML explica en detalle cómo realizar distintos contrastes e incluye ejemplos que nos permitirán reproducir los resultados de trabajos originales. Para conocer más acerca de los métodos que permiten detectar la selección adaptativa leer el texto de Kosakovsky, Pond et al. [25].

En tercer lugar, además de evaluar las hipótesis de reloj molecular y de evolución adaptativa, es posible utilizar contrastes de tipo LRT para determinar si dos caracteres (sean morfológicos o moleculares) han evolucionado de forma correlacionada. La *evolución correlacionada* también se conoce como coevolución o evolución dependiente. Con frecuencia, los análisis de correlación clásicos (como el coeficiente de correlación de Pearson) no pueden aplicarse directamente en sistemas biológicos. Estos métodos asumen la independencia de los datos, pero las especies y sus atributos no son independientes, sino que están relacionados mediante su filogenia [13]. Muchas de las características que comparten los seres vivos se deben a descendencia común de éstos, por lo que la presencia conjunta de ciertas características no implica necesariamente que éstas hayan coevolucionado, sino que puede sencillamente deberse a un

origen común. En el ejemplo de la Figura 9.7 se muestran dos escenarios evolutivos completamente distintos en los que en ambos el coeficiente de Pearson resulta en una correlación alta. Sin embargo, el panel izquierdo refleja que esta correlación es fruto de descendencia común, mientras que el panel derecho refleja claramente la coevolución de dos caracteres con independencia de su filogenia. Este tipo de análisis que tiene en cuenta la historia evolutiva, puede realizarse con BayesTraits (Tabla 9.1) y se basa, nuevamente, en contraste de modelos evolutivos mediante un LRT.



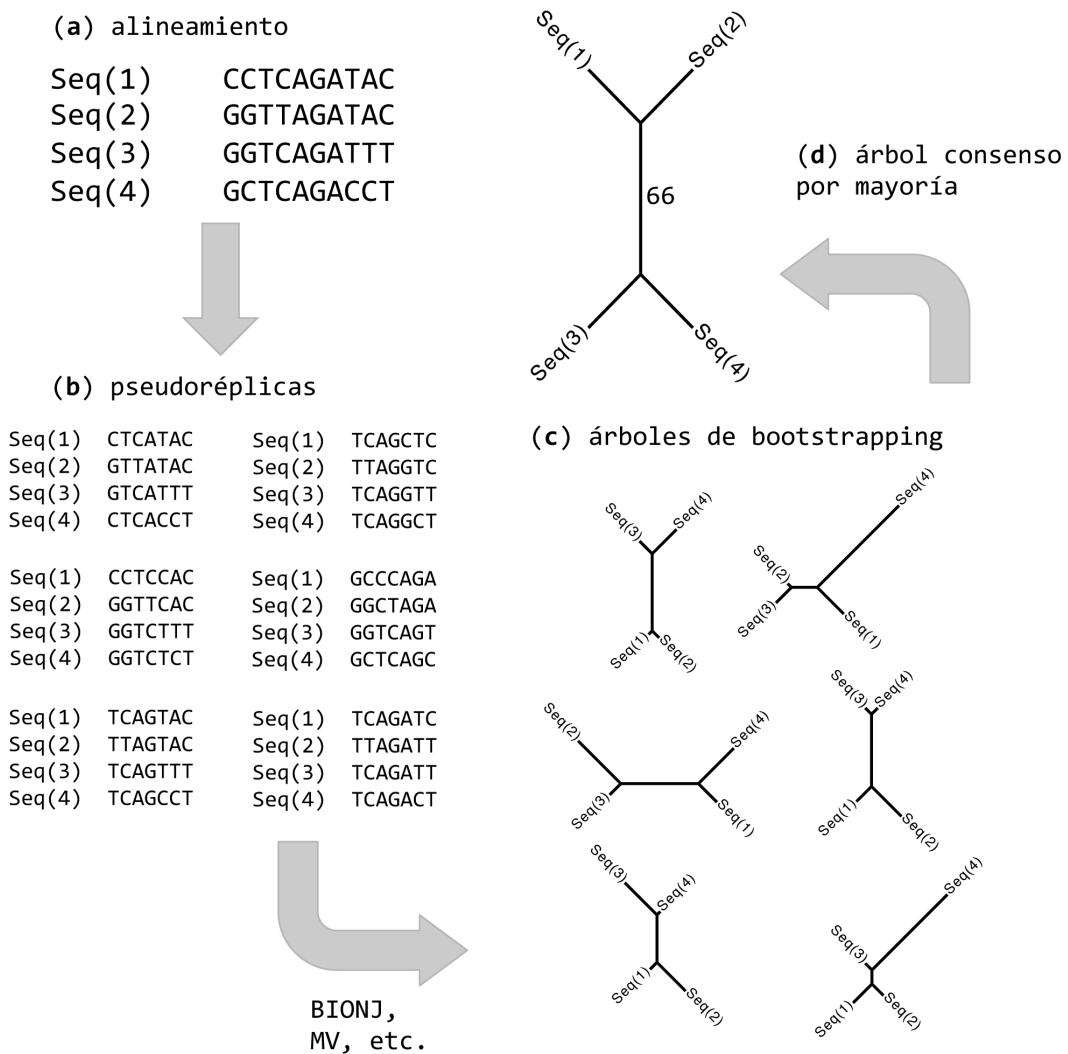
**Figura 9.7:** Cálculo de correlaciones en un contexto evolutivo. Se muestra cómo dos escenarios evolutivos completamente diferentes pueden dar lugar a un mismo valor de correlación de Pearson, si bien en un caso la correlación refleja descendencia común (a) y en otro coevolución (b).

### 9.3.2. Robustez de un árbol filogenético y contraste de árboles alternativos

Uno de los aspectos más importantes del análisis filogenético es el de poder determinar cuán fiable es el árbol que hemos obtenido, y si existen hipótesis alternativas con una fiabilidad similar. La inferencia filogenética mediante métodos de MV produce un único árbol, el árbol que con mayor verosimilitud ha dado lugar a los datos observados. Por esta razón, no conocemos de manera directa si existe otro árbol con una verosimilitud similar no significativamente diferente desde el punto de vista estadístico.

De cualquier modo, la reconstrucción filogenética por MV no define intervalos de confianza para los árboles estimados y es necesario utilizar métodos alternativos. La aproximación más corriente es utilizar el bootstrapping no paramétrico [12], que nos informa acerca de la estabilidad de las relaciones filogenéticas del árbol de MV obtenido (aunque no puede informar de la presencia de otros árboles subóptimos). El *bootstrapping no paramétrico* es una técnica estadística de remuestreo con reemplazamiento, que funciona de la siguiente manera: a partir de nuestro alineamiento original (Figura 9.8(a)) se generan pseudoréplicas muestreando aleatoriamente una proporción concreta de las columnas (sitios) del alineamiento original (Figura 9.8(b)). En este muestreo con reemplazamiento se escogen columnas

al azar para generar las pseudorélicas, pudiendo incluir una misma columna varias veces mientras que otras nunca se incluirán. Una vez que se han generado un cierto número de pseudorélicas (por ejemplo, 1000), se infiere un árbol filogenético con un método particular (por ejemplo, MV) para cada una de ellas, de manera que obtenemos 1000 árboles de MV (Figura 9.8(c)). El conjunto de árboles obtenidos a partir de las pseudorélicas se resume en un árbol final que conocemos como árbol de consenso por mayoría (Figura 9.8(d)). Este árbol incluirá un número (típicamente expresado en %) para cada uno de los nodos del árbol en función del número de veces que se encuentre esta relación en particular en el conjunto de los 1000 árboles de *bootstrap*. Generalmente se considera que un valor igual o mayor al 70 % es un soporte elevado para las relaciones de parentesco implicadas ([17]).



**Figura 9.8:** Bootstrapping no paramétrico para estimar la robustez de un árbol. El alineamiento original (a) se remuestra al azar y con reemplazamiento N veces, obteniendo N alineamientos con la misma longitud (b). Para cada uno de estos alineamientos se estima el mejor árbol, ya sea por MV u otros métodos, obteniéndose N árboles (c). Finalmente, se analizan los N árboles y se genera un árbol consenso (d).

A diferencia de los métodos filogenéticos de MV, el resultado final de la IB tiene en cuenta un número elevado árboles, lo que nos permite realizar una estima de la robustez de nuestro árbol final gracias al

proceso de búsqueda por MCMC. Para ello, hacemos uso de los árboles obtenidos en las generaciones tras la convergencia (la fase estacionaria), que se analizan *a posteriori* para calcular la frecuencia con la que las relaciones de parentesco ha sido obtenidas en el conjunto de árboles. Esta frecuencia se expresa mediante valores de probabilidad posterior para cada uno de los nodos, de modo análogo a la frecuencia calculada a partir de las réplicas de *bootstrapping* no paramétrico. En el caso de probabilidades posteriores, se considera que un nodo en particular es fiable cuando su probabilidad posterior es superior a 0.95.

Además de estimar la fiabilidad de los árboles filogenéticos, es posible utilizar test estadísticos para saber si nuestros datos, además de favorecer una hipótesis filogenética concreta (nuestro árbol reconstruido) podrían soportar o rechazar otras hipótesis filogenéticas, como las propuestas en estudios previos. Este tipo de análisis son muy interesantes, y pueden aplicarse, entre otras cosas, para conocer si nuestros datos de secuencia apoyan o rechazan determinadas relaciones de parentesco derivadas del análisis de datos morfológicos o de otro tipo.

Existen varios métodos para comparar la topología de árboles filogenéticos. Los más empleados utilizan el remuestreo de caracteres y la comparación de las verosimilitudes de dos árboles alternativos. Sin embargo, dado que las topologías no son hipótesis anidadas (un árbol generalmente no representa un caso particular de otro árbol), las diferencias de verosimilitud entre árboles no pueden aproximarse mediante una distribución chi-cuadrado y no es posible utilizar un LRT como hemos explicado hasta ahora.

Una posibilidad para comparar hipótesis no anidadas mediante LRT es utilizar el *bootstrapping* paramétrico, que nos permite determinar la distribución nula del estadístico D [19]. El *bootstrapping* no paramétrico, también conocido como simulación de Monte Carlo, consiste en generar réplicas de los datos y estimar mediante un método como la MV un conjunto de árboles (con su topología, longitudes de rama y parámetros del modelo) que formarán la distribución nula. En este caso, el LRT será significativo si las diferencia entre las verosimilitudes de los árboles de la distribución nula y la hipótesis alternativa superan el 5 % de los casos [19]. El *bootstrapping* paramétrico puede resultar computacionalmente muy intensivo, especialmente para grandes conjuntos de datos. Por esta razón, la comparación de topologías alternativas suele realizarse mediante contrastes de topologías basados en *bootstrapping* no paramétrico.

Los test de topologías alternativas más populares son tres. El *contraste de Kishino-Hasegawa* (KH) [24] permite comparar la diferencia de verosimilitudes entre dos árboles seleccionados *a priori*. Este contraste estadístico sólo es válido si los árboles se escogen previamente y no se derivan del mismo alineamiento que se utilizará para compararlos posteriormente. El *contraste de Shimodaira-Hasegawa* (SH) [42] permite comparar un conjunto de árboles seleccionados *a posteriori*, lo cual nos permite incluir el árbol de MV obtenido a partir de nuestro alineamiento. Sin embargo, el contraste SH es muy conservador y sus resultados son muy dependientes del número de árboles considerados. Existe un tercer tipo de contraste conocido como *test aproximadamente no sesgado* (AU, del inglés *approximately unbiased test*) que está menos sesgado que los contrastes desarrollados con anterioridad (KH, SH), y por ello es el más utilizado actualmente [41]. El contraste AU utiliza una aproximación de *bootstrap* multiescala para controlar el error de tipo I (rechazar inapropiadamente la hipótesis nula). El procedimiento de *bootstrap* multiescala consiste en generar pseudoréplicas de *bootstrap* como en el caso de *bootstrapping* no paramétrico, pero en este caso las pseudoréplicas son de longitud variable, de modo que ciertas pseudoréplicas serán de longitud mayor y otras de longitud menor que el alineamiento original. Tras contar el número de veces que una hipótesis filogenética es apoyada por las pseudoréplicas, se obtienen valores de *bootstrap* para cada conjunto de pseudoréplicas de una determinada longitud. El valor de probabilidad del test AU se calcula a partir de las diferencias en estos valores de *bootstrap* en función de las longitudes de las pseudoréplicas. Para conocer más acerca de los test de topologías consultar el

texto de Heiko Schmidt [40].

## 9.4. Reconstrucción de estados ancestrales

Además de permitirnos estudiar las relaciones de parentesco entre organismos (o genes) y comprender el proceso evolutivo que ha dado lugar a éstos, la inferencia filogenética también nos permite reconstruir el pasado. Si bien los fósiles pueden darnos información muy valiosa acerca de las características externas y el modo de vida de especies extintas, la información molecular de éstas es generalmente inexistente (aunque hemos sido capaces de secuenciar el genoma del Neandertal).

Existen diversos métodos que nos permiten inferir cuáles eran los *estados ancestrales* de un carácter, ya sea morfológico, molecular (por ejemplo, los aminoácidos) o incluso una variable cuantitativa [31]. Los métodos basados en parsimonia prefieren las reconstrucciones más sencillas (que requieran un número menor de cambios). Estos métodos tienen ciertas limitaciones, por lo que generalmente preferimos utilizar métodos probabilísticos que reconstruyen los estados ancestrales de una forma estadística. La MV permite hallar cuál era el estado ancestral más probable, mientras que los métodos Bayesianos calculan la probabilidad posterior para cada posible estado de dicho carácter. Los programas más robustos y prácticos para reconstruir estados ancestrales mediante métodos probabilísticos (MV e IB) son MrBayes, BayesTraits y PAML (Tabla 9.1). Existen guías y tutoriales que explican detalladamente cómo utilizar estos programas.

A continuación, presentaremos un ejemplo que sirve para ilustrar cómo resolver enigmas del pasado mediante la reconstrucción de los estados ancestrales de una proteína. Las opsinas son una proteínas que contienen como grupo prostético el 11-*cis*-retinal que se excita cuando capta un fotón. El cambio conformacional consecuente en la proteína produce una señal que se traduce a través de las proteínas G y finalmente llega a nuestro cerebro, confiriéndonos la capacidad de ver. Existen distintas opsinas, y cada una es sensible a una longitud de onda (color) característica. Es sabido que los mamíferos poseemos una excelente visión nocturna (los humanos no tanto, en realidad), y se ha propuesto que esta cualidad, junto a las de ser homeotermos y tener un buen abrigo (el pelo), habría permitido a nuestros ancestros llevar un estilo de vida nocturno durante el reinado de los dinosaurios en el Jurásico, que se creían animales principalmente diurnos. No obstante, un estudio reconstruyó la secuencia ancestral de las opsinas de los arcosaurios (grupo que engloba, entre otros, cocodrilos, dinosaurios y aves) y demostró, tras ensayar su actividad molecular *in vitro*, que la proteína reconstruida mediante métodos filogenéticos era completamente funcional (a pesar de su enorme antigüedad de 420 millones de años) y que además mostraba una elevada fotosensibilidad similar a la de las opsinas de los mamíferos actuales [6]. Este estudio puso en duda la hipótesis de que los dinosaurios eran diurnos, puesto que al menos potencialmente serían capaces de ver en la oscuridad. Éste y otros ejemplos de la utilidad de la reconstrucción de estados ancestrales pueden encontrarse en el artículo de Thornton [46].

## 9.5. Guía rápida de reconstrucción filogenética

En este capítulo hemos presentado los fundamentos de la reconstrucción filogenética con cierto detalle. A continuación ofrecemos una guía rápida (pero minuciosa) para reconstruir con fiabilidad un árbol filogenético.

1. *Selección de los datos.* La elección de qué secuencias y de qué especies hemos de incluir en el análisis depende del tipo de pregunta que queramos responder. Además, este paso suele ser iterativo, pues la obtención de un árbol nos puede sugerir cómo mejorar el muestreo de caracteres

o secuencias, por ejemplo, buscando un grupo externo (o *outgroup*) más adecuado o incluyendo o excluyendo ciertas secuencias.

2. El primer paso es la construcción de un *alineamiento múltiple de secuencias*, ya sean nucleótidos o aminoácidos. En el caso del alineamiento de nucleótidos correspondientes a secuencias que codifican proteínas, es recomendable utilizar herramientas como TranslatorX (Tabla 9.1) que determinan el alineamiento de nucleótidos en función del alineamiento de aminoácidos correspondiente. Esto lo hacemos así porque es más fácil alinear secuencias de aminoácidos que secuencias de nucleótidos, y además se respeta la integridad de los codones.
3. Seguidamente debemos asegurarnos de la *calidad de nuestro alineamiento*. Errores de secuenciación o de anotación incorrecta de los genes puede producir secuencias con inserciones o delecciones considerables. Además, es necesario eliminar las posiciones de homología posicional dudosa (regiones muy variables o con muchos gaps), para lo cual es conveniente utilizar herramientas como GBlocks o TrimAl (Tabla 9.1). De este modo reducimos el problema de la falsa homología, y hasta cierto punto el de saturación y homoplásia, mejorando el resultado de la inferencia filogenética.
4. Antes de proceder a inferir el árbol filogenético, normalmente debemos *determinar el modelo de evolución que se ajusta mejor a nuestros datos*. Los programas jModelTest y ProtTest implementan la selección de modelos de nucleótidos y aminoácidos (respectivamente), y ModelGenerator es capaz de trabajar con ambos tipos (Tabla 9.1). Estos programas ordenan los distintos modelos en función de nuestro criterio de selección (generalmente, AIC). A partir de esta lista, elegiremos aquel modelo que mejor se ajuste a nuestros datos (o sus particiones) y que al mismo tiempo sea implementado en el programa de reconstrucción que vayamos a utilizar (no todos los modelos están disponibles en los programas de reconstrucción filogenética).
5. *Los métodos de distancias* como BIONJ nos permiten obtener un *árbol de forma rápida*, que nos puede servir para hacernos una idea de las relaciones filogenéticas entre nuestras secuencias. Así, estos árboles preliminares nos permiten identificar rápidamente la existencia de ramas largas, una de las fuentes principales de artefactos filogenéticos (ver también punto 6). Para este tipo de reconstrucciones podemos servirnos de los programas SeaView o MEGA (Tabla 9.1), que integran en un entorno gráfico la gestión de alineamientos múltiples y la reconstrucción de árboles por BIONJ y otros métodos (aunque no todos).
6. *Para obtener un árbol más fiable, utilizaremos métodos probabilísticos* como MV o IB. Para la reconstrucción por MV recomendamos PhyML y RAxML (Tabla 9.1). El primero puede ejecutarse en el servidor web de PhyML, desde la consola o puede ser utilizado desde el programa SeaView. La IB generalmente demanda más esfuerzo de computación y es más lenta, pero se puede paralelizar. El programa más popular de IB es MrBayes, si bien existen otras alternativas como PhyloBayes, que implementan modelos evolutivos más complejos (Tabla 9.1). MrBayes utiliza un tipo de fichero llamado nexus, donde además del alineamiento, se incluye un bloque con las instrucciones para el análisis (modelo evolutivo, número de análisis paralelos, número de cadenas MCMC, número de generaciones, etc.). El servidor web MrBayes Web Form puede servir de ayuda para preparar estos ficheros (Tabla 9.1).
7. Para *visualizar los árboles* obtenidos podemos utilizar los visores integrados de los programas SeaView o MEGA, aunque son limitados en sus funciones. De forma alternativa, podemos recurrir al programa FigTree, que, además de ser más versátil, permite preparar figuras atractivas para su publicación (Tabla 9.1).
8. Frecuentemente, la inspección de los árboles nos sugiere que debemos refinar el conjunto de datos seleccionados (vuelta al punto “0”). Por ejemplo, cuando hay secuencias que han divergido mucho

más que las demás, sus ramas extremadamente largas pueden producir el fenómeno conocido como atracción de ramas largas, un tipo de artefacto filogenético bien caracterizado. Si fuera posible es recomendable evitar la inclusión de estas secuencias.

9. Normalmente el siguiente paso es el de *determinar cuán creíble es el árbol* que hemos obtenido. Si utilizamos MV, lo más habitual es realizar un *bootstrapping* no paramétrico, que está implementado en los programas de inferencia filogenética por MV. Podemos elegir 1000 réplicas, y los nodos que obtengan un soporte mayor de 70 % se considerarán fiables. Si la reconstrucción la hemos hecho con un método de IB, el soporte de los nodos se mide mediante probabilidades posteriores, que ya vienen implícitas en los resultados de la IB. Normalmente se consideran fiables aquellos nodos cuya probabilidad posterior es mayor de 0.95.
10. Finalmente, cuando queremos publicar un árbol filogenético, lo más recomendable es *combinar dos métodos probabilísticos*, MV e IB, y representar conjuntamente los resultados de ambos, indicando los valores de *bootstrap* y probabilidad posterior junto a cada nodo del árbol (o sobre los nodos de mayor interés).

## 9.6. Programas recomendados

Programa	Propósito	Modo de uso
ALTER	Conversión de formatos de alineamientos.	web
AWTY	Estudio de la convergencia de las cadenas MCMC en análisis de IB.	web
BayesTraits	MV e inferencia Bayesiana de estados ancestrales, coevolución y otros modelos evolutivos, tanto para caracteres discretos como continuos.	local
CONSEL	Contraste de hipótesis de topologías alternativas (test KH, SH, AU).	local
FigTree	Visualización y edición de árboles filogenéticos.	local
GARLI	Reconstrucción filogenética por MV.	
GBlocks	Limpieza de alineamientos.	web, local
HYPHY	Contraste de hipótesis evolutivas. Muy versátil.	local, web
JalView	Visor y editor de alineamientos múltiples.	web, local
jModelTest	Selección de modelos de evolución de nucleótidos.	local
MEGA	Múltiples posibilidades: reconstrucción filogenética, contrastes de hipótesis.	local
Mesquite	Evolución de caracteres, edición de árboles.	local
ModelGenerator	Selección de modelos de evolución.	local
MrBayes	Reconstrucción filogenética con métodos Bayesianos, reconstrucción de estados ancestrales, modelos evolutivos de aminoácidos, nucleótidos y codones. Posibilidad de combinar datos moleculares y morfológicos.	local
MrBayes Web Form	Generación de archivos en formato nexus que incluyen comandos específicos de MrBayes.	web
PAML	Contraste de hipótesis evolutivas: selección natural, reloj molecular, evolución adaptativa, simulaciones.	local
PartitionFinder	Selección de esquemas de partición.	local
PAUP	Múltiples y diversos métodos y test.	local
Phylip	Paquete de programas con múltiples funcionalidades.	web, local
PhyML	Reconstrucción filogenética por MV. Modelos de evolución de nucleótidos, aminoácidos y codones (codonPhyML).	web, local, SeaView
PhyloBayes	Reconstrucción filogenética con métodos Bayesianos. Implementa modelos evolutivos complejos que pueden acomodar fluctuaciones en los parámetros del modelo a lo largo del alineamiento.	local
ProtTest	Selección de modelos de evolución de proteínas.	web, local
ReadSeq	Conversión de formatos de alineamientos.	web, local
SeaView	Alineamiento, Neighbour-joining, Máxima parsimonia, Phyml, bootstrapping.	local

TranslatorX	Alineamiento de secuencias nucleotídicas guiado por el correspondiente alineamiento de proteínas.	web, local
Tree-Puzzle	Reconstrucción filogenética por MV utilizando el método quartet-puzzling para la búsqueda de árboles. Incluye modelo de reloj molecular.	local
TrimAl	Limpieza de alineamientos.	web, local

**Tabla 9.1:** Algunos programas útiles para la inferencia de árboles filogenéticos y el estudio evolución molecular. Ésta no pretende ser una lista exhaustiva, sino una recomendación personal de los autores de este capítulo.



## 9.7. Bibliografía

- [1] F. Abascal, R. Zardoya, and D. Posada. Prottest: selection of best-fit models of protein evolution. *Bioinformatics*, 21(9):2104–2105, 2005.
- [2] L. Bromham. Why do species vary in their rate of molecular evolution? *Biology Letters*, 5(3):401–404, 2009.
- [3] D. Bryant, N. Galtier, and M.-A. Poursat. *Likelihood calculation in molecular phylogenetics*, pages 33–58. Oxford University Press, Oxford, New York, 2005.
- [4] S. Capella-Gutierrez, J. M. Silla-Martinez, and T. Gabaldon. trimal: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics*, 25(15):1972–3, 2009.
- [5] J. Castresana. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Molecular Biology and Evolution*, 17(4):540–552, 2000.
- [6] B. S. W. Chang and M. J. Donoghue. Recreating ancestral proteins. *Trends in Ecology & Evolution*, 15(3):109–114, 2000.
- [7] M. D. Crisp and L. G. Cook. Do early branching lineages signify ancestral traits? *Trends in Ecology & Evolution*, 20(3):122–128, 2005.
- [8] C. Darwin. *On the origin of species by means of natural selection, or preservation of favoured races in the struggle for life*. John Murray, London, 1859.
- [9] A. J. Drummond, S. Y. W. Ho, M. J. Phillips, and A. Rambaut. Relaxed phylogenetics and dating with confidence. *PLoS Biology*, 4(5):e88, 2006.
- [10] A. W. F. Edwards and L. L. Cavalli-Sforza. *Reconstruction of evolutionary trees*, pages 67–76. Systematics Association, London, 1964.
- [11] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
- [12] J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39(4):783–791, 1985.
- [13] J. Felsenstein. Phylogenies and the comparative method. *The American Naturalist*, 125(1):1–15, 1985.
- [14] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, Inc., Sunderland, Massachusetts, 2004.
- [15] O. Gascuel. Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14(7):685–695, 1997.
- [16] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [17] D. M. Hillis and J. J. Bull. An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. *Systematic Biology*, 42(2):182–192, 1993.
- [18] M. Holder and P. O. Lewis. Phylogeny estimation: traditional and bayesian approaches. *Nature Reviews Genetics*, 4:275–284, 2003.
- [19] J. P. Huelsenbeck and K. A. Crandall. Phylogeny estimation and hypothesis testing using maximum likelihood. *Annual Review of Ecology and Systematics*, 28(1):437–466, 1997.
- [20] L. S. Jermiin, V. Jayaswal, F. Ababneh, and J. Robinson. *Phylogenetic model evaluation*, volume 452 of *Methods in Molecular Biology*, pages 331–364. Springer Verlag, Notowa, 2008.
- [21] T. H. Jukes and C. R. Cantor. *Evolution of protein molecules*, volume III, pages 21–132. Academic Press, New York, 1969.
- [22] T. Keane, C. Creevey, M. Pentony, T. Naughton, and J. McInerney. Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. *BMC Evolutionary Biology*, 6(1):29, 2006.
- [23] M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, 1983.

- [24] H. Kishino and M. Hasegawa. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from dna sequence data, and the branching order in hominoidea. *Journal of Molecular Evolution*, 29(2):170–179, 1989.
- [25] S. L. Kosakovsky Pond, A. F. Y. Poon, and W. Frost, Simon D. *Estimating selection pressures on alignments of coding sequences*, pages 419–490. Cambridge University Press, New York, second edition edition, 2009.
- [26] R. Lanfear, B. Calcott, S. Y. Ho, and S. Guindon. Partitionfinder: combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Mol Biol Evol*, 29(6):1695–701, 2012.
- [27] P. Lemey and D. Posada. *Molecular clock analysis*, pages 362–377. Cambridge University Press, New York, second edition edition, 2009.
- [28] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.
- [29] J. A. Nylander, J. C. Wilgenbusch, D. L. Warren, and D. L. Swofford. Awty (are we there yet?): A system for graphical exploration of mcmc convergence in bayesian phylogenetics. *Bioinformatics*, 24(4):581–583, 2008.
- [30] K. E. Omland, L. G. Cook, and M. D. Crisp. Tree thinking for all biology: the problem with reading phylogenies as ladders of progress. *BioEssays*, 30(9):854–867, 2008.
- [31] M. Pagel. Inferring the historical patterns of biological evolution. *Nature*, 401(6756):877–884, 1999.
- [32] D. Posada. jmodeltest: phylogenetic model averaging. *Molecular Biology and Evolution*, 25(7):1253–1256, 2008.
- [33] D. Posada. *Selecting models of evolution*, pages 345–361. Cambridge University Press, New York, second edition edition, 2009.
- [34] D. Posada and T. R. Buckley. Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests. *Syst Biol*, 53(5):793–808, 2004.
- [35] J. H. Reeves. Heterogeneity in the subsitution process of amino acid sites of proteins coded by mitochondrial dna. *Journal of Molecular Evolution*, 35:17–31, 1992.
- [36] F. Ronquist, P. van der Mark, and J. P. Huelsenbeck. *Bayesian phylogenetic analysis using MrBayes*, pages 210–266. Cambridge University Press, New York, second edition edition, 2009.
- [37] A. Rzhetsky and M. Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10(5):1073–1095, 1993.
- [38] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.
- [39] M. J. Sanderson. Nonparametric approach to estimating divergence times in the absence of rate constancy. *Molecular Biology and Evolution*, 14:1218–1231, 1997.
- [40] H. Schmidt. *Testing tree topologies*, pages 381–403. Cambridge University Press, New York, second edition edition, 2009.
- [41] H. Shimodaira. An approximately unbiased test of phylogenetic tree selection. *Systematic Biology*, 51:592–508, 2002.
- [42] H. Shimodaira and M. Hasegawa. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular Biology and Evolution*, 16:1114–1116, 1999.
- [43] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. *Phylogenetic inference*, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.
- [44] S. Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on Mathematics in the Life Sciences*, 17:57–86, 1986.
- [45] J. L. Thorne, H. Kishino, and I. S. Painter. Estimating the rate of evolution of the rate of molecular evolution. *Molecular Biology and Evolution*, 15(12):1647–1657, 1998.
- [46] J. W. Thornton. Resurrecting ancient genes: experimental analysis of extinct molecules. *Nature Review Genetics*, 5:366–375, 2004.
- [47] Z. Yang. *Computational molecular evolution*. Oxford Series in Ecology and Evolution. Oxford University Press, New York, 2006.

- [48] E. Zuckerkandl and L. Pauling. *Molecular disease, evolution, and genetic heterogeneity*, pages 189–225. Academic Press, New York, 1962.



## Parte III

# Biología estructural de proteínas



# Capítulo 10

## Alineamiento de estructura de proteínas

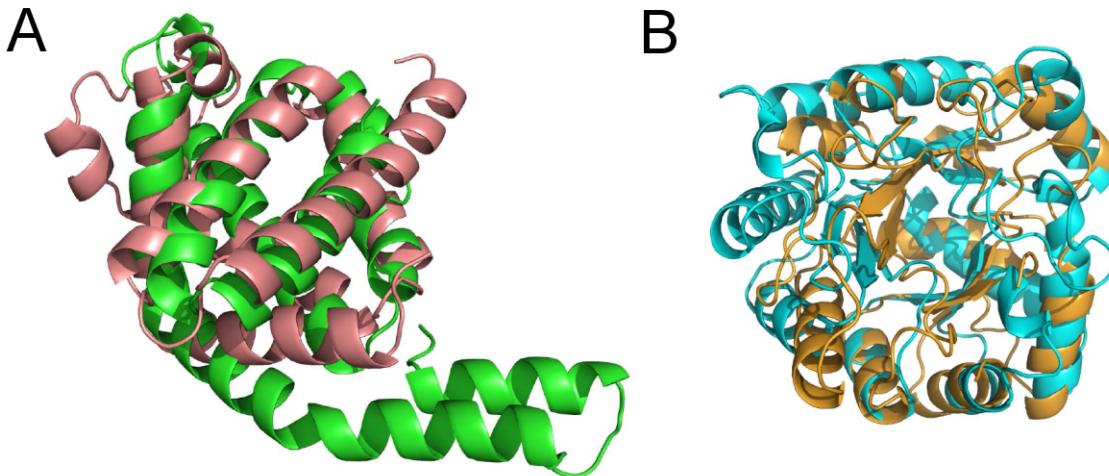
*Alberto Pascual-García*

### 10.1. Introducción

En el presente capítulo vamos a presentar una introducción a algunos métodos computacionales orientados a *comparar de manera objetiva estructuras de proteínas*. La motivación reside en que sabemos que la estructura de la proteína nos puede ayudar a inferir la función de la misma. La secuencia de aminoácidos determina el plegamiento nativo, y a su vez el plegamiento determinará los movimientos posibles de la proteína con la consecuente exposición de los distintos tipos de aminoácidos, y por extensión de sus posibilidades funcionales. Esta cuestión es tratada en mayor detalle en el Capítulo 11.

La observación de que *la divergencia estructural entre proteínas que comparten un ancestro común es proporcional al tiempo que hace que se separaron de dicho ancestro* [2], sugiere que la comparación estructural entre homólogos podría permitir inferir funciones desconocidas (ver Capítulo 12). Pero en realidad el escenario es bastante más complejo, pues podemos contemplar prácticamente todas las posibilidades en la terna secuencia, estructura y función. Es posible por ejemplo encontrar proteínas con la misma función en estructuras claramente distintas (y sin homología reconocible) pero que comparten una región funcional muy local de elevada similitud estructural. También encontramos conjuntos de proteínas homólogas, con elevada similitud estructural a nivel global y funciones distintas [1, 5, 6, 11]. En la Figura 10.1A se muestra el alineamiento entre dos proteínas sin homología reconocible, pero con una similitud global clara si bien también con diferencias en los motivos periféricos. Del mismo modo mostramos dos proteínas con aparente similitud global pero cuyos plegamientos son considerados distintos en las clasificaciones estructurales, e incluso con contradicciones entre ellas en la Figura 10.1B (ver Capítulo 12).

Pero esta introducción a la relación secuencia-estructura-función a la luz de la evolución, nos da una idea del escenario en el que tiene que moverse una persona interesada en desarrollar métodos computacionales para comparar estructuras. Encontrar similitudes estructurales entre dos estructuras cualesquiera implica que somos capaces de *identificar similitudes tanto locales como globales*, incluyendo posibles reordenamientos de sus elementos de estructura secundaria. Por ejemplo, si tuviéramos los motivos estructurales  $a - b - c$  a lo largo de la secuencia de la proteína  $A$  y los motivos  $b - a - d$  a lo largo de la proteína  $B$ , pero estuvieran espacialmente colocados de manera equivalente, deberíamos de ser



**Figura 10.1:** A. Alineamiento entre la subunidad alpha de la aloflicocianina (en verde, con PDB 1all y clasificada como “Globin-like” en SCOP) y la oxi-mioglobina (en marrón, con PDB 1a6m y clasificada como “Globin”). A pesar de no ser proteínas con homología reconocible sus plegamientos son globalmente similares. B. Lo mismo ocurre con la hidrolasa dependiente de metal (en azul, con PDB 1j6o y clasificada como “TIM barrel”) y la hipotética proteína YcdX de *Escherichia Coli* (con PDB 1m65 y clasificada como “7-stranded beta/alpha barrel”) [22].

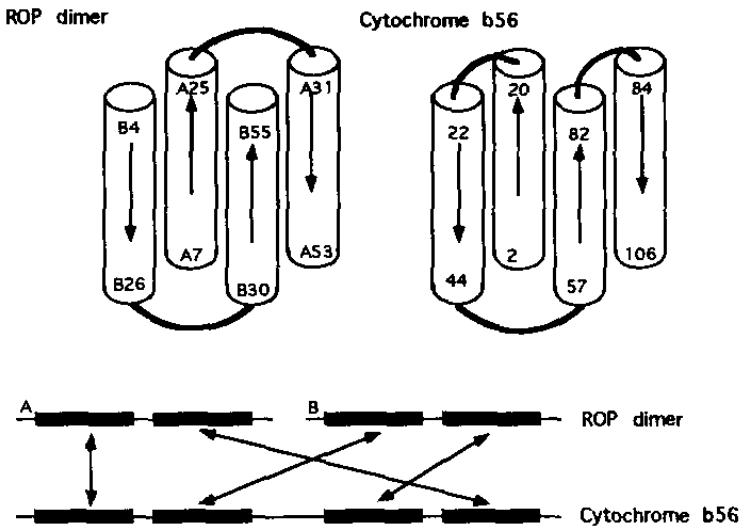
capaces de llevar a cabo el alineamiento (ver Figura 10.2).

Cabría preguntarse además hasta qué punto los algoritmos deberían desarrollarse siguiendo *criterios puramente geométricos* o deberían de considerar *información evolutiva*, como en el caso de los algoritmos de alineamiento de secuencias [25]. La mayoría de los algoritmos de comparación estructural consideran medidas geométricas ‘*ad-hoc*’. Considerar medidas estrictamente geométricas permiten generar medidas objetivas, pero exige una interpretación evolutiva y funcional posterior. Este hecho explica las *discrepancias entre las denominadas clasificaciones estructurales*, que en algunos casos utilizan información evolutiva y funcional, con respecto a los intentos de clasificación puramente geométricos. Esta cuestión es detallada también en la sección ??.

El número y variedad de *algoritmos de alineamiento estructural* se ha duplicado cada cinco años durante los últimos treinta años [4], y sin embargo no existe consenso sobre qué método y medidas es el mejor ya que no existe un ‘*golden standard*’ que sirva de referencia para mejorar dichos métodos, y se llegan a menudo a conclusiones contradictorias (ver por ejemplo las comparaciones que se encuentran en [20] y [25]). Un buen lugar para encontrar métodos disponibles *online* con una descripción sencilla de los mismos es la *Wikipedia*<sup>1</sup>, así como *software*<sup>2</sup>. Aquí hemos seleccionado algunos algoritmos que creemos son representativos de las aproximaciones generales más importantes a tener en cuenta, para hacernos una idea aproximada de toda la variedad que se esconde tras el problema de alineamiento estructural.

<sup>1</sup>Métodos de alineamiento estructural. [http://en.wikipedia.org/wiki/Structural\\_alignment](http://en.wikipedia.org/wiki/Structural_alignment)

<sup>2</sup>Software de alineamiento estructural. [http://en.wikipedia.org/wiki/Structural\\_alignment\\_software](http://en.wikipedia.org/wiki/Structural_alignment_software)



**Figura 10.2:** Representación esquemática del dímero ROP y el citocromo b56 [8]. Ambas proteínas comparten cuatro alpha hélices con conectividades distintas. En el caso del dímero, está formado por dos hebras idénticas (cadenas A y B) mientras que en el citocromo tenemos una única cadena. Necesitaremos por tanto una búsqueda local no secuencial para poder reconstruir el alineamiento. Figura reproducida con permiso del editor.

## 10.2. Descripción general del método

La primera división a tener en cuenta es entre métodos que consideran o no información evolutiva. Aquí vamos a tratar de métodos que consideran criterios puramente geométricos por ser claramente mayoritarios, si bien al final expondremos un ejemplo de medida que pretende incorporar información evolutiva, y remitimos al lector a algunos de los intentos de incorporar información evolutiva en las referencias [3, 13]. La mayoría de los *métodos geométricos* no hacen uso de la secuencia de las proteínas, sino que trabajan directamente con las *coordenadas de los átomos en el espacio* y en la mayoría de los casos únicamente con las coordenadas de los carbonos *alpha* ( $C_\alpha$ ). Por tanto definiremos una proteína genérica  $A$  mediante el conjunto de coordenadas  $x, y, z$  de cada uno de los carbonos *alpha* de sus residuos  $a_i$ , es decir:  $A = \{a_i\}_{i=1}^n = \{(x_{i_1}, y_{i_1}, z_{i_1})\}_{i=1}^n$ , donde  $n$  es el número de residuos. Esta descripción reducida de la proteína es lo que se conoce como *esqueleto* (del inglés *backbone*) de las proteínas que, aun siendo una simplificación importante es necesaria, ya que veremos que el problema sigue siendo de una elevada complejidad. Queremos entonces comparar el esqueleto de la proteína  $A$  con el de otra proteína  $B = \{b_i\}_{i=1}^m$  donde  $m$  es su número de residuos y, en el caso más general, es distinto de  $n$ . Nos gustaría obtener como salida del algoritmo un valor numérico indicando la similitud (o disimilitud) entre ambas estructuras.

Hay dos obstáculos que rápidamente salen a la luz en cuanto planteamos el problema. El primero es que las coordenadas de cada proteína están en *sistemas de referencia* distintos, por lo que se ha de encontrar una transformación entre los sistemas. El segundo es que las *longitudes de las proteínas* son distintas. Por tanto, nuestro problema consiste en encontrar una operación  $f : A \rightarrow B$  tal que para cada residuo  $a_i \in A$  encontramos una *correspondencia inyectiva* con un residuo  $b_k \in B$ . Además, el conjunto de todas las correspondencias encontradas debería ser tal que dicha correspondencia maximiza una función que consideraremos que mide la información estructural que ambas proteínas comparten. El problema consistente en decidir si existe una cierta correspondencia cuya mejor *transformación*

rígida aproxime las dos estructuras a una distancia prefijada entre los residuos en correspondencia. Su solución requiere la búsqueda y evaluación de todas las combinaciones posibles entre residuos, lo cuál es un problema no resoluble en tiempo polinomial, por lo que *podríamos diferenciar a los distintos algoritmos esencialmente por:* 1) en el modo en que reducen la búsqueda de combinaciones con una *aproximación heurística inicial* y 2) en el *tipo de función que evalúa la información estructural* que las proteínas comparten. Y, como es de esperar, la heurística utilizada va de la mano de la función que se pretende evaluar y viceversa, por lo que desde el punto de vista didáctico no es sencillo decidir qué exponer en primer lugar. Nosotros vamos a mostrar en primer lugar cómo reducir la combinatoria, que en la mayoría de los casos se resuelve mediante comparaciones locales entre ambas proteínas, y después veremos el modo en que la correspondencia se reconstruye optimizando una determinada función global.

Vamos a precisar de manera más formal estos pasos y a tratar algunos ejemplos. En los ejemplos vamos a presentar principalmente las aproximaciones seguidas por los *algoritmos de alineamiento Dali* [8], y *MAMMOTH* [20], si bien examinaremos otros algoritmos para explicar otras metodologías por su relevancia u originalidad. Elegimos estos algoritmos no solamente por ser ampliamente utilizados sino porque el primero se considera uno de los mejores algoritmos de alineamiento (es uno de los más antiguos y ha sido mejorado en sucesivas versiones) y el segundo es uno de los más rápidos, lo que lo hace apropiado para análisis masivos como el estudio del universo de estructuras de proteínas. Este *balance entre calidad del alineamiento y velocidad* queda patente en los distintos pasos que presentaremos, lo que nos resulta bastante ilustrativo.

### 10.3. Comparaciones locales

El primer paso para realizar un alineamiento de proteínas es la *descomposición de cada una de las proteínas problema en fragmentos* seguido de la comparación de todos los fragmentos contra todos de ambas proteínas. Los fragmentos son típicamente hexámeros o heptámeros, que es el tamaño mínimo para tener sensibilidad suficiente para *detectar similitudes locales*. La información que se utiliza típicamente es estrictamente estructural, si bien proliferan los métodos en los que alguna información adicional reduce el número de comparaciones a tener en cuenta. Por ejemplo, en Matras [9] se utiliza también información sobre la exposición al solvente del residuo.

Comencemos analizando cómo trabaja MAMMOTH. Este algoritmo considera todos los heptámeros en la proteína  $A \{(a_i, \dots, a_{i+6})\}_{i=1}^{n-6}$  y en la proteína  $B \{(b_k, \dots, b_{k+6})\}_{k=1}^{m-6}$  realizando una comparación de todos los heptámeros contra todos. Para realizar esta comparación considera una esfera de radio unidad y, para cada par de heptámeros a comparar, crea vectores unitarios centrados en el origen de dicha esfera en la dirección de  $C_\alpha \rightarrow C_{\alpha+1}$ . Como la separación entre dos carbonos  $\alpha$  consecutivos es aproximadamente fija (3,84Å) esta representación contiene la información que necesitamos sobre el esqueleto de la proteína. Tendremos por tanto 6 vectores unitarios  $u_i = \{\bar{u}_{j,j+1}\}_{j=i}^{j=i+6}$  para cada heptámero  $u_i$  de la proteína  $A$  y otros 6 vectores unitarios,  $v_k = \{\bar{v}_{l,l+1}\}_{k=l}^{k=l+6}$ , para cada heptámero de la proteína  $B$ , y podemos imaginar que introducimos los 12 vectores en la esfera de radio unidad centrados en el origen. A continuación, manteniendo fijos los vectores asociados a una de las proteínas, buscamos una única rotación solidaria de los vectores de la segunda (es decir, mantenemos fijos sus ángulos relativos), de modo que minimicemos la distancia cuadrática con respecto a los vectores de la primera proteína, que no es más que *minimizar la distancia entre todos los pares de vectores correspondientes para cada proteína*, donde los superíndices denotan las respectivas componentes de los vectores unitarios en las direcciones de  $x, y, z$ . Esta medida se denomina *URMS* del inglés *Unit-vector Root Mean Square* y tiene aplicaciones también como coordenada de reacción en el plegamiento de proteínas o para

monitorizar trayectorias en dinámica molecular [10].

$$URMS_{uv} = \sqrt{\sum_{i,k=1}^6 (u_{i,i+1}^x - v_{k,k+1}^x)^2 + (u_{i,i+1}^y - v_{k,k+1}^y)^2 + (u_{i,i+1}^z - v_{k,k+1}^z)^2} \quad (10.1)$$

Es interesante destacar de esta medida que se puede dar una estimación del valor esperado para dos polímeros construidos con direcciones al azar:  $URMS_R = \sqrt{2,0 - \frac{2,84}{\sqrt{L}}}$ , donde  $L$  es el número de vectores unitarios. Esto nos permite derivar una medida de similitud entre cada par de heptámeros  $u$  y  $v$  como:

$$S_{uv} = \frac{(URMS_R - URMS_{uv})}{URMS_R} \Delta(URMS_{uv}, URMS_R) \quad (10.2)$$

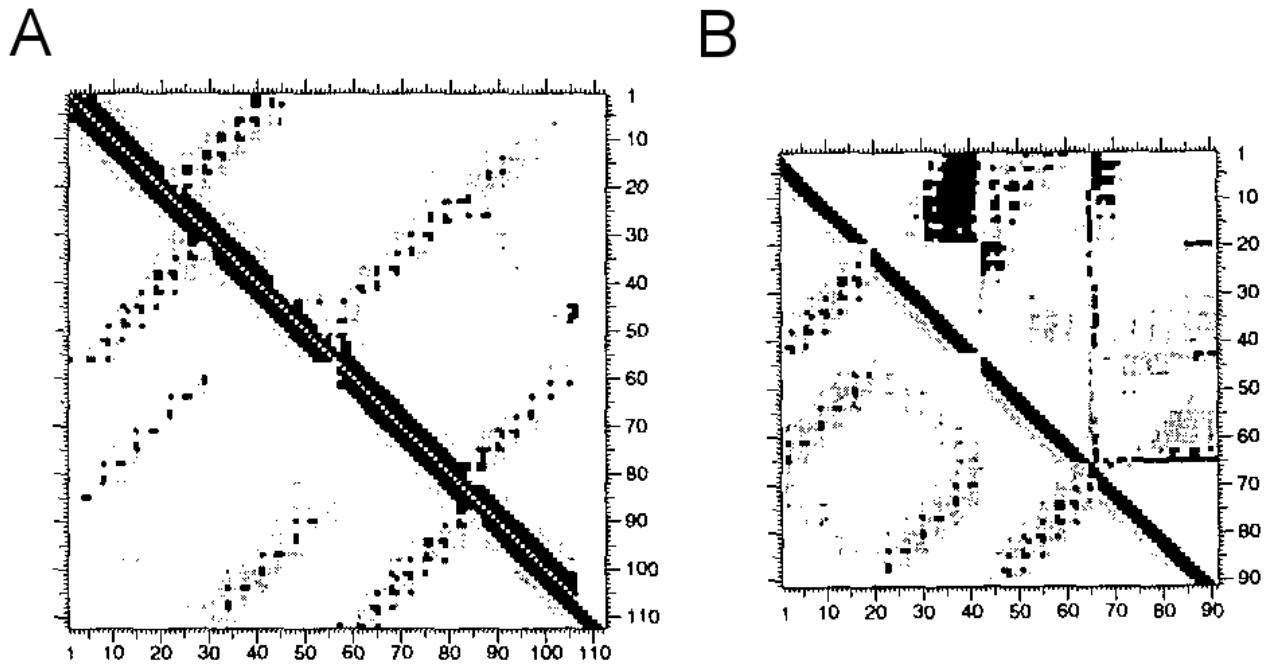
donde  $\Delta(URMS_{uv}, URMS_R) = 0$  si  $URMS_R < URMS_{uv}$  y tendrá un valor cualquiera, que determinará la escala de la medida de similitud, por ejemplo 10 en el caso de MAMMOTH, en caso contrario. De modo que si nuestra proteína tiene longitud  $n$ , este procedimiento nos permitirá determinar una matriz  $S_{uv}$  de tamaño  $n - 6 \times n - 6$  que será nuestra entrada para reconstruir el alineamiento global de la proteína, como veremos en el siguiente apartado.

En el caso del *programa de alineamiento Dali*, la aproximación que se realiza consiste en considerar también fragmentos de proteínas, pero primero calculando una *representación en dos dimensiones de cada proteína denominada matriz de distancias*. La matriz de distancias de una proteína genérica  $A$  que denotamos por  $d_{ij}^A$  es simplemente una matriz en la que, en cada celda, tenemos la distancia euclídea entre los dos residuos  $i$  y  $j$ . Esta representación “astuta” de la proteína, contiene toda la información estructural (salvando la quiralidad) y nos permite comparar fácilmente dos estructuras. Dali subdivide la matriz de distancias de la primera proteína  $d_{ij}^A$  en submatrices de distancias de tamaño  $6 \times 6$ :  $\{(a_i, \dots, a_i + 5; a_j, \dots, a_j + 5)\}$  (con  $i = 1, \dots, n - 13$  y  $j > i + 6, \dots, n$ ) y realiza la misma descomposición sobre la matriz de distancias  $d_{kl}^B$ :  $\{(b_k, \dots, b_k + 5; b_l, \dots, b_l + 5)\}$  en la proteína  $B$ . A continuación, *binariza* la matriz de distancias convirtiéndola en una *matriz de contactos*  $C_{ij}$  (ver Figura 10.3), es decir, la convierte en una matriz de unos y ceros en donde:

$$C_{ij} = \begin{cases} 1 & \text{si } d_{ij} \leq d_0 \\ 0 & \text{si } d_{ij} > d_0 \end{cases}$$

donde  $d_0 = 4\text{\AA}$  es la distancia mínima para considerar que dos residuos están en contacto. La ventaja de considerar matrices de contactos es que podemos rápidamente capturar la información estructural relevante ya que, como se puede ver en la sección de plegamiento de proteínas, los contactos que tienen lugar entre residuos alejados en la secuencia de la proteína son determinantes en el plegamiento de la misma.

La similitud entre dos submatrices de contactos se calculará simplemente como el número de contactos compartidos, que llamamos *solapamiento de contactos*. Como contrapartida, y dado que el tamaño de la matriz de distancias crece como el cuadrado de la longitud de la proteína ( $n^2$ ), el número de comparaciones a realizar crecerá como ( $n^2 m^2$ ), lo que hace esta aproximación computacionalmente mucho más intensa. Por ello Dali realiza una serie de filtros para proporcionar una lista de comparaciones reducida y no redundante (que no vamos a tratar) que será la entrada para el apartado que discutiremos a continuación en el que optimizamos una función global. Simplemente mencionar que esta reducción es tal que el orden de magnitud de la matriz es aproximadamente el mismo (para comparaciones de proteínas de cualquier longitud) que el que gestionaríamos con MAMMOTH para una comparación de dos proteínas de 200 residuos.



**Figura 10.3:** A. En la figura se representan las matrices de distancias de las proteínas del ejemplo mostrado en la Figura 10.2 en una única matriz (pues son simétricas y de longitud similar). Las distancias están discretizada en contactos a menos de 8 Å (negro), 12 Å (gris oscuro) y 16 Å (gris claro). El dímero ROP se representa en la parte triangular inferior y el citocromo b56 en la superior. B. Se representa en la triangular superior la matriz de diferencias de distancias entre ambas proteínas, manteniendo como referencia a ROP en la inferior tras eliminar la región no alineada. Para la matriz de diferencia de distancias el código de colores va del blanco (diferencia de menos de 1 Å), al negro (más de 4 Å). Figura reproducida con permiso del editor.

#### 10.4. Construcción del alineamiento: superposición rígida, flexible y una medida elástica.

Nuestro siguiente objetivo consiste en, a partir de la información que hemos obtenido de las comparaciones locales, obtener un alineamiento global óptimo desde el punto de vista estructural. En general la mayoría de los algoritmos construyen una primera matriz de similitudes locales como hemos mostrado en el caso de MAMMOTH, que permite realizar con Programación Dinámica un primer alineamiento global usando un método como el Needlman-Wunsch [18]. Discutiremos después el caso de Dali, que se diferencia en lo que vamos a exponer a continuación del resto.

En general, la actualización del alineamiento inicial utilizando información estructural global pasa por un proceso de superposición óptima. Para entender este proceso, consideramos el caso más simple de superposición, que es aquél que tiene lugar entre dos proteínas de la misma longitud con un elevado parecido entre ellas. En este caso no es necesario realizar alineamiento alguno pues existe una relación uno a uno entre los residuos  $a_i$  de la primera proteína  $A$ , y los residuos  $b_i$  de la segunda proteína que llamamos  $B$ . En este caso, llamamos *superposición óptima* a la posición relativa entre dos proteínas que maximiza (o minimiza) una medida global de similitud entre ambas cuando movemos una respecto a la otra únicamente mediante operaciones de cuerpo rígido, a saber, traslaciones y rotaciones. Si llamamos

$T$  a esta transformación de cuerpo rígido, una posible medida a optimizar sería la raíz cuadrada de la *desviación cuadrática media* (*RMSD* del inglés *root mean square deviation*) entre los residuos de ambas proteínas. Si llamamos  $D$  a su desviación cuadrática:

$$D = \sum_{i=1}^n \|a_i - T(b_i)\|^2$$

Definimos el valor del *RMSD* como:

$$RMSD = \sqrt{\frac{D}{n}} \quad (10.3)$$

La idea de la optimización del alineamiento inicial, consiste en aplicar este tipo de transformaciones de cuerpo rígido (superposiciones por tanto) a un primer conjunto de residuos que consideramos bien alineados, para progresivamente ir incorporando nuevos residuos a la vez que evaluamos la bondad de la superposición con una medida como el *RMSD*. Expongamos el problema más en detalle. Supongamos que nuestro primer alineamiento  $f : A \rightarrow B$  nos ha mapeado  $N_c$  residuos  $a_i \in A$  con el mismo número de residuos  $b_k \in B$ . Es conveniente para discutir las siguientes medidas y dado que tenemos identificados los pares y podemos ordenarlos, el unificar entonces la notación de nuestros subíndices y considerar el mismo subíndice para cada par alineado, es decir el par  $(a_i, b_k)$  pasaremos a llamarlo  $(a_i, b_i)$ . Pues bien, a continuación aplicaremos una transformación  $T$  de cuerpo rígido que maximice una función  $F(D_c)$  que contiene típicamente también un término de desviación cuadrático  $D_c$ :

$$D_c = \sum_{i \in N_c} \|a_i - T(b_i)\|^2$$

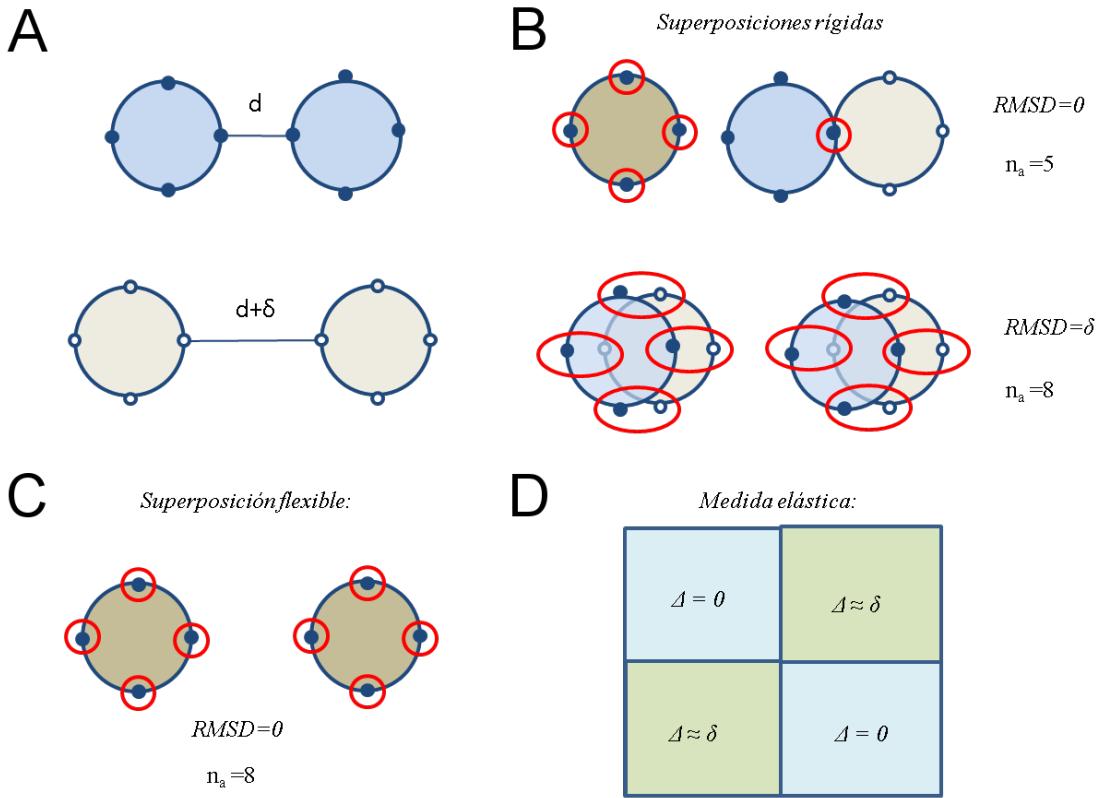
que ahora vemos está particularizado únicamente a los residuos alineados en común. Se procederá a modificar iterativamente el alineamiento inicial buscando que la función  $F(D_c)$  se minimice. Una forma sencilla de  $F(D_c)$  sería por ejemplo el denominado *RMS*( $N_c$ ):

$$RMS(N_c) = \sqrt{\frac{D_c}{N_c}} \quad (10.4)$$

Hay que observar aquí que, si quisiéramos optimizar esta medida, tendríamos que elegir entre, o bien restringir su valor a un umbral máximo e intentar maximizar el número de residuos alineados ( $N_c$ ), o bien intentar minimizar su valor restringiendo el valor de  $N_c$ . La primera opción es la que siguen algoritmos como MAMMOTH utilizando programación dinámica con la heurística de MaxSub [27], o los conocidos algoritmos CE [26] y ProSup [12]. La segunda opción es la elegida por ejemplo en el algoritmo LOVOalign [15]. La Figura 10.4 ejemplifica los distintos casos posibles.

Tenemos además que hacer notar en este punto que estas medidas tendrán una dependencia con la longitud de la proteína. Esto es debido a que la probabilidad de encontrar un mapeo entre dos pares de residuos al azar aumenta con la diferencia de la longitud de las proteínas que se está comparando, lo cual estará presente en el alineamiento que realizamos inicialmente y como también sucede en los alineamientos de secuencia. Y este hecho está aún presente en la superposición estructural óptima que estamos tratando, por lo que habrá que eliminar esta dependencia en la medida de salida que obtengamos como veremos en la sección siguiente. Sin embargo, se puede realizar una *normalización* en este paso para evitar la dependencia con la longitud, como se propone en el algoritmo TM-align [29], donde se incluye una normalización en la medida que proponen:

$$TMscore = \frac{1}{n_{min}} \sum_{i=1}^{N_c} \frac{1}{1 + (\|a_i - T(b_i)\| / g(n_{min}))^2}$$

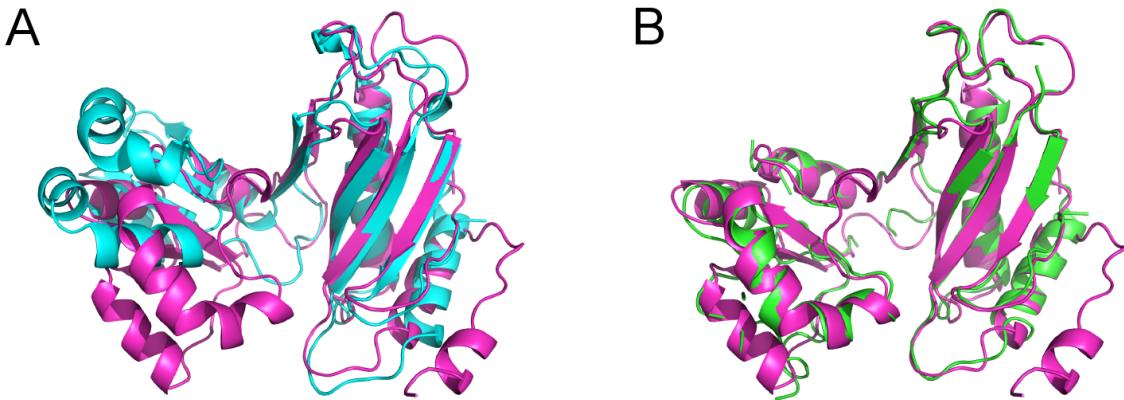


**Figura 10.4:** Superposición de dos estructuras ideales (A). En la superposición rígida (B) se maximiza o bien  $N_c$  fijando un  $RMSD$  mínimo o viceversa. Las elipses rojas muestran residuos superpuestos y vemos que, o bien no superponemos todos o lo conseguimos a costa de un  $RMSD$  poco satisfactorio. En la superposición flexible (C) la proteína se puede fragmentar (hecho que hay que controlar) y después superponer cada fragmento de manera independiente, lo que en este caso permite una superposición óptima. Por último, la base de las medidas elásticas residen en la matriz de diferencia de distancias, que es independiente del sistema de referencia, y permite también encontrar la superposición óptima (D). Figura adaptada de [4] con permiso del editor.

donde  $g(n_{min}) = 1,24\sqrt[3]{n_{min} - 15} - 1,8$  siendo  $n_{min}$  la longitud mínima de las proteínas comparadas y los distintos valores son parámetros optimizados experimentalmente. De este modo, se elimina la dependencia con la longitud mínima de las dos proteínas ya que la normalización está implícita en el proceso de superposición óptima. Existiría aún cierta dependencia con respecto a la longitud de cada una de ellas por separado, pero es importante únicamente para rangos de *scores* bajos [23]. Como veremos más adelante en la sección en la que tratamos las *normalizaciones*, una de las propiedades deseables de un *score* es que se aproxime a alguna distribución estadística para poder ser interpretado en términos de un *P – valor*, con el objeto de establecer un umbral de significatividad. En el caso de TM-score, valores por encima de 0,4 – 0,5 se consideran significativos [28], y hay una probabilidad alta de que se pueda considerar en el mismo *fold* por encima de 0,56 [23].

Hasta ahora, en el proceso de superposición óptima intentamos maximizar el número de residuos alineados considerando toda la proteína en los movimientos de cuerpo rígido. Pero entonces, si tuviéramos una proteína que posee dos conformaciones muy distintas, por ejemplo como consecuencia de que su función así lo requiere, tendríamos serias dificultades para encontrar un alineamiento y superposición

óptima, como se muestra en la Figura 10.4 esquemáticamente y un caso real en la Figura 10.5. Esto podría suceder por ejemplo en aquellas proteínas que tienen un cambio conformacional denominado de bisagra (del inglés hinge), en el que dos subdominios de la proteína se mueven relativamente uno respecto del otro como dos cuerpos rígidos con un eje (o bisagra) común.



**Figura 10.5:** Alineamiento estructural con superposición rígida (izquierda) y flexible (derecha), obtenida con ProtDeform [24], entre la proteína ribosomal L1 de la arquea metanógena *janaschii* (azul y verde respectivamente, con PDB '1cjs') y un mutante (PDB '1ad2') cuyo cambio fundamental es el reemplazamiento de la serina 179 por una cisteína. Este organismo es uno de los microbios que es capaz de vivir en condiciones más extremas, por lo que el estudio de su genoma desperta un gran interés. El alineamiento flexible permite encontrar una similitud estructural mucho mayor que el que obtenemos en el alineamiento rígido, observamos como contrapartida la introducción de gaps que se hace patente en las regiones con discontinuidades. Los alineamientos han sido gentilmente cedidos por Jairo Rocha.

Para encontrar una solución a este tipo de situaciones se han desarrollado los denominados *algoritmos de alineamiento flexible* como ProtDeform [24] o el alineamiento múltiple Matt [16]. La idea general de estos algoritmos es muy similar a la de los algoritmos que hemos presentado, pero su diferencia fundamental reside en el proceso de superposición óptima. En este proceso, se permite separar los movimientos de cuerpo rígido en regiones independientes, de modo que si existen cambios conformacionales podremos encontrar la superposición óptima como concatenación de superposiciones. La dificultad residirá entonces en determinar automáticamente en qué situación hay que dividir el problema, ya que una excesiva fragmentación podría permitir valores muy altos de la función de optimización considerada, pero sin sentido biológico. Por tanto, la determinación del tamaño de los dominios óptimo implica en sí mismo un proceso adicional de optimización que hace que estos algoritmos sean computacionalmente más pesados. Pero progresivamente vemos que las mejoras en estos algoritmos los convierten en alternativas muy atractivas respecto de otros algoritmos más establecidos. Por ejemplo, en el caso de ProtDeform se han realizado modificaciones orientadas a incrementar la velocidad y conseguir un *score* también independiente de la longitud, consiguiendo muy buenos resultados en relación al problema de la clasificación [23].

Terminamos esta sección comentando la metodología que usa Dali para construir el alineamiento global, ya que su aproximación es distinta a la del resto de los algoritmos. El hecho de que Dali considere matrices de contactos hace la comparación independiente del sistema de referencia, lo cual es una ventaja muy importante ya que evita realizar el proceso de superposición óptima mediante movimientos de cuerpo rígido. Para reconstruir el alineamiento en Dali se propone una medida en el que se comparan las distancias internas entre ambas proteínas, a través de las matrices de distancias respectivas.

Sean dos pares de residuos alineados entre las proteínas *A* y *B* : $(a_i, b_i)$  y  $(a_j, b_j)$ . Si consideramos las

distancias internas entre los residuos  $a_i$  y  $a_j$ , que recordamos hemos llamado  $d_{ij}^A$  y la correspondiente distancia entre los residuos  $b_i$  y  $b_j$ , que llamamos  $d_{ij}^B$ , definimos el *score*  $\phi$  entre las posiciones alineadas  $i$  y  $j$  como:

$$\phi(i, j) = \begin{cases} \left( \theta - \frac{|d_{ij}^A - d_{ij}^B|}{\bar{d}_{ij}} \right) w(\bar{d}_{ij}) & i \neq j \\ \theta & i = j \end{cases}$$

donde  $\bar{d}_{ij} = (d_{ij}^A + d_{ij}^B)/2$ , el parámetro  $\theta = 0,2$  nos indica que hay un 20 % de tolerancia en las distancias de contacto típicas que podemos encontrar, y  $w(\bar{d}_{ij}) = \exp(-\bar{d}_{ij}^2/\alpha^2)$  es una función que pesa la importancia de la diferencia que hemos encontrado en relación al tamaño típico de un dominio estructural, que es de  $\alpha = 20\text{\AA}$ . De este modo los contactos muy alejados en distancia se consideran, pero se infravaloran a través de esta función. A esta medida se le califica como *medida elástica*, porque permite ser tolerante con la acumulación de variaciones geométricas en la exploración de la reconstrucción global óptima.

Equipados con esta función, *Dali* reconstruye el alineamiento global explorando al azar las posibles combinaciones de submatrices con un algoritmo de Montecarlo con criterio de Metrópolis. En esta búsqueda, el criterio de aceptación se construye alrededor de una medida de similitud global que se define como:

$$S = \sum_{i=1}^{N_c} \sum_{j=1}^{N_c} \phi(i, j)$$

de modo que la probabilidad de aceptación será mayor si esta medida global crece. Se consideran distintos mecanismos de control en la reconstrucción del alineamiento tanto para que el mapeo entre residuos sea inyectivo (evitando redundancias) como para encontrar la reconstrucción óptima, ya que el carácter estocástico de la búsqueda en principio podría dar resultados diferentes cada vez que se alinean dos proteínas. Esto hace que *este algoritmo, aunque es probablemente el más preciso, sea también el más pesado*. Por este motivo se han buscado modificaciones que lo hacen más ligero creando además una base de datos para almacenar alineamientos muy útil para cálculos masivos, ya que el ejercicio se reduce a una búsqueda en la base datos en vez de el cálculo de nuevo de los alineamientos, el denominado DaliLite [7].

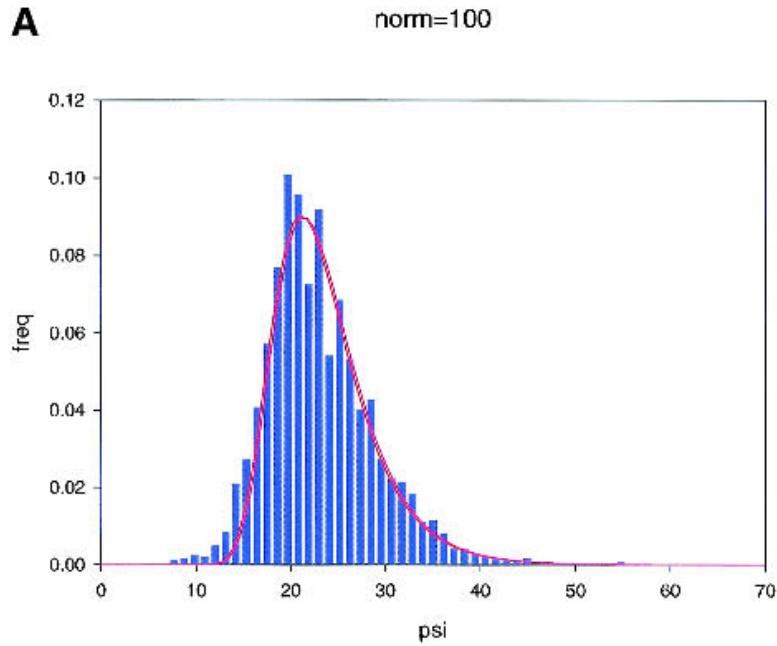
## 10.5. Medidas de similitud

### 10.5.1. Medidas crudas y normalizaciones

En general, la mayoría de los algoritmos de alineamiento tienen como salida algunas medidas típicas. Una de estas medidas suele ser el *porcentaje de residuos alineados*, conocido como *PSI* (del inglés *percentage of structural identity*) definido como  $PSI = N_c/n_{min}$ . También podemos encontrar el *RMS* ( $N_c$ ) definido en la Ecuación 10.4 o el *solapamiento de contactos* entre ambas proteínas, que denominamos  $q$ . Como hemos indicado anteriormente, estas medidas que podríamos denominar “crudas”, tienen la desventaja de contener una dependencia importante con la longitud de las proteínas alineadas, con lo que todos los algoritmos dan alguna medida adicional que evita este tipo de sesgos y permite además darnos una idea de la significatividad de la medida. Ponemos a continuación algunos ejemplos de estas medidas.

En algoritmos como MAMMOTH, existe una dependencia clara de estas medidas con respecto a la longitud. Por ejemplo, si consideramos todos los pares de proteínas que tienen un valor de  $n_{min}$  determinado y pintamos el *histograma de su PSI*, observaremos que en todos los casos se aproxima a

una *distribución denominada de valores extremos* (Sección 3.3.3) para comparaciones entre pares de proteínas que no tienen homología reconocible, lo que sugiere un patrón similar para comparaciones al azar. Encontraríamos un comportamiento similar para otras medidas como el solapamiento de contactos. Podemos entonces normalizar nuestra medida si encontramos los parámetros de la distribución para cada valor de  $n_{min}$ . Supongamos que, en vez de seguir una distribución de valores extremos, sigue una distribución gaussiana. Esta suposición no es en absoluto cierta porque son distribuciones bien diferenciadas. Pero simplificará la explicación y la diferencia en los valores que obtendríamos tras normalizar no es muy importante desde el punto de vista práctico si los valores que consideramos de  $PSI$  son suficientemente altos, es decir, en el extremo de la cola superior de la distribución de datos observados.



**Figura 10.6:** Distribución del  $PSI$  para pares de proteínas con  $n_{min} = 100$  obtenido de [20]. Los datos se ajustan bien a una distribución de valores extremos (curva roja) para todos los valores de  $n_{min}$ . Al igual que una distribución gaussiana, esta distribución tiene dos parámetros que podremos obtener como función de  $n_{min}$  y que llamamos  $\overline{PSI}(n_{min})$  y  $\sigma_{PSI}(n_{min})$ . En nuestro caso, estas funciones se ajustan bien a funciones analíticas (en particular a leyes de potencias) lo que permite el cálculo directo de una medida como un  $Zscore$  para cualquier par. Figura representada con permiso del editor.

Con esta consideración procederíamos calculando para cada valor de  $n_{min}$  la media  $\overline{PSI}(n_{min})$  y desviación estándar  $\sigma_{PSI}(n_{min})$ , que vemos son funciones de  $n_{min}$ . En la Figura 10.6 podemos observar la distribución de  $PSI$  para pares no relacionados con  $n_{min} = 100$ . Podemos por tanto proponer una medida que evalúe la significatividad de un determinado valor del  $PSI$ , conocido el valor de  $n_{min}$  del par analizado, como un  $Zscore$  (Sección 3.3.3):

$$Zscore = \frac{PSI - \overline{PSI}(n_{min})}{\sigma_{PSI}(n_{min})} \quad (10.5)$$

Y sabemos que un valor de  $Zscore > 2$  se puede considerar como significativo. Sin embargo tenemos que notar aquí que, cuando comparamos dos proteínas, no sabemos si están o no relacionadas a priori evolutivamente. Por tanto, aplicar este procedimiento podría penalizar la significatividad de aquellas proteínas que, estando relacionadas tienen un valor de  $\sigma_{PSI}(n_{min})$  grande, ya que esta normalización

la vamos a calcular a ciegas, es decir, sin hacer suposiciones previas sobre si están o no relacionadas evolutivamente.

Un modo de evitar esta penalización es realizar un estudio para ver si existe cierto valor  $\epsilon$  de la medida de similitud que estamos utilizando (sea el porcentaje de identidad estructural, el solapamiento de contactos u otro similar) por encima del cual sabemos que existe una relación evolutiva. Si encontramos este valor, que dependerá también de la longitud,  $\epsilon = \epsilon(n_{min})$ , podemos evitar la normalización sugerida en la Ecuación 10.5. ¿Qué expresión podemos seguir en este caso y qué normalización habrá que realizar? Podemos proponer aquí una hipótesis evolutiva.

### 10.5.2. Una medida con motivación evolutiva

Sabemos que la probabilidad  $P\{A_i = B_i\}$  de que dos residuos de las proteínas  $A$  y  $B$ , que son homólogas, sean iguales en secuencia en la posición  $i$  decrece con el tiempo con un ritmo  $1/\tau$  de manera exponencial:  $\exp(-t/\tau)$ . También podemos estimar la probabilidad condicionada de que, si ha habido al menos un cambio en una de las dos secuencias, suceso que tiene probabilidad  $(1 - \exp(-t/\tau))$ , las dos secuencias tengan el mismo residuo. La probabilidad  $p$  del suceso “tener el mismo residuo” se puede estimar a partir de las frecuencias relativas  $f$  de cada tipo de residuo  $a$  como  $p = \sum_a f(a)^2$ . Por tanto la probabilidad condicionada será:  $p(1 - \exp(-t/\tau))$  y la probabilidad final que buscamos será:

$$P\{A_i = B_i\} = e^{-t/\tau} + p(1 - e^{-t/\tau}) \quad (10.6)$$

Si asumimos que cada posición evoluciona de manera independiente, lo cual no es cierto pero es una asunción casi inevitable, podemos relacionar esta probabilidad con la identidad en secuencia (porcentaje de identidad entre dos secuencias)  $P\{A_i = B_i\} \approx SI$ , de modo que en un instante de tiempo  $t$  la *divergencia evolutiva entre las dos secuencias* se puede calcular, despejando la Ecuación 10.6 como:

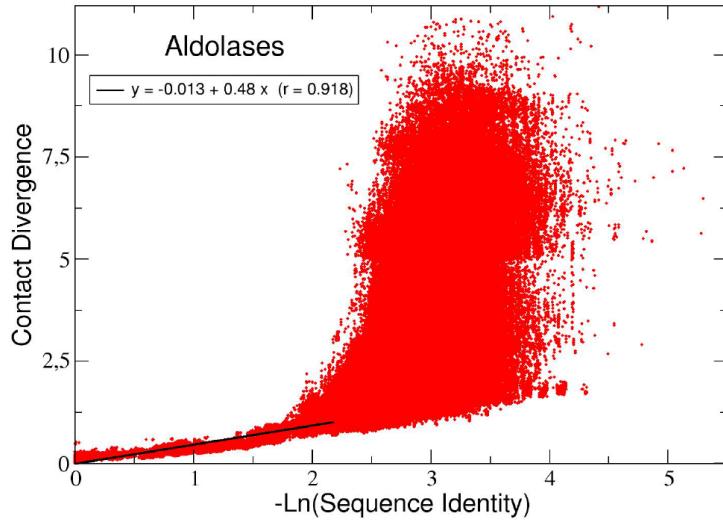
$$\text{Divergencia\_secuencia} = t/\tau = -\ln\left(\frac{SI - p}{1 - p}\right) \quad (10.7)$$

Esta medida, cuando  $SI \gg p$  coincide con la fórmula que se obtendría para estimar la divergencia evolutiva con un proceso de Poisson [19], luego estamos considerando un proceso de Poisson corregido. Chothia y Lesk [2] observaron que la divergencia en estructura es también proporcional al tiempo que ha pasado desde que ambas estructuras comenzaron a diverger (Capítulo 12). Así que la idea para trabajar con una similitud estructural es que, si la medida que utilizamos es mayor que el umbral  $\epsilon(n_{min})$  que determinemos y dada la relación entre la divergencia en secuencia y en estructura encontrada por Chothia y Lesk, es legítimo proponer una medida similar a la divergencia en secuencia para la *divergencia en estructura* [21]:

$$\text{Divergencia\_estructura} = -\ln\left(\frac{q - q_\infty(n_{min})}{1 - q_\infty(n_{min})}\right) \quad (10.8)$$

Donde  $q$  es el solapamiento de contactos y  $q_\infty(n_{min})$  es el análogo de  $p$  en el análisis que hemos hecho de secuencia, y hemos colocado el subíndice  $\infty$  para enfatizar que la función proporciona valores del solapamiento de contactos en el límite de tiempos evolutivos muy largos, que también es una función de  $n_{min}$ . Por tanto, si nuestra medida de similitud  $q$  cumple que  $q > \epsilon(n_{min})$  utilizaremos la Ecuación 10.8, mientras que en caso contrario utilizaremos la Ecuación 10.5, que sabemos que funciona bien para pares no relacionados.

La parte positiva de esta medida es que, a pesar de que hay varios parámetros y asunciones a tener en cuenta, nos permite cuantificar la relación entre la divergencia en secuencia y en estructura que, si bien



**Figura 10.7:** Relación entre la divergencia en secuencia y estructura para la superfamilia de las aldolases. La definición de la divergencia de contactos nos permite estimar el ritmo de divergencia entre ambas, para lo cual se realiza un ajuste en la región compatible con una ordenada en el origen significativamente cercana a cero. Más detalles en [21].

sus valores absolutos hay que tomarlos con precaución, sus valores relativos entre familias de proteínas distintas nos pueden permitir hacer estudios evolutivos [21].

Esta medida tiene un comportamiento similar al denominado TM-score que vimos en la sección en la que discutíamos el proceso de superposición típico de los algoritmos rígidos (ver Figura 10.4). Sin embargo no se construye con ninguna hipótesis evolutiva y su interpretación es menos directa.

## 10.6. Alineamiento múltiple

Nuestro siguiente objetivo consiste en alinear estructuralmente un conjunto arbitrario de proteínas. Queremos encontrar el alineamiento común que contenga el máximo número de residuos con la menor distancia espacial posible entre sus residuos, medida con alguna expresión como el *RMS* mostrado en la Ecuación 10.4. De este modo, si tenemos ya no un par sino todo un conjunto de proteínas homólogas, el encontrar el mayor número de posiciones espaciales conservadas implica que podemos calcular la variabilidad en secuencia que se observa en dichas posiciones, lo cual nos permite establecer hipótesis sobre la evolución y funciones de los miembros del conjunto. Por tanto sus aplicaciones son numerosas, desde la mejora de los mismos alineamientos en secuencia (Capítulo 8) a la mejora de la clasificación de estructura de proteínas (Subsección 12.3.1), puntos que giran a su vez alrededor de los métodos de predicción de estructura.

Los *algoritmos de alineamiento múltiple* utilizan aproximaciones variadas y no vamos a repasar aquí las diferencias entre los distintos métodos con el mismo detalle que hemos realizado el análisis de los métodos de alineamiento de pares. Vamos a aprovechar la descripción que ya hemos realizado del alineamiento de pares MAMMOTH para ver cómo se generaliza el algoritmo de manera natural a su versión múltiple: MAMMOTH-mult [14]. El lector que haya seguido la explicación de la sección de alineamiento de pares podrá, a la vista de la generalización de este algoritmo, seguir fácilmente otras aproximaciones como el ya mencionado Matt [16], o cómo se determinan los clusters estructurales en

la base de datos HOMSTRAD [17].

Consideremos un conjunto  $P$  de proteínas que queremos alinear. Un comienzo razonable podría ser el mismo que realiza la versión de pares de MAMMOTH, a saber, se divide cada proteína en heptámeros y se comparan los heptámeros todos contra todos de cada par de proteínas del grupo. Hasta aquí, realizaríamos el mismo análisis local que en dicha versión si la corriéramos para cada par de proteínas. Pero recordemos que tenemos que encontrar después el alineamiento óptimo para todas las proteínas, optimizando globalmente el mismo con operaciones de cuerpo rígido. Y hay muchas combinaciones posibles para ir seleccionando proteínas y optimizando dicha superposición, lo que nos lleva a un problema computacionalmente demasiado pesado si quisieramos considerar todos los modos en que seleccionamos subconjuntos de proteínas. Así que MAMMOTH-mult realiza un par de pasos previos a la construcción del alineamiento múltiple.

### 10.6.1. Primeros pasos

El *primer paso* consiste efectivamente en realizar un *alineamiento de pares de todas las proteínas contra todas* con la versión estándar de pares. El *segundo paso* consistirá en, a partir de la medida de similitud que obtenemos del algoritmo bien normalizada, *utilizar un algoritmo aglomerativo (average linkage)* (Subsección 12.3.1)) *para relacionar jerárquicamente las estructuras en un árbol*. De este modo tenemos una aproximación razonable al ordenamiento ideal en el que tendríamos que ir incorporando estructuras para la reconstrucción del alineamiento múltiple. Obviamente observamos ya un par de elecciones arbitrarias en este primer paso (el algoritmo de alineamiento de pares con una medida concreta asociada, y el algoritmo aglomerativo) y otras opciones podrían determinar resultados distintos. Por este motivo en los algoritmos es importante hablar de la robustez del método, es decir, cuánto dependen sus resultados de las elecciones que se realizan. Vamos a asumir que el método es robusto ante estas elecciones y veamos en qué consiste específicamente el alineamiento múltiple.

### 10.6.2. Construcción del alineamiento

Una vez que tenemos el árbol resultado del proceso de aglomeración, lo recorremos desde las hojas hasta la raíz, parandonos en cada nudo en donde dos ramas se separan. Si llamamos  $A$  y  $B$  a las ramas (ya no a las proteínas), que ahora contendrán  $P_A$  y  $P_B$  proteínas ( $P_A + P_B \leq P$ ), se van a dividir los alineamientos ya definidos en cada rama en heptámeros, y se van a utilizar los *URMS* calculados en la comparación previa de pares, pero donde ahora queremos pesar la contribución de cada proteína a cada heptámero. Siguiendo la notación de la Ecuación 10.1, de los heptámeros  $u$  y  $v$  que pasan por las posiciones  $i \in A$  y  $k \in B$  asignamos a cada par de posiciones el valor *URMS* más pequeño, es decir  $URMS_{ik} = \min(URMS_{uv})$ ;  $i \subset u$ ,  $k \subset v$ . Y entonces calculamos el promedio para cada par de posiciones como:

$$URMS_{ik}^{AB} = \frac{1}{P_A} \frac{1}{P_B} \sum_{a=1}^{P_A} \sum_{b=1}^{P_B} URMS_{ikab} \quad (10.9)$$

cuyos valores se pueden normalizar del mismo modo que en la Ecuación 10.2 para obtener una matriz de similitudes. Los siguientes pasos, al igual que MAMMOTH de pares, consiste en realizar un alineamiento local con Needleman-Wunsch [18] para conseguir las correspondientes asignaciones entre residuos y se realiza una superposición tridimensional utilizando también MaxSub [27].

La novedad consiste en que, a partir de la superposición, queremos obtener una nueva *matriz de similitud* al estilo de la construida en la Ecuación 10.9, pero ahora teniendo en cuenta las posiciones

espaciales obtenidas mediante la superposición:

$$S_{ik}^{AB} = \frac{1}{P_A} \frac{1}{P_B} \sum_{a=1}^{P_A} \sum_{b=1}^{P_B} \left( w_e U RMS S_{ikab} + w_d e^{-\alpha d_{ikab}^2} \right) \quad (10.10)$$

donde  $d_{ikab}$  es la distancia euclídea entre las posiciones  $i$  y  $k$  de las proteínas  $a$  y  $b$ , y se introducen los parámetros  $\alpha$  que controla el decaimiento en función de la distancia.  $w_e$  junto con  $w_d$  controlan el peso relativo que daremos a la componente relacionada con el esqueleto (*backbone*) y la distancia cartesiana, respectivamente. Dichos parámetros se optimizan utilizando alineamientos estructurales curados manualmente, utilizando un algoritmo de Monte Carlo por ejemplo.

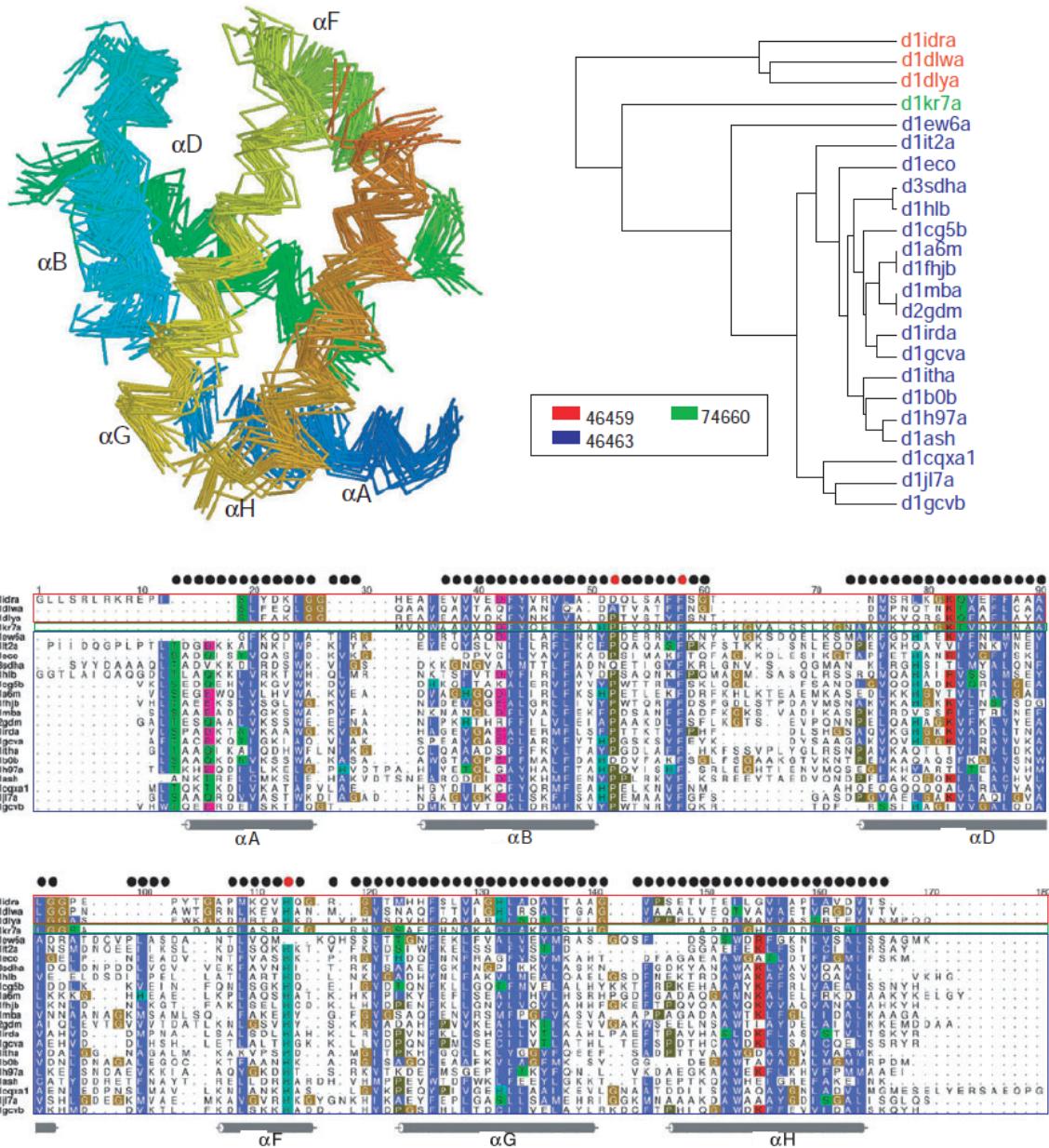
El último paso consiste en forzar al conjunto de residuos alineados para todas las proteínas, el denominado *core* del alineamiento compuesto por  $N_c$  residuos, a tener una *superposición óptima* para todas las proteínas. Para ello se van seleccionando las proteínas que comprenden el alineamiento de manera ordenada y, para cada una de ellas, se realizan movimientos de cuerpo rígido manteniendo a todas las demás fijas hasta encontrar un valor de  $RMS(N_c)$  mínimo, calculado según la Ecuación 10.4. Este proceso se itera sistemáticamente hasta que la siguiente función de error converge:

$$\varepsilon_{core} = \sum_{m=1}^{N_c} \sum_{a \neq b}^{P_A+P_B} d_{mab}^2$$

donde nuevamente  $d_{mab}$  es la distancia euclídea para la posición alineada  $m$  entre las proteínas  $a$  y  $b$ .

Una vez hemos conseguido que converja el error, iremos recorriendo el árbol hasta el siguiente nudo, hasta que lleguemos a la raíz en donde  $P_A + P_B = P$ . Como salida obtendremos todos los cálculos realizados en los primeros pasos por el algoritmo de comparación de pares y algunas medidas propias del alineamiento múltiple. Algunas de estas medidas son el *tamaño del core* denominado *estricto*, que es aquél con un 100 % de conservación de sus residuos en secuencia que están además alineados (cualesquiera de las parejas de proteínas consideradas) a menos de 4 Å. También se define un *core* denominado *laxo*, que es aquél con un 66 % de conservación y donde las proteínas están alineados a menos de 3 Å respecto del promedio del alineamiento en cada una de las posiciones. Sobre estas definiciones se pueden realizar medidas generales como el *RMS* promedio de cada tipo de *core* o su desviación típica.

En la Figura 10.8 presentamos el alineamiento múltiple de 23 proteínas de la familia de las globinas según están clasificadas en la clasificación estructural manual de SCOP (ver sección de evolución de estructura de proteínas). Esta superfamilia está subdividida en familias correspondientes a las globinas canónicas de unión al grupo hemo (código 46463), las globinas truncadas de protozo/bacteria (46459) y las hemoglobinas neuronales (74660). El alineamiento estructural y el dendograma que se puede reconstruir del protocolo de alineamiento, muestra una clara separación de los tres grupos. Se representa además el alineamiento en secuencia consecuencia del alineamiento estructural. De este último alineamiento es interesante observar la conservación en la posición estructural de la histidina situada en la posición 113. Esta histidina está situada en el sitio activo, y establece la unión con el átomo de hierro del grupo hemo.



**Figura 10.8:** Alineamiento estructural de la superfamilia de las globinas (arriba izquierda) obtenido con MAMMOTH múltiple [14]. Arriba a la derecha se muestra el dendograma asociado, y abajo el alineamiento en secuencia. Más detalles en el texto. Figura reutilizada con permiso del editor.

## 10.7. Discusión

En la presente sección hemos abordado el problema del alineamiento de estructura de proteínas. Habiendo en abstracto, *el problema del alineamiento se podría interpretar como la búsqueda de un sistema de referencia común a un conjunto de entidades, con el objeto de identificar la información que tienen en común*. En nuestro caso, esta información está codificada en la estructura de las proteínas, y su identificación hemos visto que nos abre numerosas puertas. Los distintos eventos evolutivos que conocemos, son los mecanismos mediante los cuales está información ha fluido en el universo de estructura de proteínas que podemos observar actualmente. Por otra parte, el modo en el que la información es capaz de transmitirse y modificarse a lo largo del proceso evolutivo está condicionado a los requerimientos biológicos derivados de la funcionalidad de las proteínas. La posibilidad de que un cambio evolutivo se fije en una proteína, vendrá determinado por su viabilidad termodinámica y cinética, que son condiciones necesarias para que la proteína realice su función. Esto nos llevará a interesarnos por la física del plegamiento de proteínas. Pero no son condiciones suficientes, porque pueden existir aminoácidos cuya localización concreta es esencial para realizar la función, como vimos con la histidina en el ejemplo de las globinas, lo que nos lleva a interesarnos por las particularidades químicas de las mismas.

*El alineamiento estructural es*, por tanto, una herramienta muy potente en tanto en cuanto nos permite abrir *una ventana en la cual asomarse a todo el proceso evolutivo*. Qué eventos son dominantes, qué estructuras son las más relevantes o qué motivos locales determinan la función, son estudios que pueden fundamentarse a partir de los análisis obtenidos mediante el alineamiento estructural junto con la información proveniente de la secuencia y de las funciones conocidas. Con todas estas posibilidades en mente, invitamos al lector a afrontar las secciones de plegamiento de proteínas (Capítulo 11), evolución de estructura de proteínas (Capítulo 12) y modelización. El recorrido que proponemos entrará en la física con los modelos de plegamiento para después asomarse al pasado en la sección de evolución. Allí llegaremos incluso a conjeturar sobre los eventos dominantes en instantes de tiempo incluso anteriores a los dominios estructurales tal y como los conocemos, con una aproximación fuertemente basada en la comparación mediante alineamientos estructurales. Todo este recorrido nos hará llegar finalmente a la modelización, que será nuestro objetivo desde el punto de vista práctico más potente, pues su aplicación permite ampliar nuestras posibilidades de comprender la función de proteínas cuyo rol es desconocido, con la potencialidad que conlleva el comprender dicha función desde el punto de vista de las aplicaciones biomédicas.

## Agradecimientos

El autor agradece especialmente la revisión crítica del manuscrito a Jairo Rocha. Este trabajo ha sido posible gracias a la financiación del Ministerio de Economía y Competitividad a través de una beca FPI (BES-2009-013072). La investigación en el CBMSO es apoyada por la Fundación Ramón Areces.



## 10.8. Bibliografía

- [1] P. Bork, C. Sander, and A. Valencia. Convergent evolution of similar enzymatic function on different protein folds: The hexokinase, ribokinase, and galactokinase families of sugar kinases. *Protein Science*, 2(1):31–40, 1993.
- [2] C. Chothia and A. M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO Journal*, 5(4):823–826, Apr. 1986. PMID: 3709526 PMCID: PMC1166865.
- [3] G. Csaba, F. Birzele, and R. Zimmer. Protein structure alignment considering phenotypic plasticity. *Bioinformatics*, 24(16):i98–i104, Aug. 2008.
- [4] H. Hasegawa and L. Holm. Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, 19(3):341–348, June 2009.
- [5] H. Hegyi and M. Gerstein. Annotation transfer for genomics: Measuring functional divergence in multi-domain proteins. *Genome Research*, 11(10):1632–1640, Oct. 2001.
- [6] F. G. Hoffmann, J. C. Opazo, and J. F. Storz. Gene cooption and convergent evolution of oxygen transport hemoglobins in jawed and jawless vertebrates. *Proceedings of the National Academy of Sciences of the United States of America*, 107(32):14274–14279, Aug. 2010. PMID: 20660759.
- [7] L. Holm and J. Park. DALI Lite workbench for protein structure comparison. *Bioinformatics*, 16(6):566–567, June 2000.
- [8] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1):123–138, Sept. 1993.
- [9] T. Kawabata. MATRAS: a program for protein 3D structure comparison. *Nucleic Acids Research*, 31(13):3367–3369, July 2003.
- [10] K. Kedem, L. P. Chew, and R. Elber. Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories. *Proteins: Structure, Function, and Bioinformatics*, 37(4):554–564, 1999.
- [11] L. N. Kinch and N. V. Grishin. Evolution of protein structures and functions. *Current Opinion in Structural Biology*, 12(3):400–408, June 2002.
- [12] P. Lackner, W. A. Koppensteiner, M. J. Sippl, and F. S. Domingues. ProSup: a refined tool for protein structure alignment. *Protein Engineering*, 13(11):745–752, Nov. 2000.
- [13] X. Liu, Y.-P. Zhao, and W.-M. Zheng. CLEMAPS: multiple alignment of protein structures based on conformational letters. *Proteins: Structure, Function, and Bioinformatics*, 71(2):728–736, 2008.
- [14] D. Lupyan, A. Leo-Macias, and Á. R. Ortiz. A new progressive-iterative algorithm for multiple structure alignment. *Bioinformatics*, 21(15):3255–3263, Aug. 2005.
- [15] L. Martínez, R. Andreani, and J. M. Martínez. Convergent algorithms for protein structural alignment. *BMC Bioinformatics*, 8(1):306, Aug. 2007.
- [16] M. Menke, B. Berger, and L. Cowen. Matt: Local flexibility aids protein multiple structure alignment. *PLoS Comput Biol*, 4(1):e10, Jan. 2008.
- [17] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Science*, 7(11):2469–2471, 1998.
- [18] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, Mar. 1970.
- [19] M. Nei and S. Kumar. *Molecular Evolution and Phylogenetics*. Oxford University Press, June 2000.
- [20] A. R. Ortiz, C. E. Strauss, and O. Olmea. MAMMOTH (matching molecular models obtained from theory): An automated method for model comparison. *Protein Science*, 11(11):2606–2621, 2002.
- [21] A. Pascual-García, D. Abia, R. Méndez, G. S. Nido, and U. Bastolla. Quantifying the evolutionary divergence of protein structures: The role of function change and function conservation. *Proteins: Structure, Function, and Bioinformatics*, 78(1):181–196, 2010.

- [22] A. Pascual-García, D. Abia, Á. R. Ortiz, and U. Bastolla. Cross-over between discrete and continuous protein structure space: Insights into automatic classification and networks of protein structures. *PLOS Computational Biology*, 5(3):e1000331, Mar. 2009.
- [23] J. Rocha and R. Alberich. The significance of the ProtDeform score for structure prediction and alignment. *PLoS ONE*, 6(6):e20889, June 2011.
- [24] J. Rocha, J. Segura, R. C. Wilson, and S. Dasgupta. Flexible structural protein alignment by a sequence of local transformations. *Bioinformatics*, 25(13):1625–1631, July 2009.
- [25] M. I. Sadowski and W. R. Taylor. Evolutionary inaccuracy of pairwise structural alignments. *Bioinformatics*, 28(9):1209–1215, May 2012.
- [26] I. N. Shindyalov and P. E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(9):739–747, Sept. 1998.
- [27] N. Siew, A. Elofsson, L. Rychlewski, and D. Fischer. MaxSub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785, Sept. 2000.
- [28] Y. Zhang and J. Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.
- [29] Y. Zhang and J. Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309, Jan. 2005.

## Capítulo 11

# Modelos simplificados de plegamiento de proteínas

*Alberto Pascual-García*

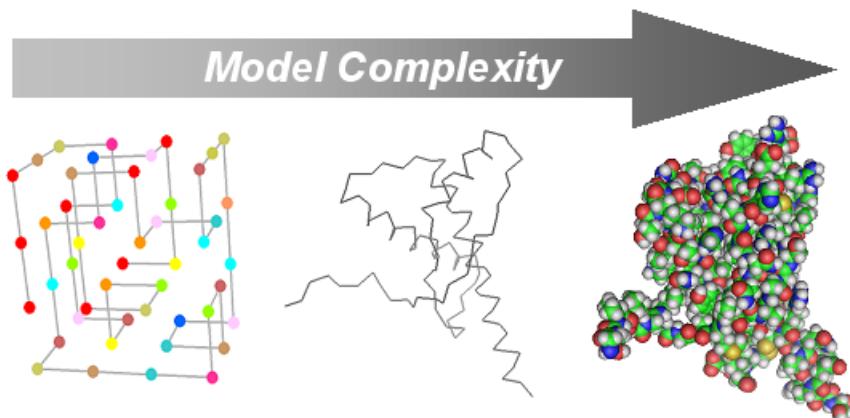
### 11.1. Introducción

En la presente sección vamos a presentar algunas aproximaciones computacionales orientadas a comprender los fundamentos del plegamiento de proteínas. *Por plegamiento entendemos el proceso espontáneo por el cual la proteína alcanza su conformación funcional*, es decir, su conformación nativa. Este proceso es espontáneo, por tanto termodinámicamente favorable, si se dan las condiciones apropiadas en el ambiente en el que está inmersa la proteína, tales como la concentración de sales o la temperatura. Además hay que tener en cuenta que en el medio natural puede ser necesaria la mediación de chaperonas, que evitan agregaciones o conformaciones subóptimas. También existe un creciente interés por regiones de las proteínas, que pueden incluso representar incluso una fracción muy significativa de la longitud total, cuya estructura no es posible resolver experimentalmente pero que pueden tener una relevancia funcional (las llamadas regiones intrínsecamente desordenadas) que serán tratadas en el Capítulo 13.

Desde el punto de vista biológico la importancia de conocer el plegamiento de una proteína reside fundamentalmente en la necesidad de conocer o profundizar en su función, si bien es un proceso interesante desde otros puntos de vista como por ejemplo el físico, ya que es un claro ejemplo de proceso autoorganizativo. Existen esencialmente *dos problemas de plegamiento*: 1) el que se centra en *entender el proceso* de plegamiento en sí, tanto dinámica como termodinámicamente y 2) el orientado a la *predicción de la estructura* de las proteínas. Nosotros discutiremos aquí el primer punto, dentro de una línea argumental que dará especial énfasis a los aspectos evolutivos, lo que nos permitirá ver la relación entre los dos problemas.

La literatura alrededor del problema de plegamiento es ingente y, dada la limitación de espacio de la que disponemos, es inevitable centrarse en únicamente algún tipo de modelos. Nosotros vamos a considerar una representación muy simplificada de las proteínas (ver la Figura 11.1, representación más simplificada). La justificación de esta simplificación tan fuerte reside en que podremos tratar el problema del plegamiento desde una perspectiva que nos permitirá sacar algunas conclusiones sobre el rol de la evolución (que se ampliará en el Capítulo 12) y con la que habremos establecido las bases para comprender también el problema de la modelización de estructuras de proteínas. Por tanto, los

modelos de plegamiento más detallado no serán tratados exhaustivamente en el libro, si bien creemos que el lector interesado habrá obtenido las bases necesarias para una aproximación autónoma a dichos problemas si combina los resultados de esta sección con los que puede encontrar en el capítulo de dinámica molecular y modelización de estructura de proteínas. En particular, dentro de los modelos simplificados que vamos a discutir, hemos seleccionado modelos que nos permitirán alcanzar una serie de objetivos, alrededor de los cuales está organizado el capítulo y que explicamos a continuación.



**Figura 11.1:** Complejidad de la representación en modelos de estructura de proteínas. Un nivel mayor de detalle como en los modelos que consideran todos los átomos en dinámica molecular (derecha) nos permite responder a preguntas concretas pero el coste computacional es muy elevado. A medida que vamos reduciendo la complejidad de la representación, como en los modelos que consideran carbonos alpha fuera de la red en la dinámica molecular de Langevin (centro) o simplemente aminoácidos en un retículo (izquierda), el coste computacional se reduce y podemos afrontar preguntas más generales como los principios básicos del plegamiento de proteínas. Esta figura ha sido gentilmente cedida por la Dra. Patricia Faísca<sup>1</sup>.

En la *primera sección*, hacemos una *introducción a resultados e ideas históricamente relevantes como la paradoja de Levinthal, el experimento de Anfinsen o el paisaje energético*. Los dos primeros puntos se refieren al problema del plegamiento en general, y será con el concepto de paisaje energético y el principio de mínima frustración cuando comenzaremos a introducir los modelos específicos que utilizaremos.

En la *segunda sección* presentaremos un *tratamiento analítico de dichos modelos simplificados de plegamiento*. Estos modelos nos permitirán hacer cierto tratamiento analítico sencillo en un problema que es muy complejo de atacar desde el punto de vista teórico. Se mostrarán algunos resultados teóricos que, aun siendo sencillos, serán accesibles a alumnos con cierto nivel en matemáticas y física, y serán probablemente difíciles para aquéllos que no dispongan de esta formación, que podrán prescindir de esta sección. El objetivo es que los alumnos interesados en la formalización tengan la oportunidad de encontrar desarrollos orientados a tratar conceptos relacionados con el problema de plegamiento, difíciles de encontrar de manera asequible para estudiantes de grado o máster en la literatura. Esto nos ha llevado a seleccionar modelos que se considerarían antiguos en este área. Este criterio de selección tiene como parte negativa que no permite conocer el estado actual del arte. Pero tiene como valor añadido el comenzar a aprender los conceptos en el orden en el que históricamente han sido explorados, lo cual permite fundamentar unas bases mínimas para continuar autónomamente el aprendizaje.

La *tercera sección* trata *ejemplos concretos* utilizando el mismo tipo de modelos. Los ejemplos que se discuten han sido seleccionados siguiendo dos criterios. El primero es que están basados en artículos

<sup>1</sup>Protein Folding Group, Universidad de Lisboa. [http://www.ciul.ul.pt/\\$\sim\\\$patnev/index.html](http://www.ciul.ul.pt/$\sim\$patnev/index.html)

científicos asequibles, lo cual creemos que es importante pues existe típicamente una falta de acercamiento de los estudiantes a la literatura científica. Por otra parte son ejemplos que tratan los conceptos discutidos anteriormente y hacen especial énfasis alrededor de argumentos evolutivos, lo cual permitirá al lector relacionar conceptos con otros capítulos del libro.

## 11.2. Conceptos básicos

### 11.2.1. La paradoja de Levinthal y el experimento de Anfinsen

Para comenzar, vamos a intentar evaluar la complejidad del problema que estamos afrontando. *Imaginemos que la conformación en la cual la proteína es funcional es única*, es decir, hay un único conjunto de coordenadas describiendo las posiciones de los átomos de la proteína en el espacio. Esta asunción es muy fuerte, porque obviamente la proteína estará en movimiento por su interacción con el medio. Pero sí es realista pensar que *la proteína se encuentra estadísticamente en un conjunto de conformaciones*, y que estas conformaciones son esencialmente las mismas si utilizamos medidas globales de “grano grueso” como el número de contactos (ver Capítulo 10). Estaríamos promediando (e incluso en algunos casos despreciando) magnitudes más detalladas como la posición de las cadenas laterales o movimientos de cuerpo rígido de elementos de estructura secundaria. No tenemos que perder de vista en ningún momento que, el no considerar *a priori* estas diferencias “en el detalle”, no implica que no puedan tener importancia en el proceso analizado. Así que debemos considerar este hecho cuando extraigamos conclusiones de cualquier análisis obtenido únicamente considerando una *descripción de “grano grueso”*. Sin embargo vamos a ver que, incluso considerando una descripción simplificada, la complejidad sigue siendo enorme.

Si consideráramos una proteína de unos 100 aminoácidos y cada aminoácido tuviera, por ejemplo, dos conformaciones posibles, explorar todas las conformaciones implicaría visitar  $2^{100} \sim 10^{30}$  estados posibles. Las proteínas muestran estados alrededor de su estado nativo a un ritmo cuyo orden de magnitud oscila entre  $0,1ps$  y  $10ns$ , pero ni siquiera esa velocidad es suficiente para explorar todas las conformaciones en un tiempo biológicamente relevante. Por ejemplo, para poner en un contexto computacional el problema, supongamos que somos capaces de explorar del orden de  $10^6$  conformaciones en un día. Esto es el número de conformaciones que se explora en una dinámica molecular utilizando 24 procesadores estándar y un paso de integración típico de  $0,01ps$  para generar  $1ns$  de trayectoria. *Necesitariamos entonces  $10^{24}$  días en explorar todas las conformaciones*, ¡lo cual es bastante más que la edad del universo!

Estábamos asumiendo en el cálculo anterior que la configuración en la que la proteína es funcional es única (salvo fluctuaciones), o si existen varias configuraciones son muy pocas en relación al número total de configuraciones posibles. Esta asunción se fundamentó en los años 60 gracias a los experimentos que llevó a cabo Anfinsen y sus colaboradores [1] sobre la cinética de la ribonucleasa y se ha convenido en llamar la *hipótesis termodinámica*. En dichos experimentos monitorizaban las propiedades estructurales en un proceso en el que, partiendo de unas condiciones estables para el plegamiento, llevaban a la proteína hasta la desnaturalización para volver después de nuevo a las condiciones iniciales. Su observación fue que, tras la desnaturalización, las propiedades estructurales no habían cambiado, lo que parecía sugerir que la proteína recuperaba de nuevo su estado nativo. La importancia de estos experimentos, por los cuales recibió junto a Moore y Stein (quienes determinaron la secuencia de la enzima) el Premio Nobel de Química, reside en que implica que *toda la información necesaria para que la proteína se pliegue en su estado nativo se encuentra codificada en la secuencia de aminoácidos*.

Este hecho motivó rápidamente la pregunta de si sería entonces posible entonces predecir la estructura

nativa de una proteína a partir de su secuencia. Pero unos pocos años más tarde, en 1969, Levinthal [17] planteó un importante problema conceptual que hemos ya ilustrado en el pequeño ejercicio anterior: si el considerar los grados de libertad de la proteína nos lleva a que el número de configuraciones posibles es astronómicamente grande, *¿cómo puede una proteína plegarse en una escala de tiempo biológicamente relevante?* Como hemos visto, aun asumiendo que somos capaces de muestrear las configuraciones en un tiempo muy pequeño, encontrar el estado nativo con una búsqueda exhaustiva o al azar sería inviable. Si la velocidad de plegamiento típica es del orden de milisegundos e incluso del microsegundo [16], ¿cómo se resuelve la paradoja de Levinthal?

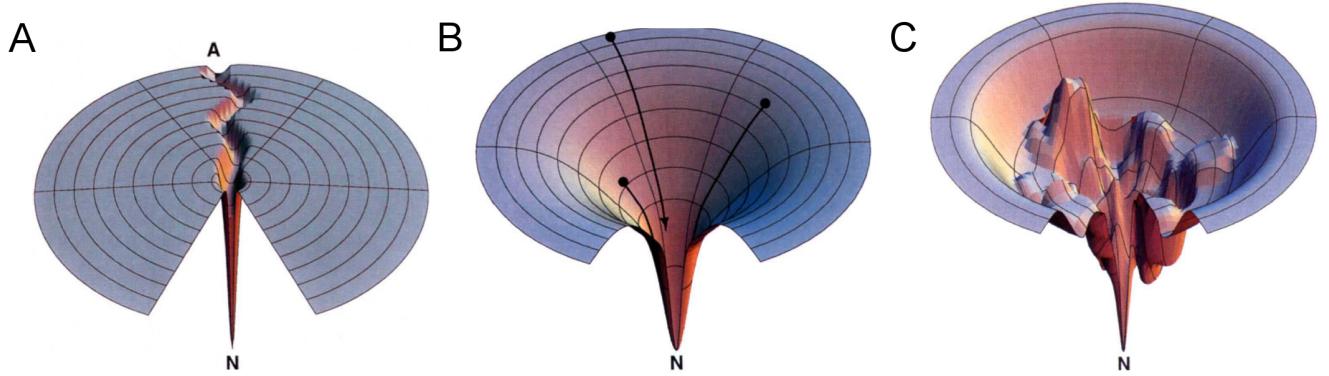
Por un lado, la *paradoja de Levinthal* lo que nos dice es que, *en un escenario en el que todas las conformaciones son equiprobables una búsqueda al azar es simplemente inviable*. Si los estados fueran efectivamente equiprobables, implicaría que encontrar la estructura nativa requeriría que existieran algunos caminos muy concretos que guiaran a la proteína hacia ese estado nativo, es decir, que existiera un *control cinético* muy estricto. Por el otro, los experimentos de Anfinsen señalan que es suficiente con que la proteína esté bajo *control termodinámico*, es decir, el estado plegado representa el mínimo absoluto de la superficie de energía libre. La realidad es mucho más compleja y en el plegamiento tanto el control cinético como el termodinámico son relevantes [6]. Cuánta relevancia tiene el control cinético o el termodinámico en el plegamiento de una proteína dependerá para comenzar de la naturaleza de la proteína y de su contexto ambiental. Nuestro objetivo será intentar dar una aproximación a esta pregunta desde el punto de vista físico y evolutivo.

### 11.2.2. El paisaje energético y el principio de mínima frustración

Desde el punto de vista de la paradoja de Levinthal, lo que plantea es un conflicto entre que la proteína encuentre un *mínimo de energía* (es decir, está bajo *control termodinámico*) y el que lo alcance *suficientemente rápido* (lo que implicaría algún tipo de *control cinético*). Lo primero que tenemos que observar es que este conflicto surge de la asunción de que el estado nativo de una proteína es igual de probable que cualquier otro. Esto sería como intentar meter una canica de un milímetro cuadrado de radio en el agujero de un campo totalmente plano y del tamaño de dos veces la superficie de España. Y en esta imagen lo principal es darse cuenta de que la canica difícilmente encontrará el agujero mientras el campo sea plano, pero la situación puede cambiar mucho si la superficie tuviera pendiente. Si consideráramos la superficie de energía libre de una proteína, el que la superficie no sea plana significaría que las distintas configuraciones de la proteína no son equiprobables. Ahora bien, *si exigimos únicamente control cinético podríamos tener una superficie de energía en el que hay algún tipo de “canalización” que nos lleva del estado desnaturalizado al plegado*, como se muestra en la Figura 11.2.

Si la proteína al desnaturalizarse se encuentra en el punto *A*, encontrará la canalización que le lleva hasta el estado nativo, pero este escenario es entonces fuertemente dependiente de las condiciones iniciales. La desnaturalización de la proteína (y, en general, las conformaciones intermedias) no deben considerarse como una conformaciones únicas sino como conjuntos estadísticos (en inglés *ensemble*) repartidos por la superficie de energía para determinados valores de la misma. De este modo, la paradoja de Levinthal sería más bien un artefacto que surge al considerar un escenario como el que acabamos de describir. A principios de los ochenta, surgió una visión en la que lo que se considera es que *la superficie de la energía libre no es plana, sino que más bien tendría una forma de embudo* (del inglés *funnel*) como en la Figura 11.2. Esto implicaría que *no existen caminos concretos por los cuales la proteína deba pasar, y por tanto con un número pequeño de conformaciones intermedias*. Sino que más bien existirían conjuntos estadísticos suficientemente grandes para que el control cinético tenga lugar, por los cuales la proteína puede alcanzar el mínimo de energía, haciendo compatible el control cinético y termodinámico.

¿Qué particularidades podemos esperar de esta superficie? Por un lado, esperaríamos que tuviera



**Figura 11.2:** Ilustración de posibles paisajes energéticos. En el eje vertical se representa la energía libre sin considerar la entropía conformacional, una “energía libre interna”. Un punto en la superficie representaría una conformación, que a su vez contiene un número elevado de grados de libertad. Frente a un escenario en el que el paisaje energético es plano excepto para el estado nativo, como se asume en la paradoja de Levinthal que implicaría una búsqueda al azar entre conformaciones equiprobables, el estado nativo (*N*) podría ser alcanzado más rápidamente desde un estado desnaturalizado (*A*) si existiera un camino en la superficie energética preferente (A). Un paisaje en forma de embudo (B) permitiría considerar conjuntos estadísticos de conformaciones energéticamente más favorables. De este modo la proteína alcanzaría el mínimo de energía (control termodinámico) pero habría múltiples caminos. En un paisaje más realista (C) hay que considerar que la superficie no es suave porque existen fuentes de frustración (ver texto). Figuras extraídas de [8] con permiso del editor.

cierta forma suave de embudo, lo que permitiría “guiar” a la proteína hacia el estado nativo en una escala de tiempo razonable. Esto no significa que sea sencillo porque, al descender por el embudo, el espacio conformacional sigue siendo grande y encontraremos aún cuellos de botella en la cinética del plegamiento en regiones energéticamente cercanas al estado nativo precisamente porque el número de conformaciones accesibles es aún elevado, como veremos en uno de los ejemplos más adelante (una buena aclaración al respecto la encontramos en [14]).

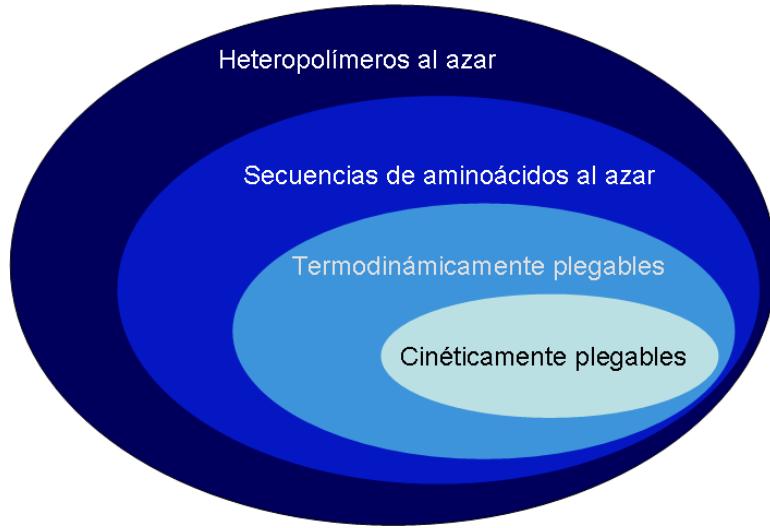
Además, la complejidad de las interacciones de la proteína que se encuentra sometida a las fluctuaciones del medio nos lleva a pensar que habrá *múltiples mínimos locales debidos a conformaciones no nativas*, pero con cierta globularidad, que se forman durante el proceso y que darían lugar a un paisaje energético más bien rugoso como muestra la Figura 11.2. Estos mínimos locales serían “*trampas*” en las que la proteína se podría quedar atrapada si se dan las condiciones ambientales apropiadas.

Es lógico postular entonces que el paisaje energético de una proteína real sea algo intermedio entre una superficie suave y en forma de embudo y una superficie rugosa sin una pendiente global determinada. Este fue el razonamiento seguido por Bryngelson y Wolynes [4] para abrir una vía a modelos simplificados de la superficie energética de proteínas, basado en la bien conocida teoría física de los vidrios de espín. Según su planteamiento, un polímero con una secuencia al azar debería de presentar una superficie rugosa debido a que tendrá una fuente importante de *frustración*. La frustración debemos entenderla como un conflicto irresoluble entre distintas tendencias que tienen lugar en el sistema que es sometido a nuestro análisis, donde tendencia se puede entender en un sentido muy general. Por ejemplo, en cualquier clasificación diremos que está frustrada si no es posible realizarla sin ambigüedad. Veremos un ejemplo al respecto en el capítulo de evolución de estructura de proteínas, al intentar clasificar estructuras (Subsección 12.3.1). En el caso que nos ocupa, la frustración surge de que existen interacciones que entran en conflicto tanto por su naturaleza (determinada por la secuencia de la proteína) como por las restricciones de carácter geométrico. Si imaginamos por ejemplo una secuencia

al azar en donde consideramos únicamente una propiedad, por ejemplo tenemos residuos hidrofílicos e hidrofóbicos, no será posible encontrar una configuración en la que todos los residuos hidrofóbicos estén en contacto formando un núcleo compacto alrededor del cual se encuentren los residuos hidrofílicos, que sería la configuración más estable en medio acuoso. Así que en una secuencia al azar, esperaremos que la superficie de energía libre sea rugosa, y que haya *numerosos mínimos locales* en donde la proteína se queda atrapada en *configuraciones semicompa*ctas.

Por otro lado, los paisajes energéticos suaves a pesar de no tener trampas locales, parten de un gran número de estados de energía elevada para ir reduciéndose a unos pocos de baja energía. Diremos entonces que existe un “*cuello de botella*” si existe alguna región de la superficie de energía libre en donde esta reducción es muy abrupta, y que podremos estimar con la disminución relativa del número de estados. Si este cuello de botella no es muy dramático, la proteína encontrará el estado nativo en un tiempo biológicamente relevante, como veremos en uno de los ejemplos discutidos. Este tipo de paisajes (superficie no muy rugosa y cuello de botella suficientemente ancho) será esperado entonces en sistemas en donde exista una alta *cooperatividad*, ya que implica una coordinación que reduce las posibles fuentes de frustración. La última característica que queremos resaltar de los sistemas que poseen un paisaje energético suave es que sus fases termodinámicas están determinadas por la temperatura, como veremos más adelante (ver Figura 11.7).

Bryngelson y Wolynes observaron que, si bien en proteínas reales debe existir cierta frustración, como se encuentra en polímeros cuya secuencia es al azar, esta frustración debería de ser mínima para que la cinética de la proteína sea lo suficientemente rápida para plegarse en el medio celular. Además, las estructuras globulares son muy compactas, lo que implica que la estructura secundaria y terciaria no entran en conflicto. Y además implicaría que, desde el punto de vista evolutivo, las secuencias de proteínas reales deberían haber sido seleccionadas para cumplir los requerimientos termodinámicos y cinéticos compatibles con su plegamiento. Llegamos entonces a un escenario en el que *las proteínas reales son un subconjunto de todas las cadenas polipeptídicas al azar que podríamos construir*, como ilustramos en la Figura 11.3.



**Figura 11.3:** Partición del conjunto de heteropolímeros al azar en subgrupos en función de las restricciones que imponemos. El conjunto de polímeros basados en los aminoácidos encontrados en la naturaleza es un primer subgrupo dentro del cual encontraremos aquéllos termodinámicamente plegables, y a su vez aquéllos cinéticamente plegables a una temperatura relevante para la función biológica. Téngase en cuenta que las superficies relativas de los conjuntos no son realistas, ya que las diferencias entre cada grupo y el siguiente serían mucho mayores. Figura adaptada de [22].

## 11.3. Fundamentos teóricos

En la presente sección, vamos a presentar algunos *fundamentos teóricos* que, de manera natural, nos llevan a comprender formalmente los posibles ingredientes necesarios *para describir una superficie de energía* razonablemente realista. Si bien sería conveniente ciertos conocimientos en mecánica estadística para comprender esta sección no es necesario comprender los detalles matemáticos alrededor de los resultados, si bien mostramos algunos por completitud. Para los lectores que lo requieran, una primera introducción a la física estadística la encontrarán en [31] y un buen resumen pedagógico de la aplicación de la mecánica estadística a los modelos simples de plegamiento se encontrará en [24]. Es en la última sección en donde, dotados de los conocimientos adquiridos, abordaremos ejemplos computacionales concretos que ayudarán a afianzar los contenidos.

### 11.3.1. El paisaje energético del modelo de energía al azar

Es conveniente comenzar entendiendo qué propiedades esperaríamos si tuviéramos el modelo más simple posible. En general, el modelo más simple es aquél que explica un fenómeno observado como consecuencia del azar. Si introdujéramos nuestro modelo en el contexto estadístico de un *contraste de hipótesis*, podríamos de hecho considerar dicho modelo como aquél asociado a una *hipótesis nula* (modelo nulo). Este ejercicio es importante realizarlo siempre que se intente introducir un nuevo formalismo, ya que será la descripción más económica del proceso que estemos observando. Sus resultados servirán como referencia para evaluar la significatividad de los patrones que encontramos experimentalmente al contrastarlos con el modelo nulo, patrones que aceptaremos como *hipótesis alternativa* si conseguimos falsar el modelo nulo. La hipótesis alternativa contendrá algún ingrediente adicional al efecto del azar (cuya evaluación es probablemente la que motiva nuestro análisis).

En el caso que nos ocupa, podríamos considerar como aproximación de orden cero la aplicación del

modelo de energía al azar propuesto por Derrida [7] y aplicado por vez primera en polímeros por Bryngelson y Wolynes [4]. Vamos a examinar con este modelo qué superficie de energía obtenemos, para lo cual vamos a realizar asunciones muy fuertes. La superficie de energía la entendemos en este ejercicio como el espectro energético de los distintos estados en los que podemos encontrar a nuestra proteína problema.

Dada una cadena de monómeros, consideraremos un *conjunto de microestados* (conformaciones estructurales),  $m_1, m_2, \dots, m_M$  con sus respectivas energías  $E_1, E_2, \dots, E_M$ . El número  $M$  de microestados es enorme como ya hemos visto. En particular, lo podemos estimar como  $M = \gamma^N$ , donde  $N$  es el *número de residuos* de la proteína y  $\gamma$  es el *número de configuraciones por residuo*. El valor de  $\gamma$  será menor que 5 cuando hagamos representaciones reducidas de la proteína, como las que hemos explicado, pero puede llegar a ser del orden de 10 si se consideran las cadenas laterales. En cualquiera de los casos, se irá reduciendo a medida que la proteína se vaya compactando.

¿Cómo estimaremos las energías? En primer lugar hay que tener en cuenta que, para una conformación  $m_i$  determinada su energía depende, no solamente de la estructura que la define, sino de lo que formalmente llamaremos una *realización del desorden*. En nuestro caso, una realización del desorden será una secuencia de aminoácidos que generamos al azar (de aquí el nombre de modelo de energía al azar). Lo que pretendemos al considerar un número grande de secuencias (realizaciones), es asignar a cada conformación todas las energías que se obtendrían ajustando cada una de las secuencias a dicha conformación. Esto podríamos hacerlo en la práctica asumiendo algún modelo sencillo para representar las estructuras (como veremos en la siguiente sección) y ajustando cada secuencia a las estructuras, para lo cual calcularemos la energía asociada al par secuencia-estructura con alguna definición que estimemos oportuna.

De momento vamos a asumir simplemente que, para cada conformación  $m$ , las realizaciones de desorden (secuencias) ajustadas sobre dicha conformación nos dan un *conjunto de energías* que siguen una cierta *distribución de probabilidad*:

$$P(E) = \frac{1}{\sqrt{2\pi\Delta E^2}} \exp\left(-\frac{E^2}{2(\Delta E)^2}\right) \quad (11.1)$$

Donde estamos diciendo que estas energías son simplemente variables aleatorias que siguen una distribución Gaussiana con ancho  $\Delta E$ . Vamos a hacer además una asunción aún más fuerte. Asumamos que, dada una conformación  $m_i$ , y otra conformación con energía  $m_j$ , sus distribuciones de energías no están correlacionadas, es decir:  $P(E_i, E_j) = P(E_i)P(E_j)$ . Esta es una asunción de *independencia entre las conformaciones* que es obviamente falsa, porque habrá muchas conformaciones tan parecidas entre sí que las correspondientes energías que calculemos al generar secuencias en ambas sí estén correlacionadas. Podrían ser incluso exactamente la misma distribución aun siendo conformaciones distintas, basta imaginar dos conformaciones que sean la una imagen especular de la otra. Pero esta asunción va a simplificar mucho nuestro cometido, pues ahora queremos saber cómo se distribuyen los microestados (conformaciones) a lo largo de los posibles valores de la energía. Es decir, queremos calcular  $n(E)$ , que es el número de estados comprendidos entre  $(E, E + dE)$ . Como tenemos el número estimado de configuraciones podemos calcular fácilmente el valor esperado del *número de estados*  $\langle n(E) \rangle$ , que resulta:

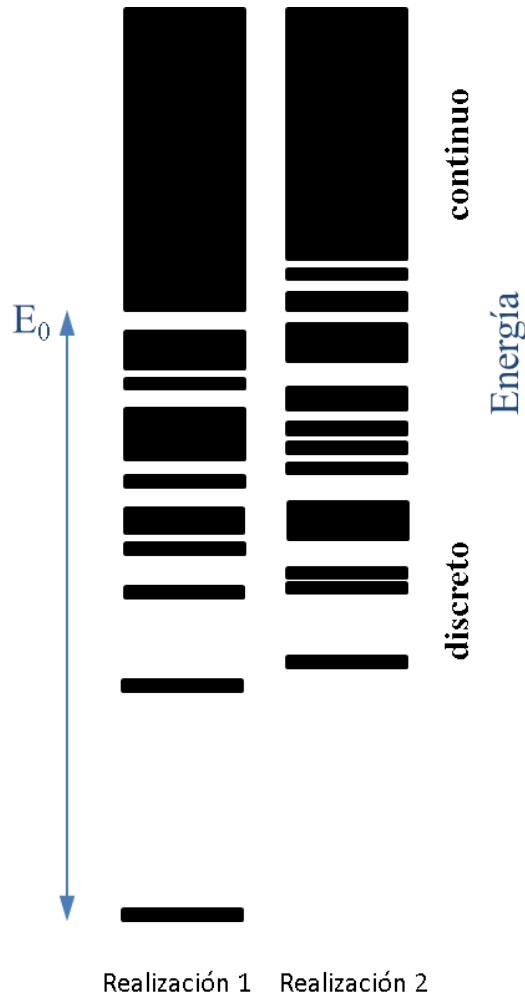
$$\langle n(E) \rangle = \gamma^N P(E) \sim \exp\left(N \ln \gamma - \left(\frac{E}{\Delta E}\right)^2\right) \quad (11.2)$$

De esta expresión vemos que, como el número de conformaciones  $\gamma^N$  es muy grande, la densidad de estados también lo será siempre y cuando  $P(E)$  no sea muy pequeña. Si  $P(E)$  no es pequeña, quiere decir que nos encontraremos un espectro continuo de energías. Y en ese continuo no seremos capaces de distinguir los estados independientemente de la realización de secuencia que tengamos. Pero si la

probabilidad disminuye lo suficiente la densidad de estados cambia significativamente. Siguiendo la Ecuación 11.2, el número de estados y la probabilidad son del mismo orden ( $\gamma^N P(E) \sim 1$ ) para una determinada *energía crítica*  $E_0$ :

$$E_0 \approx -\Delta E \sqrt{\ln \gamma} \quad (11.3)$$

tal que si  $|E| > E_0$ , la densidad de estados es menor que 1. Y este valor dependerá, a su vez, de  $\Delta E$ , que podemos interpretar como la rugosidad de la superficie de energía, de modo que disminuyendo la rugosidad podemos alcanzar antes el valor crítico, facilitando el plegamiento. Hay que observar además que, si tenemos menos de un estado en el intervalo de energía  $(E, E + dE)$ , esto significa que habrá regiones de dicho intervalo en donde no encontramos ningún estado. Tenemos por tanto que la distribución de energías por debajo de la energía crítica se *discretiza*, es decir, habrá determinados intervalos en los que encontramos algún estado y otros en los que no encontraremos nada.



**Figura 11.4:** Espectros de energía típicos del modelo de energía al azar. A diferencia de lo que ocurre en la región continua del espectro que es indistinguible para las distintas realizaciones del desorden, a partir de una energía crítica  $E_0$  encontramos una discretización que sí es específica para cada realización del desorden considerado.

¿De qué dependerá que encontremos o no un microestado en un determinado intervalo y cuál encontraremos? Pues dependerá precisamente de la secuencia con la que estemos trabajando, porque dicha secuencia se encontrará para cada microestado con una determinada probabilidad asociada en su

correspondiente Gaussiana. Para la conformación o conformaciones en las que dicha secuencia tiene probabilidades muy bajas de la energía obtendremos una contribución al espectro de energía. Será entonces en esta región discretizada en donde podremos distinguir las realizaciones, y podríamos considerar el espectro como una “*huella de identidad*” para cada realización (ver Figura 11.4). Este resultado tan sencillo es relevante porque nos abre la puerta a plantearnos si esa huella de indentidad podría ser distinta cuando tratamos con secuencias al azar o cuando tratamos con secuencias reales, ya que estas últimas han superado un proceso de purificación a través de la selección natural. Veremos más adelante que parece que sí podemos pensar en un determinado rasgo característico de los espectros de proteínas reales respecto de aquéllas que no lo son.

Para continuar con nuestro ejemplo, podemos preguntarnos ahora cuál es el *estado termodinámicamente más probable para una realización*. Para ello multiplicamos la densidad de estados de energía  $E$  por el factor de Boltzmann, con lo que obtenemos:

$$p(E) = \frac{1}{Z} \gamma^N P(E) \exp(-E/k_B T) \quad (11.4)$$

Donde  $Z$  es la función de partición. Como hemos visto, el número de estados (y por tanto la entropía) se reduce drásticamente por debajo de un cierto valor mínimo. Si maximizamos la Ecuación 11.4 con respecto de la energía, obtendremos el número de estados ocupados a una temperatura dada [3]:

$$\gamma^N P(E_{m.p.}) = \exp\left(\frac{S}{k_B} - \frac{\Delta E^2}{2(k_B T)^2}\right) \quad (11.5)$$

Donde  $S$  es la entropía y el subíndice *m.p.* nos indica que es la energía más probable. De este ejercicio obtenemos algunos resultados también interesantes para nuestra comprensión del proceso de plegamiento. La dependencia con la temperatura nos lleva a observar que, al reducirse, entraremos en la región en donde el número de estados tiende a cero. La transición de una a otra región tendrá lugar para un valor crítico de la entropía  $S_0$ , y a la temperatura asociada le llamaremos temperatura de transición vítreo<sup>2</sup>,  $T_G$ . Este valor lo obtenemos cuando en la Ecuación 11.5 el argumento de la exponencial tiene un valor en torno a cero, que implica que encontramos la región discreta:

$$T_G = \left( \frac{\Delta E^2}{2k_B S_0} \right)^{1/2} \quad (11.6)$$

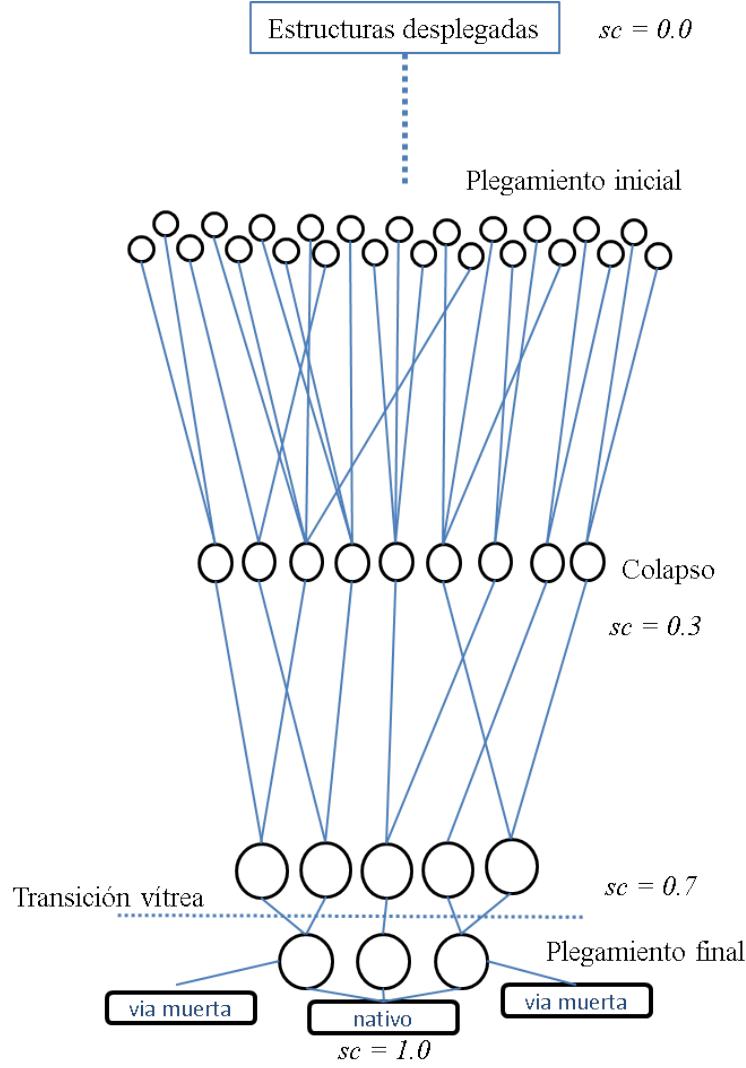
A esta temperatura el sistema sufre una *transición de fases* y se “congela” accediendo a unos pocos estados (ver Figura 11.5). Más adelante veremos cómo podemos incluir más propiedades de la superficie de energía para derivar un mapa de fases en el que aparecerá una nueva temperatura relevante, la *temperatura de plegamiento*  $T_F$ , que podremos relacionar con la *temperatura de vitrificación*  $T_G$ . Y veremos que la relación entre ambas es una de las características que nos permitirán diferenciar a las proteínas reales de las generadas por azar.

### 11.3.2. Un paisaje energético un poco más realista

Como explicamos anteriormente, podemos esperar que en proteínas reales los contactos nativos sean más favorables energéticamente de lo esperado por azar, como resultado del proceso evolutivo. Un modo de implementar computacionalmente estas ideas fue aplicar un tipo de modelo como el propuesto por *Gō* [29]. En este tipo de modelos se intenta *estudiar la dinámica y termodinámica de plegamiento de la*

---

<sup>2</sup>El subíndice *G* lo utilizamos para referirnos a la notación que se utiliza típicamente en inglés, donde se denomina *glass transition*. Del mismo modo utilizaremos el subíndice *F* para referirnos a la temperatura de plegamiento, del inglés *folding*.



**Figura 11.5:** Representación esquemática de los caminos en el plegamiento de la proteína. Hay distintos caminos posibles para cada conformación, lo que hace que el estado nativo pueda ser cinéticamente accesible. Se describen los distintos estadios en función del parámetro de orden  $sc$  (solapamiento de contactos). Tras la transición vítrea hay pocos caminos posibles y algunos de ellos pueden llevar a conformaciones mal plegadas (vías muertas). Figura adaptada de [22]

proteína una vez es conocido su estado nativo. Como en el modelo de energía al azar, el valor de estos modelos residen en su capacidad de aumentar nuestra comprensión del problema (porque obviamente no nos va a servir para plegar una proteína cuando las condiciones del modelo implican que ya la conocemos). De modo que se construye un potencial en el que, si la interacción que tiene lugar entre dos residuos es la misma que la que obtienen en su estado nativo será energéticamente más favorable que en otra configuración de contactos, en donde será equivalente a lo esperado en una interacción al azar. Esto inducirá un *embudo* dentro de la superficie rugosa que construimos considerando interacciones al azar, como vimos en la Figura 11.2.

Para monitorizar la dinámica a lo largo de la superficie, introduciremos un *parámetro de orden*, es decir, una cantidad que nos indique cómo de cerca estamos del estado nativo y por tanto nuestra posición en el embudo energético. Hay varias opciones posibles como el radio de giro o el número de ángulos que

coinciden con los del estado nativo. Nosotros, por simplicidad conceptual y por lo extendido de esta elección, vamos a trabajar con el *número de contactos compartidos* o *solapamiento de contactos* ( $sc$ ), cuya definición se puede encontrar la Sección 10.3. Dada una conformación de la proteína a lo largo de la reacción de plegamiento, definimos el solapamiento de contactos simplemente como el número de contactos compartidos entre la conformación considerada y el estado nativo.

El tratamiento analítico es prácticamente el mismo que en el modelo de energía completamente al azar. La principal diferencia es que ahora necesitamos *obtener las variables termodinámicas* como la energía media  $\bar{E}(sc)$ , rugosidad  $\sqrt{\Delta E^2(sc)}$ , entropía  $S(E, sc)$  y temperatura de transición  $T_G(sc)$ , *en función del parámetro de orden elegido* ( $sc$ ). Cuando el parámetro de orden  $sc \approx 0$ , tendremos un conjunto estadístico en el que las transiciones dependen simplemente de la rugosidad, como en el modelo de energía al azar, mientras que cuando el parámetro de orden tome valores elevados (del orden del 70 % de solapamiento) habrá pocos estados y estaremos interesados en conocer en detalle la cinética de transición entre los mismos.

Asumiremos ahora una distribución de probabilidad, para cualquier estrato del embudo energético, de que una estructura con solapamiento de contactos  $sc$  con el estado nativo tenga una energía  $E$ :

$$P(sc, E) = \frac{1}{\sqrt{2\pi\Delta E^2(sc)}} \exp\left(-\frac{[E - \bar{E}(sc)]^2}{2\Delta E^2(sc)}\right) \quad (11.7)$$

El desarrollo formal a partir de esta distribución, asumiendo la hipótesis de independencia entre estados de energía del modelo de energía al azar, es completamente análogo al caso anterior por lo que lo dejamos como ejercicio para el lector, que podrá encontrar resuelto en la referencia [3]. Vamos a interpretar directamente el resultado que obtenemos para la *energía libre*, que se obtiene como el logaritmo de los estados pesados termodinámicamente para una similitud dada [5]:

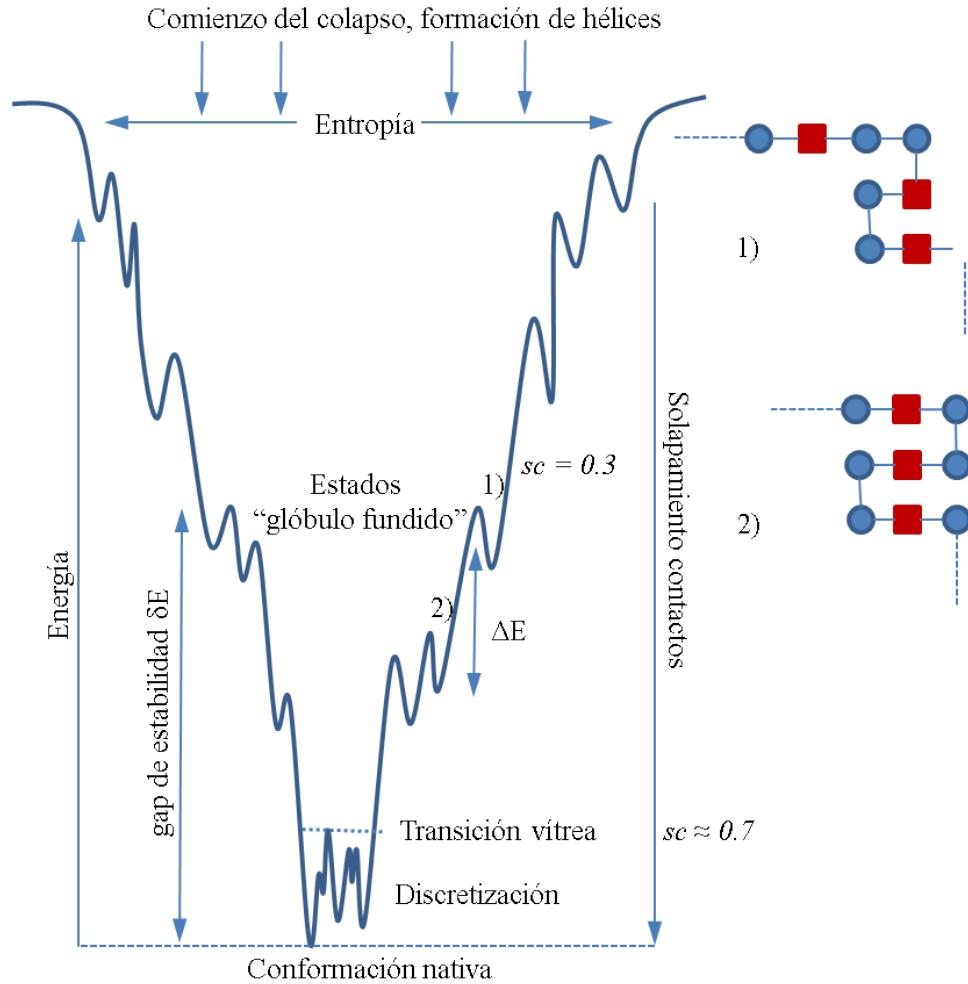
$$\begin{aligned} F(sc) &= E_{m.p.}(sc) - TS(E_{m.p.}(sc), sc) \\ &= \bar{E}(sc) - \frac{\Delta E^2(sc)}{2k_B T} - TS_0(sc) \end{aligned} \quad (11.8)$$

donde  $E_{m.p.}$  es la configuración con energía más probable dado un solapamiento de contactos. La temperatura determina el balance entre la componente energética y la entrópica, pero además podemos ver qué valores del parámetro de orden se corresponden con el dominio de uno u otro término. Cuando la temperatura es alta la energía libre tendrá un único mínimo. Este mínimo puede corresponder a dos posibles conjuntos, el de estados completamente desplegados y el de un conjunto en el que la proteína colapsa pero de manera desordenada denominado *glóbulo fundido* (del inglés *molten globule* [28]). Con nuestra descripción no sería posible separar los dos conjuntos, y necesitaríamos un segundo parámetro de orden como el radio de giro [9]. Para energías bajas tendremos otro mínimo en el que son configuraciones de estados plegados las dominantes pero también tendremos que ir a modelos más complejos para poder diferenciar entre los estados que se encuentran en una fase vítreo (compactos pero no similares al nativo) y los propiamente similares al nativo. A temperaturas intermedias ambos mínimos estarán poblados pero con una ocupación mínima entre ambos, indicando la existencia de una barrera de energía.

### 11.3.3. Cinética del plegamiento y *gap* de estabilidad

Como en el modelo de energía al azar sin parámetro de orden, podemos encontrar la *temperatura de vitrificación*, cuya expresión es la misma que la Ecuación 11.6, pero donde ahora la entropía será una función del parámetro de orden. Por debajo de esta temperatura, el sistema “expulsa” la entropía ( $S = 0$ ) y queda atrapado en un *microestado vítreo*. Aunque aún podría pasar a algún otro estado

compacto y haya menos estados en esta región, la búsqueda del estado nativo no es posible en una escala de tiempo razonable y volveríamos a un resultado como el que pretendemos resolver, a saber, la paradoja de Levinthal.



**Figura 11.6:** Ilustración de un paisaje energético en forma de embudo para una proteína pequeña (todo  $\alpha$ ), en función de los parámetros de orden  $sc$  (solapamiento de contactos) y  $E$  (la energía promediando el disolvente). Se representa la rugosidad  $\Delta E$  relacionándolo con un cambio de conformación local ilustrado en los puntos 1 y 2, y el *gap* de estabilidad  $\delta E$ . Figura adaptada de [22].

Para resolver este problema, debemos introducir correlaciones en los estados de energía de modo que estén dinámicamente conectados [3, 4, 5]. El introducir correlaciones implica que existen combinaciones de aminoácidos que son favorables para la cinética de plegamiento, lo que nos lleva a buscar entender la relación entre la superficie de energía y la secuencia. Utilizando modelos más complejos [12] podemos estimar la temperatura de plegamiento  $T_f$  en función de parámetros relacionados con la superficie de energía:

$$T_f = \frac{\delta E + (\delta E^2 - 2S_0\Delta E^2)^{\frac{1}{2}}}{2S_0} \quad (11.9)$$

En esta expresión el nuevo parámetro que aparece,  $\delta E$ , se denomina *gap de energía* o *gap de estabilidad*<sup>3</sup>

---

<sup>3</sup>Se ha preferido no traducir aquí el término *gap*, porque el referente al que alude en inglés creemos que contiene

y es un parámetro con el que vamos a poder caracterizar diferencias entre proteínas reales y polímeros al azar. En la Figura 11.6 se ilustran los ingredientes introducidos hasta el momento en el paisaje energético. Existen varias definiciones de *gap* de energía más o menos restrictivas pero que siguen la misma idea. Por ejemplo, se puede definir como la diferencia de energía entre la media de energías de los estados desplegados y la media de los estados plegados. A continuación relacionamos la temperatura de plegamiento con la temperatura de vitrificación [22], obteniendo:

$$\frac{T_f}{T_g} \approx \frac{\delta E}{\Delta E} \sqrt{\frac{2k_B}{S_0}} \quad (11.10)$$

Tanto numérica como experimentalmente se ha observado que el cociente  $T_f/T_g$  debe ser mayor que 1 para que el estado nativo sea cinéticamente accesible. Así que nos acercamos a entender qué esperaríamos de la superficie de energía en proteínas reales, porque observamos que el aumentar la relación entre el *gap* de estabilidad y la rugosidad  $\delta E/\Delta E$  mejorará la *accesibilidad del estado nativo*, y por tanto las posibilidades de acceder al mismo en un tiempo biológicamente relevante. Hay que notar que tanto la energía como la rugosidad y la entropía son variables que dependen linealmente de la longitud, por lo que esta medida es independiente de la longitud de la proteína. En las secciones siguientes ampliaremos nuestro estudio alrededor del *gap* de energía, lo que nos conducirá progresivamente a dejar más de lado la física para introducirnos en la evolución. Vamos a profundizar un poco más en la relación entre los estados plegados compactos y los estados vítreos a través de un diagrama de fases.

#### 11.3.4. Diagrama de fases

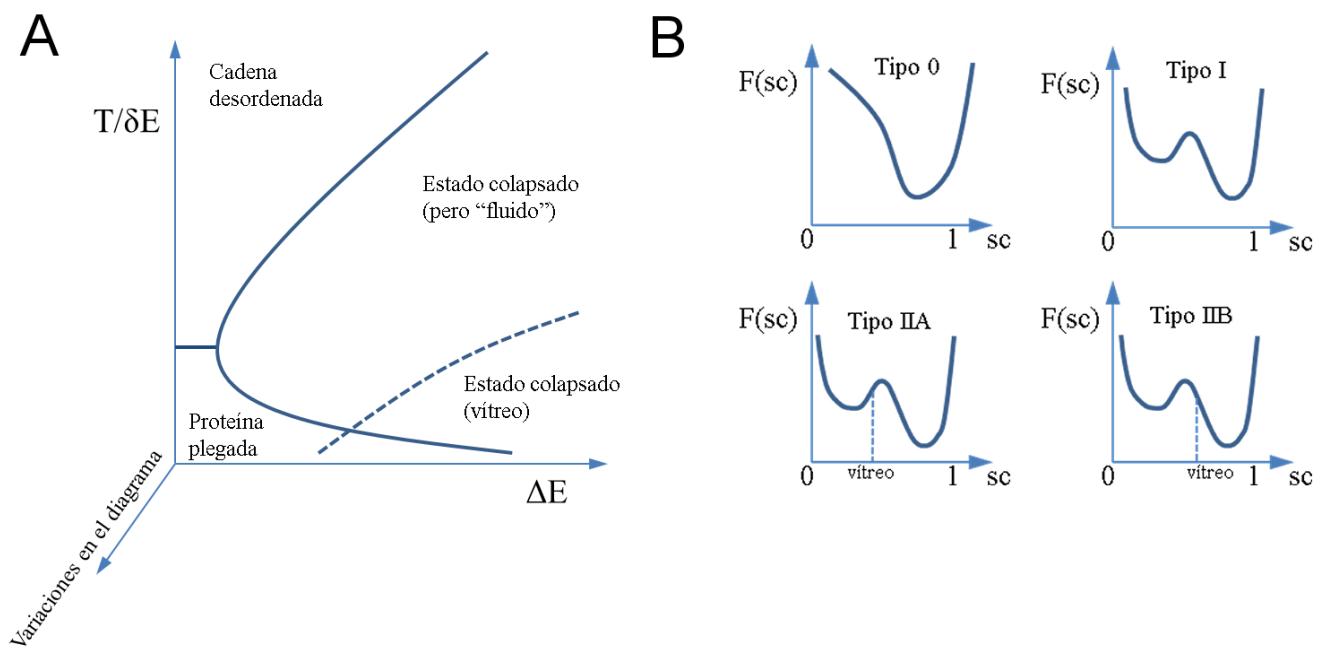
Las expresiones típicas que encontramos en la literatura para la energía libre en función de un parámetro de orden como el *sc* tienen uno o dos mínimos de energía en estos modelos formales sencillos [28]. Si tuviéramos un único mínimo cercano al escenario nativo, estaríamos ante un escenario (Tipo 0 en la Figura 11.7) que se denomina “colina abajo”, del inglés *downhill*, porque, si pensamos en el paisaje de la energía libre, no encontraríamos ninguna barrera y “bajaríamos” directamente al estado nativo [21]. Este escenario se ha observado en algunos casos experimentalmente [11] y podría ser común en condiciones fuertemente propicias para el plegamiento, principalmente para proteínas pequeñas. En este escenario que llamamos de *Tipo 0* (ver la Figura 11.7) la fase desordenada es *inestable*, y la proteína se pliega espontáneamente en un proceso cooperativo dirigiéndose hacia el mínimo de energía, como muestra en la figura el perfil de energía libre.

Existen además sistemas en los que la fase desordenada es estable (*Tipos I y II*), que son los que vamos a representar en un diagrama de fases como el que mostramos en la Figura 11.7. A estos sistemas les llamaremos *biestables*. En la figura tenemos una región en donde las conformaciones son equivalentes a las de un polímero aleatorio, otra región correspondiente al estado de glóbulo fundido (que es contigua al estado vítreo) en el que la cadena colapsa, y por último una región en la que encontraríamos propiamente el plegamiento nativo. Las transiciones entre las distintas fases vienen señaladas por líneas continuas (si son de primer orden) o discontinuas (de segundo orden). La transición del estado desplegado al plegado tiene lugar a través de una *barrera de energía* (ver las representaciones en la parte inferior de la Figura 11.7). Esta barrera de energía supone un cuello de botella en el número de caminos posibles de la proteína hacia el plegamiento, que influirá en el tiempo de plegamiento.

Las dos variables que determinan el diagrama de fases nos permiten entender bien en qué región nos encontraríamos en función de las propiedades que hemos ido mostrando. Vemos que un aumento

---

implícitamente más información que las posibles traducciones al español. Por un lado indica un salto en los valores de energía pero, como veremos, existe una discretización en el espectro con regiones vacías. Creemos que el término *gap* contiene ambos referentes, salto y vacío.



**Figura 11.7:** A. Diagrama de fases de una proteína plegable biestable. En el eje horizontal se representa la rugosidad energética y en el eje vertical la temperatura dividida por el *gap* de estabilidad. Las distintas fases estables se identifican en las distintas regiones. B. Distintos escenarios de plegamiento en los que se representa cualitativamente el perfil de energía libre (eje vertical) frente a un parámetro de orden como el solapamiento de contactos. Ver el texto principal para la explicación de las figuras. Figura adaptada de [3].

de la temperatura nos lleva a regiones en donde el polímero se encuentra en conformaciones más extendidas, pero esta variable se ve modificada por una propiedad determinada por la secuencia, el *gap* de estabilidad. Un *gap* de estabilidad mayor hará más probable mantener la proteína en su estado nativo. Por otro lado nos encontramos en el eje horizontal la segunda variable que es determinada por la secuencia,  $\Delta E$ , que sabemos está relacionada con la rugosidad de la superficie de energía. Esperamos que una mayor rugosidad incremente la probabilidad de que la proteína encuentre un estado vítreo en el que colapsa para ciertas temperaturas, en vez del estado nativo. Entrar en la fase vítreo implica que el fenómeno es menos cooperativo, lo cual hace que sea más lento y que sea necesaria la búsqueda de caminos hacia el estado nativo, es decir, es un proceso más exigente cinéticamente. Experimentalmente se deberían de observar entonces estados intermedios en los que la proteína se queda atrapada. Es importante señalar también que este diagrama de fases variaría en función de otros parámetros que no estamos teniendo en cuenta al simplificar nuestro modelo. Por un lado la temperatura afecta a la propia rugosidad y al *gap* de estabilidad. Y más en general, cualquier modificación que afecte a la hidrofobicidad modificará el diagrama, con lo que tenemos que considerarlo como una “rebanada” de un diagrama de fases mucho más complejo.

Podemos imaginar distintos tipos de plegamiento en función de las características de la proteína que observemos. Si estuviéramos a la izquierda del todo, la proteína pasaría de un estado desordenado directamente al nativo al reducir la temperatura. Este tipo de escenario lo denominamos de *Tipo I*, y la accesibilidad cinética al estado nativo viene determinada por el número relativo de estados a los que la proteína puede acceder en el cuello de botella. En este caso el plegamiento es altamente cooperativo porque no existe vitrificación, y a pesar de que existe un cuello de botella encontraremos la proteína podrá acceder a un número importante de los estados que existen en este máximo de la energía libre. La velocidad de plegamiento se podría entonces estimar a través de la fracción de estados accesibles. Es importante señalar por último que, el que la proteína pueda visitar un número importante de estados en el cuello de botella del plegamiento la hará más robusta frente a mutaciones puntuales. En el capítulo de evolución de estructura de proteínas veremos que es de esperar que una única mutación no tenga un efecto muy dramático en la estructura de la proteína. Esto lo explicaríamos diciendo que una única mutación no varía muy significativamente el número de estados accesibles en el cuello de botella. Sin embargo, si el número de estados accesibles es muy pequeño como veremos a continuación, una mutación podría tener un efecto dramático si eliminara algunos de los pocos caminos accesibles.

Según nos movemos hacia la derecha en el diagrama de fases aumenta la rugosidad de la superficie de energía, y encontraríamos una fase vítreo. En función de la rugosidad del paisaje energético (más o menos a la derecha), encontraríamos en primer lugar el cuello de botella del plegamiento antes que la fase vítreo (*Tipo IIA*) o viceversa (*Tipo IIB*), lo que determinaría la accesibilidad del estado nativo y por tanto la velocidad de plegamiento. Si se encuentra primero el cuello de botella, la proteína tendrá, como en el caso anterior, un buen número de conformaciones que le llevan hacia el estado nativo. Pero tras una relativamente rápida nucleación entrará en la fase vítreo, lo que hará que los estadios finales del plegamiento sean lentos, pues deberá encontrar caminos específicos y encontraremos conformaciones intermedias. El último escenario (*Tipo IIB*) es el más lento, porque encuentra la fase vítreo antes de entrar en el cuello de botella. Esto condicionará fuertemente los posibles caminos a los que la proteína podrá acceder en el cuello de botella, y el tiempo de plegamiento dependerá de lo que haya sucedido en la fase vítreo, por lo que su estimación es complicada. El proceso será poco cooperativo y dependerá por tanto fuertemente de las condiciones iniciales. Por este mismo motivo, cualquier pequeño cambio en la secuencia de la proteína también podría influir negativamente en su tiempo de plegamiento, por lo que sería poco robusta por ejemplo frente a mutaciones puntuales. Vamos a pasar a continuación de la perspectiva más formal a algunas aplicaciones computacionales más concretas que tratan los conceptos que hemos ido presentando.

## 11.4. Algunos ejemplos computacionales sencillos

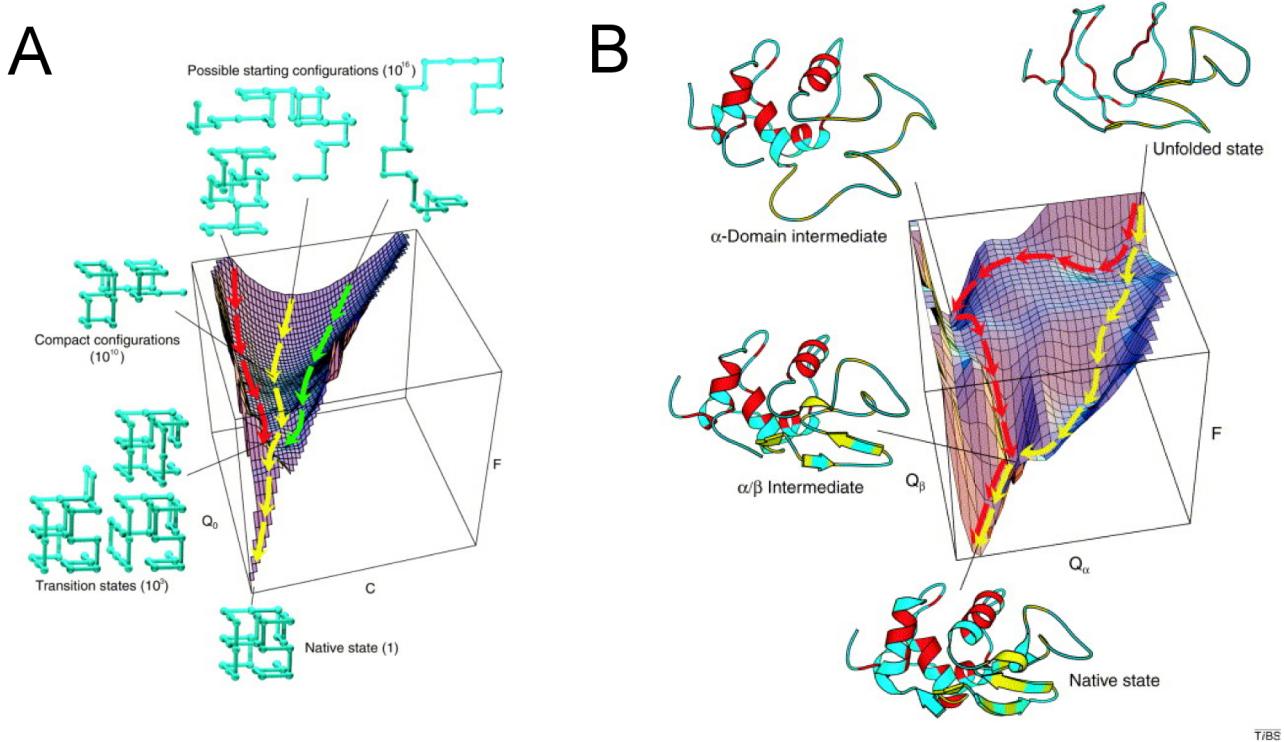
### 11.4.1. Modelos de grano grueso.

Una característica común de los modelos de sistemas con una elevada complejidad es que el primer paso siempre consiste en reducir el problema. La motivación para hacer esta reducción es variada, a veces proviene de la necesidad de hacer el problema más tratable o simplemente por intentar un abordaje basado en primeros principios. Existe además siempre un balance entre el nivel de detalle o la generalidad de la pregunta que queremos responder. La complejidad matemática aumenta tanto si queremos describir muy en detalle un proceso como si queremos que la pregunta sea muy general, por tanto el rol de la reducción de la descripción del sistema es fundamental. En la Figura 11.1 ya mostramos un ejemplo de cómo cambia la representación en función de la complejidad del modelo.

En primer lugar vamos a realizar reducciones muy importantes en la dimensionalidad del sistema a través de los llamados modelos de *grano grueso* (en inglés, *coarse grained models*). La primera simplificación consiste en reducir el número de grados de libertad de la proteína de entre todos los posibles ( $\varphi$  y  $\psi$ , ángulos dihedros de las cadenas laterales,...) al mínimo posible que nos permita tener un control suficiente sobre la estadística y la dinámica de la proteína. Hay varias opciones de simplificación posible. Por ejemplo podríamos trabajar considerando las posiciones de los carbonos  $\alpha/\gamma$  y un ángulo pseudodihedro igual a  $\varphi + \psi$  [25]. Aquí vamos a considerar en algunos apartados una simplificación incluso más fuerte, que es la propuesta por los llamados modelos de retículo (del inglés *lattice models*), en el cual los ángulos pseudodihedros están limitados a los que permite una red regular en dos o tres dimensiones (ver ilustraciones en la Figura 11.8).

La segunda simplificación importante consiste en que vamos a trabajar con un *potencial de campo medio*. Este tipo de potenciales, muy extendidos en mecánica estadística, reduce el número de componentes del campo de fuerzas de un sistema a un número menor, para lo cual realizamos promedios para algunas de estas componentes. Las cantidades seleccionadas en el promedio son aquéllas cuyas escalas espacio temporales de variación son significativamente menores que la escala de evolución del proceso que queremos observar. Por ejemplo, los grados de libertad del disolvente podemos integrarlos y añadirlos de manera efectiva a la energía interna de la macromolécula. Podemos hacer esta operación, que simplificará mucho nuestros cálculos, porque la escala de tiempo en la que el disolvente llega al equilibrio es mucho más corta que la mayoría de los movimientos de la macromolécula.

La primera simplificación que hemos propuesto sobre la representación de la proteína sigue en realidad la misma idea. En este caso, son las fluctuaciones en los ángulos de las cadenas laterales mucho menores que las de los ángulos dihedros de la cadena polipeptídica, motivo por el cual se propone reducir los ángulos a un único ángulo pseudodihedro que evolucionará a través de un potencial de campo medio. No vamos a entrar en detalle sobre cómo se integran los distintos grados de libertad en un potencial de campo medio, pero invitamos al lector interesado a consultar la referencia [2] y un ejemplo de potencial empírico en [20]. Queremos resaltar también que, a pesar de que utilizamos un potencial de campo medio para los grados de libertad del disolvente, el rol del mismo es fundamental en el proceso de plegamiento. En particular, cuando tiene lugar una interacción hidrofóbica en la macromolécula entre dos grupos, habrá moléculas de agua que se encontraban ordenadas hidratando a cada uno de ellos que pasarán al medio, aumentando así sus grados de libertad. Este hecho implica que la entropía total del sistema macromolécula más disolvente aumente, lo que permite que el proceso de plegamiento pueda tener lugar espontáneamente al desplazar la reacción en el sentido en que hay una disminución en la energía libre. Para una exposición más detallada de este punto en castellano recomendamos la referencia [13].



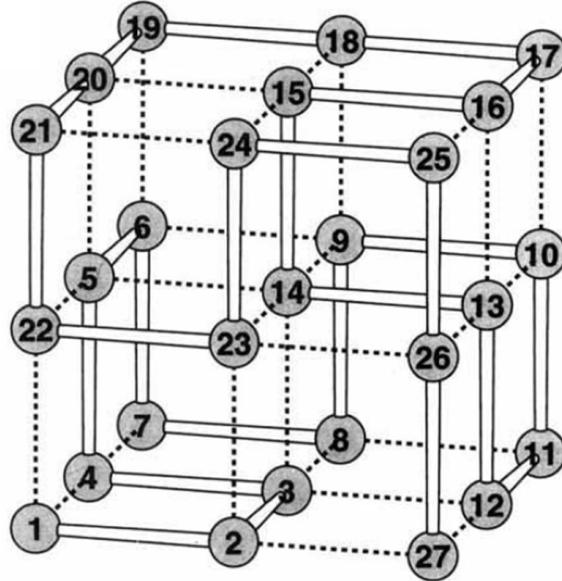
**Figura 11.8:** Ilustración en paralelo del plegamiento de un modelo 27 – *mer* (A) y de una proteína pequeña real (B). Representamos el número de conformaciones y la superficie de energía junto con algunos de los caminos seguidos en función de la región plegada, en un escenario similar al sugerido en el último ejemplo computacional que describimos debajo en la referencia [26]. La figura ha sido extraída de la referencia [9] con permiso del editor.

Lo sorprendente es que, a pesar de la simplificación, se obtienen resultados compatibles con las observaciones experimentales. Por tanto, la accesibilidad conceptual de los modelos junto con la variedad y alcance de sus resultados los hacen muy interesantes desde el punto de vista didáctico. De los diversos modelos tratados en la literatura, uno de los más utilizados es aquél en el que se considera una proteína de 27 monómeros (*modelo ‘27-mer’*) cuyo estado nativo es un cubo de lado 3, por tanto máximamente compacto. El primer problema que a uno se le ocurre es obviamente el de intentar plegar la cadena. Para ello hay que considerar dos ingredientes: una función de energía y un algoritmo de exploración de conformaciones. Estos dos ingredientes son muy generales, es decir, que complicáramos el modelo aumentaría la complejidad de ambos pero esencialmente no consideraríamos nuevos ingredientes. Cómo se pliega el modelo 27-mer y qué podemos aprender de él será sin embargo el último ejemplo que vamos a tratar, ya que es quizás el más complejo. Iremos introduciendo primero algunas funciones de energía más sencillas y veremos qué podemos aprender si consideramos la cadena ya completamente plegada.

#### 11.4.2. La transición vítrea

Un paso importante en la capacidad de responder a preguntas con el modelo 27-mer fue la resolución computacional del siguiente problema sencillo. Sabemos que una cadena de 27 monómeros se puede plegar en un cubo de lado 3, ¿pero de cuántas formas posibles se puede plegar? Este problema fue resuelto por Gutin y Shakhnovich [27], encontrando 103346 conformaciones dentro del cubo. En la

Figura 11.9 aparece representada una de estas posibles conformaciones, también llamados *caminos hamiltonianos* dentro de la red.



**Figura 11.9:** Representación de uno de los caminos posibles más compactos dentro de un cubo de lado 3. Figura extraída de [26] con permiso del editor.

La importancia de la enumeración completa reside en que nos permite considerar un espacio completo de estructuras de proteínas al que se puede sacar mucho partido. Pensemos que el orden de magnitud del número de proteínas en el *Protein Data Bank* es, a día de hoy, prácticamente el mismo (hay depositadas unas 70000 estructuras resueltas por Rayos X por ejemplo). Por tanto uno podría preguntarse si se podría trabajar con este espacio de estructuras posibles dentro de un cubo, para responder a preguntas sobre cuestiones que se observan en el espacio de proteínas reales [32] o bien como un número elevado de conformaciones de la misma proteína [27]. Hay que ser cuidadoso con el lenguaje entonces, y cuando consideremos todas las soluciones del cubo como proteínas diferentes hablaríamos de ‘*estructuras*’ mientras que hablaremos más bien de ‘*conformaciones*’ cuando considereremos dichas soluciones como distintos plegamientos posibles de la misma proteína. En cualquier caso, en los siguientes ejemplos jugaremos con ambas situaciones lo que puede crear cierta confusión en el lector a primera vista.

En primer lugar llamemos  $k$  a cada una de las soluciones dentro del cubo, y llamemos  $\alpha$  a una secuencia genérica que consideremos. Para cada secuencia, podemos proponer una energía asociada a cualquiera de las soluciones, que genéricamente llamamos  $E_k^\alpha$ . Por ejemplo, una función sencilla podría ser la siguiente:

$$E_k^\alpha = \frac{1}{2} \sum_{i,j} C_{ij}^k U^\alpha(m_i, m_j) \quad (11.11)$$

Vemos que tenemos un primer término que depende de la solución estructural concreta,  $C_{ij}^k$ , que serán los elementos de la matriz de contactos y valdrá uno si los residuos  $i, j$  están en contacto en la conformación  $k$ , o cero en caso contrario. El segundo término es una función que depende de la naturaleza de los monómeros,  $m_i$  y  $m_j$ , cuyo valor particular vendrá determinado por cada secuencia  $\alpha$  considerada. La función  $U_{ij}^\alpha = U^\alpha(m_i, m_j)$  tomará así distintos valores en función de la combinación de monómeros que interactúan.

Podemos considerar por ejemplo una secuencia que tenga simplemente aminoácidos hidrofóbicos ( $H$ ) y

polares ( $P$ ). En este caso tendríamos que los valores absolutos de energía que consideramos para nuestra secuencia Hidrofóbico-Polar serán típicamente  $|U(m_H, m_H)| > |U(m_P, m_P)| > |U(m_H, m_P)| = 0$ . Otra posibilidad podría ser el considerar que los valores de las interacciones,  $U_{ij}^\alpha$ , se obtienen al azar siguiendo una determinada distribución de probabilidad.

Consideremos esta última elección e imaginemos que generamos un número suficiente de secuencias al azar. Podemos calcular la energía para cada secuencia en cada una de las  $M = 103346$  soluciones. Lógicamente, dada una secuencia, habrá soluciones en las que consiga valores de la energía más bajos (consideramos la energía negativa) que para otras. Nos gustaría saber si desde el punto de vista termodinámico existen secuencias tales que, dada una solución estructural, dominen estadísticamente a las demás. De encontrar tales secuencias, podríamos considerar la solución asociada como una aproximación de su estructura nativa, lo que implicaría que las demás soluciones son conformaciones mal plegadas para dicha secuencia. Pero observemos aquí que una solución que se considera una conformación mal plegada para una determinada secuencia  $\alpha$ , podría ser el estado nativo para otra secuencia  $\alpha'$ , y aquí podremos encontrar cierta confusión en el glosario.

Para ver si efectivamente existen secuencias dominantes para cada solución  $k$  vamos a calcular, para cada una de las secuencias  $\alpha$  generadas al azar, sus *probabilidades de Boltzmann*:

$$p_k^\alpha = \frac{\exp\left(-\frac{E_k^\alpha}{k_B T}\right)}{Z_\alpha} \quad (11.12)$$

donde  $k_B$  es la constante de Boltzmann,  $T$  es la temperatura y  $Z$  es la función de partición que vendrá dada por:

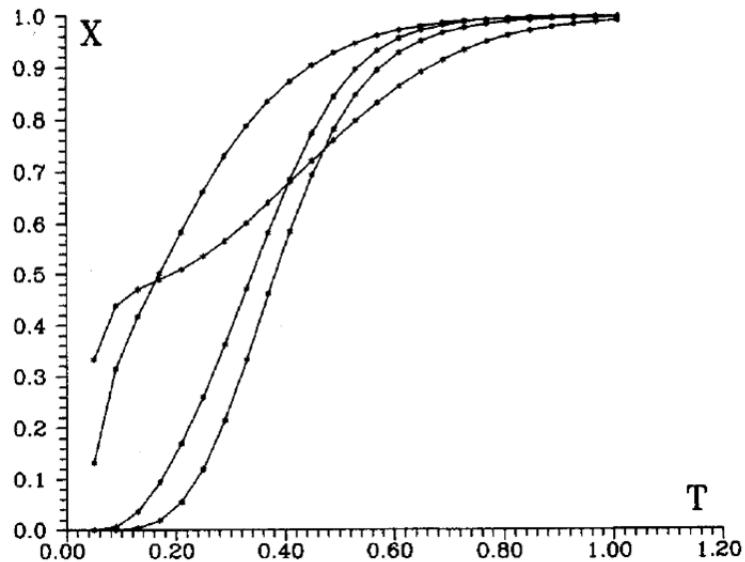
$$Z_\alpha = \sum_{k=1}^M \exp\left(-\frac{E_k^\alpha}{k_B T}\right) \quad (11.13)$$

Queremos ahora calcular una cantidad que nos determine cuáles son las conformaciones termodinámicamente más relevantes. Para ello, Gutin y Shakhnovich propusieron [27] la siguiente función:

$$X = 1 - \sum_{k=1}^M (p_k^\alpha)^2 \quad (11.14)$$

Esta función es interesante porque tiene la siguiente particularidad. Cuando la temperatura es alta, independientemente de que la secuencia encuentre conformaciones más favorables, todas las probabilidades tenderán valores muy similares. Por tanto,  $p_k^\alpha \approx 1/M$  por lo que el valor de la función  $X$  tenderá a uno. Si al descender la temperatura alguna conformación domina termodinámicamente, por ejemplo la conformación  $k = 1$ , tendremos que su probabilidad  $p_1^\alpha \approx 1$  y el valor de  $X$  tenderá a cero. Podemos darnos cuenta de que la función  $X$  se comporta como una *medida de entropía*, ya que el sumatorio se corresponde con el argumento de la entropía de Renyi de orden dos. Veamos su significado desde el punto de vista informacional. Imaginemos a nuestra secuencia fluctuando en el medio con la posibilidad de pasar de una conformación a otra y que entonces nos piden que adivinemos en qué conformación está. Si la temperatura es alta, tenemos una incertidumbre muy elevada sobre qué conformación es en la que se encuentra nuestra secuencia (mucha entropía informacional) pues todas las conformaciones tienen aproximadamente la misma probabilidad. Sin embargo, si la temperatura es baja y una conformación domina termodinámicamente no nos será difícil adivinar dónde está nuestra secuencia (baja entropía informacional). En la Figura 11.10 podemos ver el resultado de este ejercicio.

Lo primero que vemos es que, para valores altos, todas las secuencias tienen valores parecidos de la función  $X$ . Esto es de esperar porque, como hemos visto, todas las probabilidades termodinámicas tienen aproximadamente el mismo valor independientemente de la secuencia específica. Sin embargo,



**Figura 11.10:** Relación de la variable  $X$ , que podemos relacionar con una medida de entropía, con respecto de la temperatura para cuatro secuencias. Vemos que a altas temperaturas las trayectorias son prácticamente indistinguibles, mientras que a partir de cierto valor de la temperatura (alrededor de 0,6) empiezan a distinguirse, lo que podemos relacionar con la transición vítreo explicada en el modelo de energía al azar. Por debajo de este valor cada secuencia sigue una trayectoria distinta, que podemos relacionar con la discretización específica del espectro para cada secuencia en el mismo modelo. Figura extraída de [27] con permiso del editor.

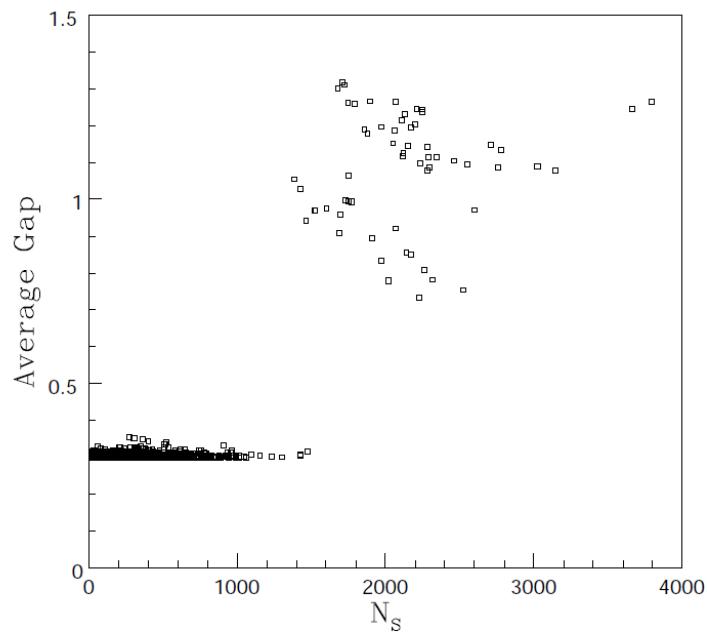
hay un valor de la temperatura, bastante similar para todas las conformaciones, por debajo del cual empieza a descender el valor de  $X$  y podemos observar comportamientos distintos en función de la secuencia hasta el punto de que, para temperaturas bajas, el comportamiento de las curvas cambia sustancialmente. Como hemos visto anteriormente en el desarrollo formal, esto es debido a que existe una única conformación que está dominando y lo que observamos es consecuencia de la relación entre la secuencia y dicha conformación, por tanto del paisaje energético.

#### 11.4.3. Diseñabilidad

Esta diferencia entre las curvas abre el siguiente interrogante, ¿de qué depende el que una secuencia tenga un espectro de energía asociado a las distintas conformaciones con determinadas características, como un determinado valor de su energía en el estado nativo o de su *gap* de estabilidad? Para responder a esta pregunta Li et. al [18] propusieron un sencillo ejercicio. Consideraron nuevamente todo el espacio de soluciones del modelo 27-mer, pero en este caso consideraron secuencias más sencillas con aminoácidos hidrofóbicos y polares, de modo que simplemente modificaríamos la función  $U_{ij}^\alpha$  en la Ecuación 11.11 respecto al caso anterior. Y a continuación determinaron cómo de diseñable era cada solución. Por *diseñabilidad* debemos entender el número de secuencias que son compatibles con una determinada estructura. En otras palabras, el número de secuencias que encuentran un mínimo absoluto de energía en una determinada estructura. Para implementarlo en nuestro modelo de juguete, lo que hicieron fue considerar un conjunto grande de secuencias, y determinar para cada una de ellas qué estructura es aquélla en la que la secuencia encuentra el valor más bajo de la energía. Contaron entonces cuántas secuencias se asociaban a cada estructura, y observaron un resultado sorprendente: la distribución era

muy asimétrica, con unas pocas soluciones (que consideraríamos distintas estructuras en este caso) en las cuales muchas secuencias encontraban su valor mínimo y muchas estructuras a las que asociaban una o muy pocas secuencias. ¿A qué se debía este curioso resultado?

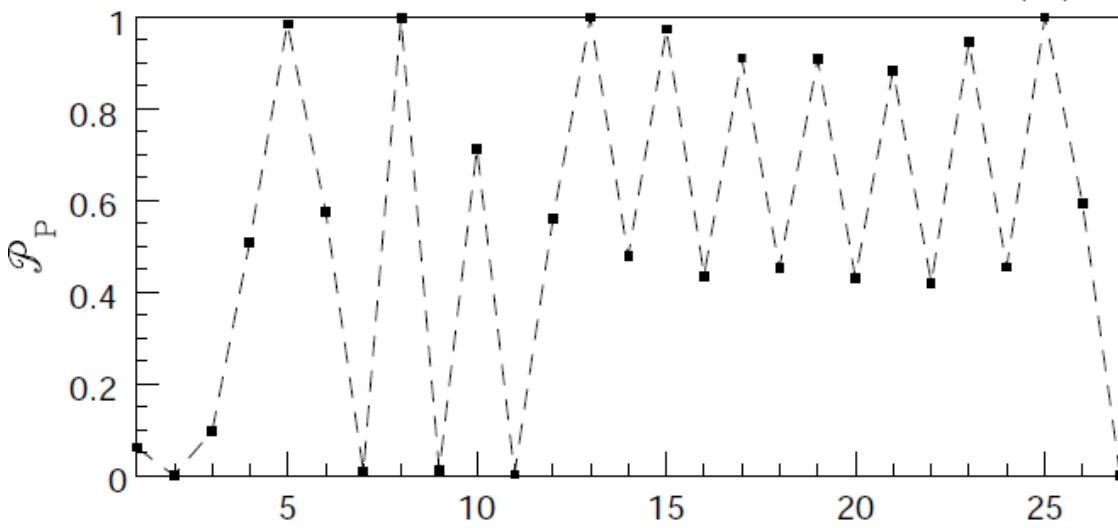
Como vimos en la Ecuación 11.10, la relación entre el *gap* de energía  $\delta E$  y la rugosidad del paisaje energético  $\Delta E$  aparecían como determinantes para optimizar la accesibilidad del estado nativo. En la referencia que discutimos, los autores midieron el *gap* de energía como la energía entre la (ahora conformación) nativa para una secuencia dada y cualquier otra conformación promediando para todas las conformaciones:  $\overline{\delta E}$ . Aquí debemos tener cuidado, porque el *gap* de estabilidad se define entre el estado plegado y la media de las conformaciones no plegadas pero aquí, tal como se ha construido el ejercicio, son conformaciones también máximamente compactas, si bien las vamos a considerar como conformaciones no plegadas. Como se muestra en la Figura 11.11, lo que observaron es que las estructuras más deseñables tenían valores de  $\overline{\delta E}$  significativamente mayores que el resto de las estructuras.



**Figura 11.11:** *Gap* de estabilidad  $\delta E$  de todas las conformaciones máximamente compactas del modelo 27-mer en función del número de secuencias que mapean a cada conformación (su deseñabilidad). Observamos que existe un salto significativo en el valor del *gap* de estabilidad a partir de cierto valor de la deseñabilidad, lo que se nos presenta como una propiedad que podría ser relevante en proteínas reales ya que les conferiría robustez mutacional. Figura extraída de la versión depositada en arXiv del artículo [18] con permiso de los autores.

Además, dichas estructuras presentaban simetrías que (con algo de imaginación) podrían ser interpretados en términos de estructura secundaria. Para entender si estas aparentes estructuras secundarias podrían tener cierta relevancia en la deseñabilidad de estas conformaciones, analizaron el conjunto de secuencias que deseñaban a las estructuras más deseñables y calcularon la probabilidad  $P_p$  de observar un aminoácido polar en una posición determinada de la secuencia. Como se muestra en la Figura 11.12, encontraron que existían posiciones con una probabilidad muy alta y otras con probabilidad nula (por tanto con probabilidad igual a uno de encontrar un aminoácido hidrofóbico). Esto sugeriría un símil entre dichas secuencias y las relaciones de homología que encontramos entre proteínas reales para las cuales existen posiciones en los que los aminoácidos están conservados y cuyas estructuras mantienen

una divergencia más lenta que la observada a nivel de secuencia. Estas relaciones serán tratadas en más detalle en las secciones de evolución y modelización de estructuras de proteínas (Capítulo 12).



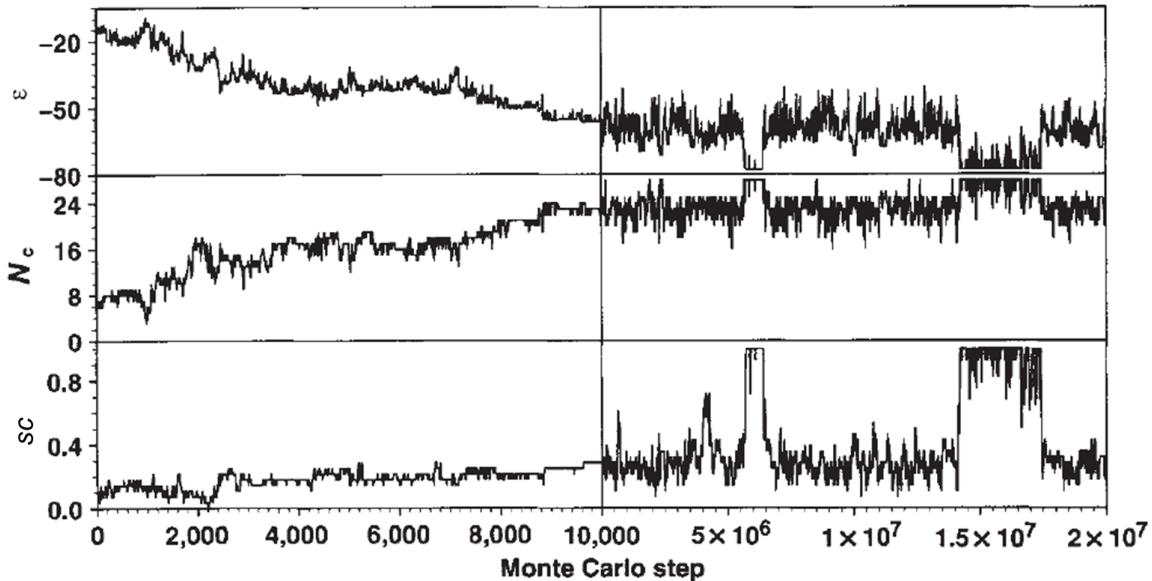
**Figura 11.12:** Probabilidad de encontrar un aminoácido polar en una posición determinada de la secuencia, calculado con el conjunto de secuencias asociadas a las estructuras más diseñables. Figura extraída de la versión depositada en arXiv del artículo [18] con permiso de los autores.

#### 11.4.4. Plegamiento de una proteína simplificada

Llegamos al final de nuestro capítulo refiriéndonos de nuevo a la pregunta que nos hacíamos inicialmente: *¿cómo puede una proteína real plegarse en un tiempo biológicamente relevante si el número de conformaciones posibles es astronómico?* Con todo lo aprendido hasta ahora, podemos intentar responder a esta pregunta con un ejercicio práctico, es decir, intentando plegar una proteína. Dado que hemos considerado hasta ahora el modelo 27 – mon, ¿seríamos capaces de plegar una secuencia elegida al azar en una conformación máximamente compacta realizando una búsqueda al azar en un tiempo razonable?

Este ejercicio es al que se enfrentaron Sali et. al, y utilizaron los ingredientes que hemos considerado hasta ahora. Generaron doscientas secuencias al azar con valores de energía de interacción elegidos de una distribución gaussiana, como en el ejemplo tratado anteriormente para explicar la transición vítrea. Aplicando la misma función de energía que en la Ecuación 11.11, realizaron una búsqueda conformacional con un algoritmo de Monte Carlo con criterio de Metrópolis [26]. En la Figura 11.13 se puede ver la monitorización de la simulación frente a distintas variables.

Su primer resultado interesante consiste en la observación de que no era posible plegar todas las secuencias en un número de pasos de Monte Carlo razonable. Sin embargo, había otras secuencias (un 15 %) que sí se plegaban en alguna de las diez simulaciones que realizaban para cada una de ellas, definiendo una *tendencia* como la fracción de esas diez simulaciones en las que la proteína alcanzaba el estado nativo. Al representar el espectro de energía de las distintas secuencias, como en la Figura 11.4, observaron que la característica más significativa de aquéllas que encontraban el estado nativo respecto de las que no lo encontraban era que el espectro de energía de las primeras tenía un mínimo de energía muy pronunciado, es decir, nuevamente el *gap* de energía era claramente mayor para aquéllas



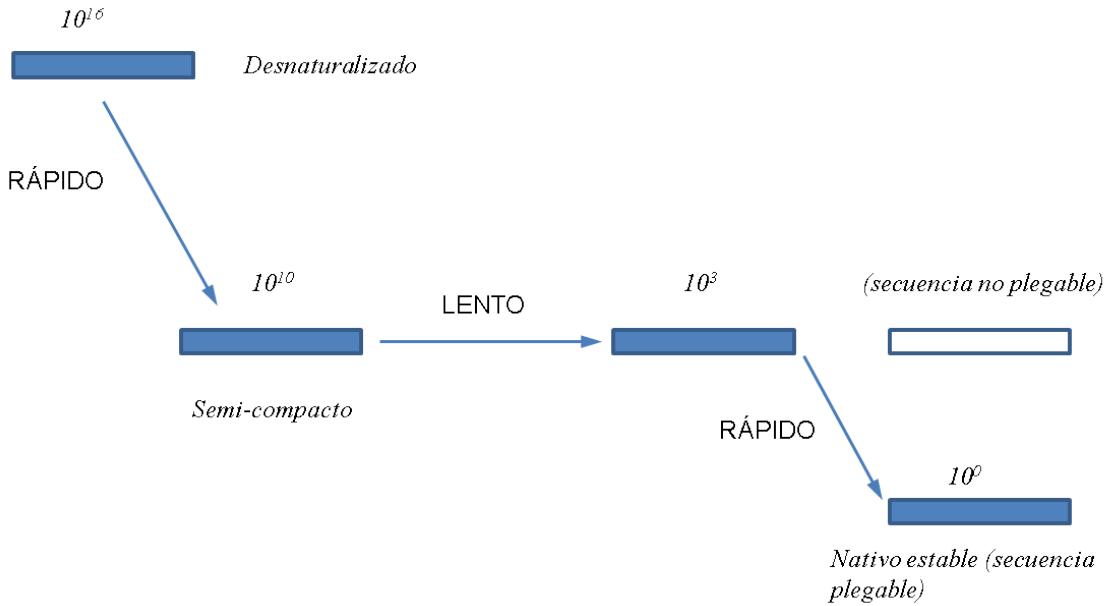
**Figura 11.13:** Ejemplo de trayectoria en la simulación por Montecarlo del plegamiento de una secuencia generada al azar en función de la energía  $\varepsilon$  (medido en unidades  $k_B T$  siendo  $k_B$  la constante de Boltzmann), del número de contactos  $N_c$ , y del solapamiento de contactos  $sc$ . Figura extraída de [26] con permiso del editor y adaptada a la notación del texto.

con tendencias mayores. En nuestro caso, la secuencia de la izquierda en la Figura 11.4 tendría una tendencia mayor a plegarse que la de la derecha.

Para interpretar estos resultados desde el punto de vista cinético, monitorizaron además el número de movimientos que eran rechazados al aplicar el algoritmo de Metrópolis a lo largo de la simulación, lo que interpretaron (con mucho cuidado) en términos de velocidad de la reacción. Como señalamos anteriormente, el solapamiento de contactos (ver sección de alineamiento de estructura de proteínas) es un buen parámetro de orden en la cinética del plegamiento. Así que se puede proponer un *modelo de tres estadios* (ver Figura 11.14) en el que relacionamos la cinética de la reacción (monitorizada por el solapamiento de contactos), la energía de las conformaciones visitadas y la velocidad de la búsqueda conformacional, y que sería compatible con los escenarios que hemos ido mostrando.

El mecanismo que describe el plegamiento consta de un *primer estadio* muy rápido en el que la proteína pasaría de un estado desplegado en el que tenemos del orden de  $10^{16}$  conformaciones a un estado semi-compacto en el que se reducen las conformaciones a  $10^{10}$ . Entonces comienza un *segundo estadio* en el que se buscan los llamados estados de transición, que son del orden de  $10^3$  estados. Éste es el cuello de botella de la búsqueda desde el punto de vista de tiempo de simulación. Porque el *tercer estadio*, si el mínimo de energía es suficientemente pronunciado, es bastante rápido una vez se encuentra un estado de transición, accediendo al estado nativo finalmente. Recordamos que una representación más explícita de la simulación y sus estadios se ha ilustrado en la Figura 11.8.

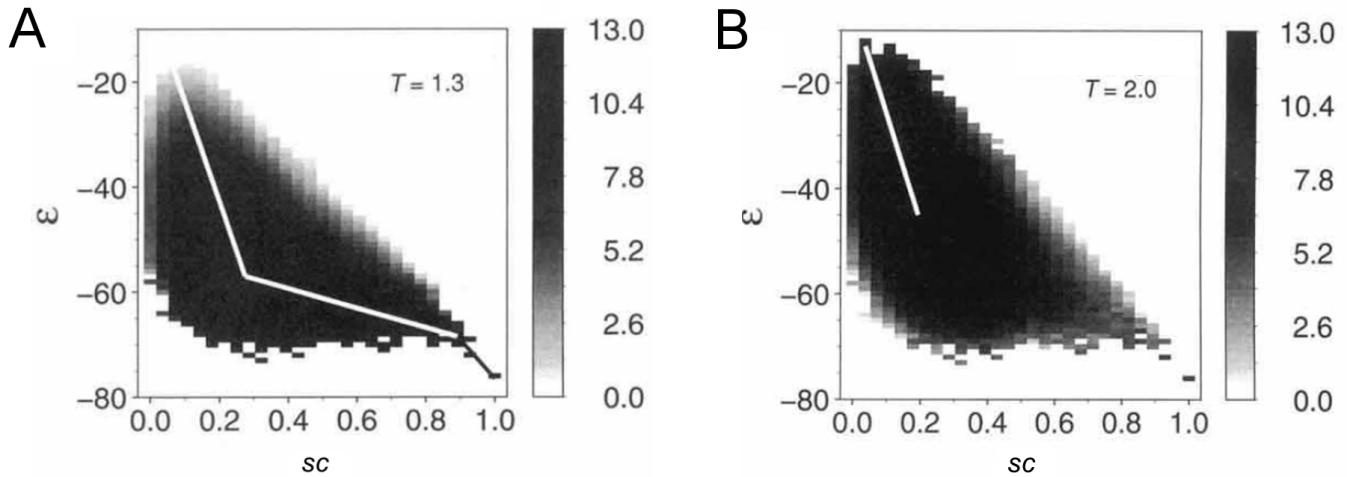
En cuanto a los tiempos de plegamiento y siguiendo el ejemplo que aparece en el artículo, en una dinámica molecular típica se observan unas tres transiciones por residuo en  $1\text{ ns}$ . Escalando los órdenes de magnitud que observamos del modelo 27-mer para una proteína de unos 80 residuos tendríamos  $10^6$  estados de transición y  $10^{18}$  estado semicompatos, por lo que nuestro cuello de botella consistiría en la exploración de  $10^{12}$  estados. Si muestreamos tres transiciones por residuo en  $1\text{ ns}$  tardaríamos alrededor



**Figura 11.14:** Representación esquemática del mecanismo de tres estadios propuesto en [26] (ver explicación en el texto).

de 4 segundos en encontrar un estado de transición, que podría ser compatible con una proteína real. Sin embargo estos órdenes de magnitud se dispararían si la proteína fuera más larga. En la discusión final del artículo haremos algunas consideraciones adicionales al respecto.

Por último queremos ilustrar el *efecto de la elección de la temperatura* en las simulaciones. En la Figura 11.15 vemos representada la densidad de estados en función de los parámetros de orden, es decir, el logaritmo de la ocupación media que se observa en cada celda a lo largo de una simulación suficientemente larga. Se representa una línea quebrada en la nube de estados, que indica una trayectoria típica observada a dicha temperatura. Para temperatura baja, se observa cómo se llega rápidamente a una conformación semi compacta tras lo cual la pendiente disminuye y va lentamente progresando aumentando el solapamiento de contactos hasta que, rápidamente, precipita sobre el estado nativo. Sin embargo no se observa la misma situación si la temperatura aumenta, ya que la cadena llega rápidamente a un estado semicompacto y se va a quedar preferentemente en dicha región, ya que los estados desnaturalizados son entrópicamente más favorables. Aunque eventualmente podría acceder al estado nativo, en estas condiciones no es estable y retornaría a una conformación semi compacta.



**Figura 11.15:** Fracción de estados ocupados durante una simulación larga de Montecarlo monitorizados por los parámetros de orden de energía  $\varepsilon$  y solapamiento de contactos  $sc$ , para dos temperaturas distintas. Figura extraída de [26] con permiso del editor y adaptada a la notación del texto.

## 11.5. Discusión

A lo largo de este capítulo sobre plegamiento de proteínas hemos buscado el proporcionar al lector una idea general de la *complejidad del problema de plegamiento de proteínas*, haciendo hincapié en la necesidad y potencialidad que revisten los modelos simplificados a la hora de responder a preguntas que, o bien por su complejidad o bien por su generalidad, no podrían ser abordadas de otro modo. Esto nos ha llevado a presentar algunos conceptos críticos como el paisaje energético y la posibilidad de relacionar propiedades específicas del mismo como la rugosidad y el *gap* de estabilidad, con propiedades medibles experimentalmente como las temperaturas de plegamiento y vitrificación.

Este análisis nos ha permitido entender la *resolución de la paradoja de Levinthal* basándonos en propiedades del paisaje energético, que nos aparece ahora más bien como un artefacto conceptual. Además hemos identificado una propiedad que emerge en los modelos simplificados, el *gap* de estabilidad, como propiedad relevante para el control cinético y por tanto para el plegamiento en tiempos biológicamente viables. Esta propiedad también nos abre la puerta a proponer modelos evolutivos, ya que sería consistente con la visión en la que aquéllas estructuras funcionales con elevada robustez mutacional son seleccionadas.

Sin embargo *debemos de ser críticos con la validez y generalidad de los resultados encontrados*. Por ejemplo, el modelo de tres estadios que hemos explicado podría ser apropiado para algunas proteínas pequeñas pero sería inviable para proteínas grandes. Pero su valor, como todo conocimiento científico, reside más que en ser un resultado que consideremos verdadero en las explicaciones que rechaza. Por ejemplo, como investigadores computacionales, nos sugiere que más que trabajar en el desarrollo de algoritmos de búsqueda conformacional el reto está en encontrar funciones de energía más precisas, ya que en el ejercicio que hemos mostrado nos ha bastado un muestreo al azar con un algoritmo de Montecarlo para plegar nuestras proteínas de juguete. Sin embargo, para proteínas reales queda mucho camino por recorrer lo que probablemente requerirá de ideas originales.

Por ejemplo, como dijimos al principio del capítulo, uno de los problemas de interés desde el punto de vista del plegamiento es la modelización de estructura de proteínas, que no hemos tratado. Pero merece la pena señalar la novedosa propuesta que, en el contexto de la modelización, ha desarrollado

el laboratorio de David Baker en la Universidad de Washington, en donde surgió la idea de diseñar un videojuego para realizar la búsqueda conformacional. Ayudados por una función de energía que evalúa la calidad del plegamiento (y que determina la puntuación en el videojuego) los aficionados al mismo son invitados a aprender a plegar proteínas y a enfrentarse con proteínas con plegamientos desconocidos. La hipótesis es que *la intuición humana puede ser capaz de encontrar la solución óptima mejor que otros algoritmos de búsqueda* (siempre apoyados por una función de energía) sobre todo en aquellas estructuras con plegamientos complejos. Y, sorprendentemente, han conseguido encontrar el plegamiento de una proteasa retroviral utilizando los resultados obtenidos por los jugadores [15], la mayoría de ellos con poco o ningún conocimiento en bioquímica.

Por último debemos de tener en cuenta que *en los procesos biológicos in vivo hay multitud de factores que no estamos teniendo en cuenta* en nuestro análisis del plegamiento y, por extensión, en la evolución de estructura de proteínas. Ingredientes tales como el tamaño de la población en que el gen asociado a la proteína está evolucionando, las condiciones del medio (que pueden inducir sesgos mutacionales) o el rol de las chaperonas, serían determinantes y nos llevan a una visión del plegamiento en donde la estabilidad del estado nativo es más bien marginal [10]. De este modo deberíamos de implementar definiciones más complejas de lo que entedemos por estabilidad, como el balance entre estabilidad frente a desplegamiento (*unfolding stability*) o a mal plegamiento (*misfolding stability*), conceptos que ya se están implementando en modelos evolutivos computacionales [19, 30]. Estos argumentos plantean la creciente necesidad de pensar en el problema de la evolución de proteínas desde una perspectiva más amplia [23], pero ya dotados de una serie de herramientas conceptuales mínimas para abordar nuevos ingredientes.

Por tanto, *el reto continúa abierto en todos los frentes*, desde la comprensión de los conceptos más generales que hemos propuesto en el presente capítulo, a las ideas originales como el videojuego Foldit<sup>4</sup> y el creciente interés por los entornos colaborativos como la iniciativa WeFold<sup>5</sup>). Y por último, no nos olvidemos de la necesidad de mirar hacia el *nuevo reto conceptual y computacional que suponen las proteínas intrínsecamente desestructuradas*.

---

<sup>4</sup>Foldit. <https://fold.it>

<sup>5</sup>WeFold. <http://www.wefold.org>



## 11.6. Bibliografía

- [1] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *PNAS*, 47(9):1309–1314, Sept. 1961. PMID: 13683522 PMCID: PMC223141.
- [2] C. L. Brooks, M. Karplus, and B. M. Pettitt. *Advances in Chemical Physics, Proteins: A Theoretical Perspective of Dynamics, Structure, and Thermodynamics*. John Wiley & Sons, July 1990.
- [3] J. D. Bryngelson, J. N. Onuchic, N. D. Soccia, and P. G. Wolynes. Funnels, pathways, and the energy landscape of protein folding: A synthesis. *Proteins: Structure, Function, and Bioinformatics*, 21(3):167–195, Feb. 1995.
- [4] J. D. Bryngelson and P. G. Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Sciences*, 84(21):7524–7528, Nov. 1987.
- [5] J. D. Bryngelson and P. G. Wolynes. Intermediates and barrier crossing in a random energy model (with applications to protein folding). *J. Phys. Chem.*, 93(19):6902–6915, 1989.
- [6] L. Cruzeiro Hansson. Protein folding: thermodynamic versus kinetic control. *Journal of biological physics*, 27:S6, 2001.
- [7] B. Derrida. Random-energy model: Limit of a family of disordered models. *Physical Review Letters*, 45(2):79–82, July 1980.
- [8] K. A. Dill and H. S. Chan. From levinthal to pathways to funnels. *Nature Structural & Molecular Biology*, 4(1):10–19, Jan. 1997.
- [9] A. R. Dinner, A. Šali, L. J. Smith, C. M. Dobson, and M. Karplus. Understanding protein folding via free-energy surfaces from theory and experiment. *Trends in Biochemical Sciences*, 25(7):331–339, July 2000.
- [10] C. M. Dobson. Protein folding and misfolding. *Nature*, 426(6968):884–890, Dec. 2003.
- [11] M. M. Garcia-Mira, M. Sadqi, N. Fischer, J. M. Sanchez-Ruiz, and V. Muñoz. Experimental identification of downhill protein folding. *Science*, 298(5601):2191–2195, Dec. 2002.
- [12] R. A. Goldstein, Z. A. Luthey-Schulten, and P. G. Wolynes. Optimal protein-folding codes from spin-glass theory. *Proceedings of the National Academy of Sciences*, 89(11):4918–4922, June 1992.
- [13] J. R. Grigera. La física de la desnaturalización de proteínas en frío: (o como cocinar un huevo a temperatura ambiente). *Revista iberoamericana de física*, 6(1):27–33, 2010.
- [14] M. Karplus. Behind the folding funnel diagram. *Nature Chemical Biology*, 7(7):401–404, July 2011.
- [15] F. Khatib, F. DiMaio, F. C. Group, F. V. C. Group, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zabranska, I. Pichova, J. Thompson, Z. Popović, M. Jaskolski, and D. Baker. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural & Molecular Biology*, 18(10):1175–1177, 2011.
- [16] J. Kubelka, J. Hofrichter, and W. A. Eaton. The protein folding speed limit. *Current Opinion in Structural Biology*, 14(1):76–88, Feb. 2004.
- [17] C. Levinthal. How to Fold Graciously. In J. T. P. Debrunner and E. Munck, editors, *Mossbauer Spectroscopy in Biological Systems: Proceedings of a meeting held at Allerton House, Monticello, Illinois*, pages 22–24. University of Illinois Press, 1969.
- [18] H. Li, R. Helling, C. Tang, and N. Wingreen. Emergence of preferred structures in a simple model of protein folding. *Science (New York, N.Y.)*, 273(5275):666–669, Aug. 1996. PMID: 8662562.
- [19] R. Méndez, M. Fritzsche, M. Porto, and U. Bastolla. Mutation bias favors protein folding stability in the evolution of small populations. *PLoS Comput Biol*, 6(5):e1000767, May 2010.
- [20] S. Miyazawa and R. L. Jernigan. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3):534–552, Mar. 1985.
- [21] V. Muñoz. Thermodynamics and kinetics of downhill protein folding investigated with a simple statistical mechanical model. *International Journal of Quantum Chemistry*, 90(4-5):1522–1528, 2002.
- [22] J. N. Onuchic and P. G. Wolynes. Theory of protein folding. *Current Opinion in Structural Biology*, 14(1):70–75, Feb. 2004.

- [23] C. Pál, B. Papp, and M. J. Lercher. An integrated view of protein evolution. *Nature Reviews Genetics*, 7(5):337–348, May 2006.
- [24] V. Pande, A. Grosberg, and T. Tanaka. Statistical mechanics of simple models of protein folding and design. *Biophysical Journal*, 73(6):3192–3210, Dec. 1997.
- [25] W. L. Peticolas and B. Kurtz. Transformation of the  $\phi$ - $\psi$  plot for proteins to a new representation with local helicity and peptide torsional angles as variables. *Biopolymers*, 19(6):1153–1166, Feb. 2004.
- [26] A. Šali, E. Shakhnovich, and M. Karplus. How does a protein fold? , *Published online: 19 May 1994; | doi:10.1038/369248a0*, 369(6477):248–251, May 1994.
- [27] E. Shakhnovich and A. Gutin. Enumeration of all compact conformations of copolymers with random sequence of links. *The Journal of Chemical Physics*, 93(8):5967–5971, Oct. 1990.
- [28] N. D. Soccia and J. N. Onuchic. Kinetic and thermodynamic analysis of proteinlike heteropolymers: Monte carlo histogram technique. *The Journal of Chemical Physics*, 103(11):4732–4744, Sept. 1995.
- [29] H. Taketomi, Y. Ueda, and N. Gō. Studies on protein folding, unfolding and fluctuations by computer simulation. i. the effect of specific amino acid sequence represented by specific inter-unit interactions. *International journal of peptide and protein research*, 7(6):445–459, Dec. 1974.
- [30] D. M. Taverna and R. A. Goldstein. Why are proteins marginally stable? *Proteins: Structure, Function, and Bioinformatics*, 46(1):105–109, 2002.
- [31] A. M. Vasilyev. *Introduction to Statistical Physics*. Imported Pubn, Sept. 1984.
- [32] K. B. Zeldovich, P. Chen, B. E. Shakhnovich, and E. I. Shakhnovich. A first-principles model of early evolution: Emergence of gene families, species, and preferred protein folds. *PLoS Comput Biol*, 3(7):e139, July 2007.

## Capítulo 12

# Evolución de estructura de proteínas

*Alberto Pascual-García*

### 12.1. Introducción

En la presente sección vamos a estudiar el impacto de los distintos eventos evolutivos que ocurren desde el punto de vista genético, en la evolución de las estructuras de proteínas. *Los distintos eventos evolutivos tendrán un impacto en la estructura de la proteína* que, al verse sometida al continuo proceso de selección, podrá ser incorporado con mayor o menor probabilidad según cómo afecte a las posibilidades de reproducción del individuo. Cuantificar este impacto es una tarea difícil entre otras cosas porque, como vimos en el Capítulo 11 y volveremos a ver en el presente capítulo, las estructuras de proteínas reales presentan cierta *robustez mutacional* que les permite acumular mutaciones (e incluso eventos más dramáticos como inserciones o delecciones) sin que su función se vea afectada. Así que para conseguir nuestro objetivo intentaremos inferir el impacto de los distintos eventos a través del análisis del espacio de todas las estructuras de proteínas conocidas, identificando en particular cuáles se nos presentan como dominantes en la evolución de las mismas.

### 12.2. Origen de nuevas proteínas

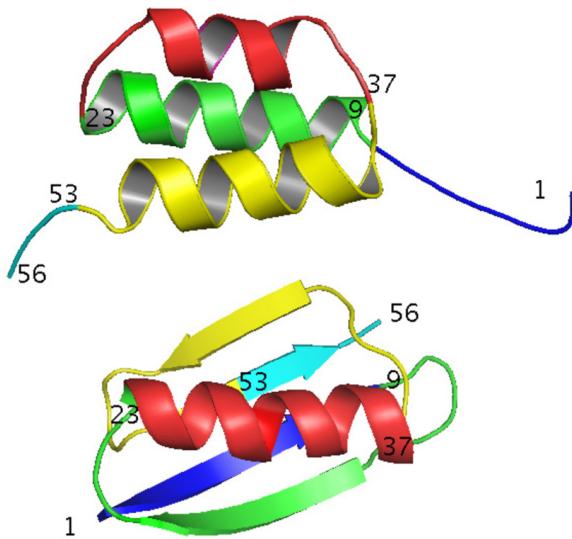
Para comenzar, vamos a recordar las *cuatro maneras principales de replicación de genes* que codifican las proteínas, si bien recomendamos al lector que tenga en cuenta en primer lugar los contenidos del capítulo de filogenia y evolución molecular (Capítulo 9).

- *Transferencia vertical.* Son aquellos genes que se transfieren desde progenitores a su descendencia. Los genes relacionados por este proceso se denominan *ortólogos*.
- *Transferencia horizontal.* Genes que se transfieren desde otro organismo de la misma o distinta especie. Como ejemplos podríamos tener la conjugación bacteriana, el intercambio entre mitocondria y núcleo o entre bacterias y virus a hospedadores eucarióticos y viceversa.
- *Duplicación génica.* Copia de un gen dentro de un mismo genoma acompañada de una diferenciación funcional, lo que se denomina neofuncionalización. Genes relacionados de este modo se denominan *parálogos*. Si la diferenciación funcional no tiene lugar, no existe presión selectiva para conservar las dos copias y uno de los dos genes se convierte en un *pseudo-gen* y se pierde.

- *Fusión génica*. A través de la fusión de dos o más fragmentos de estructura super-secundaria.

Dos genes relacionados a través de alguno de estos mecanismos evolutivos, se verán sometidos a una nueva serie de eventos que harán que la similaridad entre dichos genes disminuya. Como hemos adelantado, si los genes son ortólogos, normalmente reflejan el resultado de una *deriva aleatoria* (del inglés *random drift*) en la que ambos genes están sometidos a una presión selectiva para conservar la función de la línea ancestral. Pero si los genes son parálogos reflejaría más bien la adaptación de las distintas copias para una *diversificación de funciones*. Para una revisión integrada de los mecanismos que influyen en la evolución de proteínas recomendamos al lector la referencia [25].

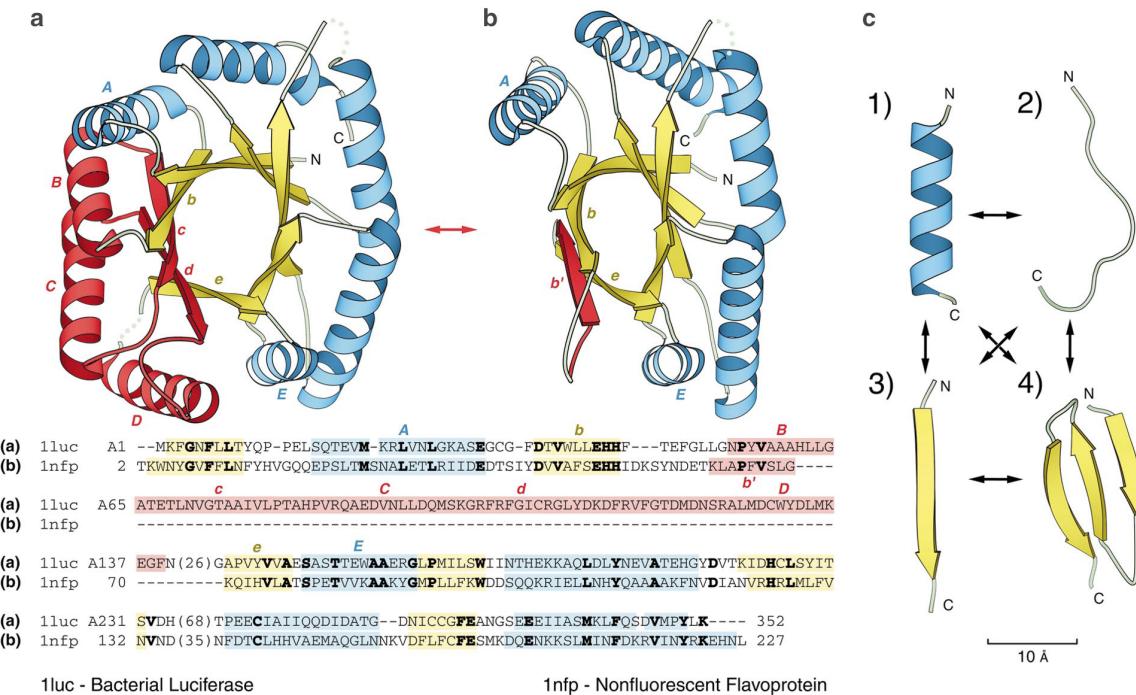
Nos queremos preguntar qué eventos son dominantes en la evolución de las estructuras de proteínas. Es decir, qué eventos determinan esencialmente la similaridad (o disimilaridad) entre las proteínas a nivel, ya no de su secuencia, sino de su estructura. Como hemos comentado en la sección de plegamiento de proteínas, la estructura juega un papel determinante en el correcto funcionamiento de la proteína, y los efectos de *los diferentes eventos evolutivos posibles sobre las estructuras pueden ser muy distintos*. Por ejemplo, una *mutación puntual* podría prácticamente no afectar a la estructura si la mutación es sinónima o codifica para un aminoácido con un rol estructural similar. Como casi siempre en biología hay excepciones. Por ejemplo se han descrito polimorfismos silenciosos que dan lugar a distintas estructuras y funciones [18], lo cual va en contra del (mal llamado dogma) central de la biología (Sección 7.2). Lo que está claro es que esperamos cambios más dramáticos si existen *inserciones y delecciones*. Por ejemplo, podría cambiar por completo la estructura si se inserta un único nucleótido, ya que al traducirse el código genético en tripletes nos cambiaría el marco de lectura y por tanto su traducción (ver Sección 7.4). Además se han presentado ejemplos como en [1] en los que un número mínimo de mutaciones pueden cambiar tanto la estructura como la función como se ilustra en la Figura 12.1.



**Figura 12.1:** La proteína de *Streptococcus G* presenta dos tipos de proteínas ( $G_A$  y  $G_B$ ) que se unen a proteínas del suero sanguíneo, con estructuras distintas. En el estudio generan mutantes de cada uno de los grupos y consiguen que mantengan la estructura y función, a pesar de que las separan tan solo tres mutaciones [1]. Los números indican las posiciones de algunos aminoácidos, los mutados están en las posiciones 20, 30 y 45. Figura reproducida con permiso de los autores.

Continuando con el ejemplo, sabemos que la relación entre el número de sustituciones con respecto al de inserciones y delecciones varía entre alrededor de 5 veces más sustituciones hasta incluso 60 veces más [5], en función del organismo y de si es o no codificante la región estudiada. Pero, *aunque la frecuencia*

de las inserciones y delecciones es menor, su efecto sobre la estructura puede ser mucho más dramático. En la Figura 12.2 podemos ver el efecto de uno de los eventos evolutivos más dramáticos que se puede encontrar en la literatura, entre la luciferasa bacteriana y su homóloga, la flavoproteína no fluorescente que codifica el gen luxF de Photobacterium, con una delección de aproximadamente 90 residuos descrita en [13].

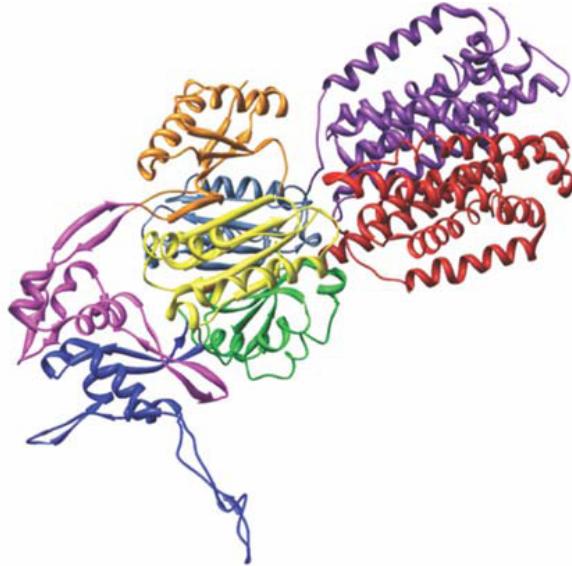


**Figura 12.2:** Comparación estructural entre la luciferasa bacteriana (izquierda) y la flavoproteína no fluorescente (derecha). Son dos proteínas homólogas a pesar de que su divergencia en secuencia es importante, como se ve en el alineamiento, con una (presumiblemente) delección de 90 residuos. A la derecha se muestran cambios típicos en estructura secundaria observados [13]. Figura reproducida con permiso del editor.

Cambios tan dramáticos ponen en jaque nuestra habilidad para modelar estructura de proteínas. Por este motivo cabe preguntarse qué escenario esperaríamos encontrar en cada caso en función del tipo y frecuencia de los distintos eventos evolutivos. Es decir, *si consideráramos todas las proteínas conocidas, ¿cómo de parecidas son globalmente?, ¿tienen similaridades locales?* Si encontramos similaridades, *¿cómo lo debemos interpretar?, ¿es este parecido debido principalmente a que están relacionadas evolutivamente?, o es en cambio una consecuencia de que realizan una función similar aunque no tengan un origen evolutivo común (como veremos, casos de convergencia funcional)*. El entender la *relación secuencia-estructura-función* bajo la luz de la evolución, nos permitirá proponer métodos para predecir la función de una nueva proteína, utilizando su información en secuencia y estructura.

Por último, es importante destacar que vamos a trabajar con dominios estructurales, como se muestra en la Figura 12.3. Un *dominio estructural* lo podríamos definir como la unidad mínima en que se puede descomponer una cadena de aminoácidos de modo que sea autónoma desde el punto de vista evolutivo. Es decir, esperamos que sea una región conservada con una estabilidad termodinámica y accesibilidad cinética suficiente para realizar una determinada función. De modo que dada una proteína larga el primer paso es el intentar descomponerla en dominios, pues cada uno de ellos sería en principio susceptible de evolucionar autónomamente. Un reto importante consiste en entender cómo estos dominios han coevolucionado, es decir, de qué modo el hecho de que se encuentran en la misma proteína ha

influido en la evolución de cada uno como estructuras autónomas. La descomposición en dominios es un problema bioinformático complejo en tanto en cuanto no es sencillo determinar criterios objetivos para realizar una descomposición precisa. Remitimos al lector interesado a consultar [15].



**Figura 12.3:** Ejemplo de descomposición en dominios de la cadena A de la proteína 1oy8 según la clasificación estructural de CATH [33]. Figura reproducida con permiso del editor.

### 12.3. Divergencia estructural gradual: especiación y duplicación génica

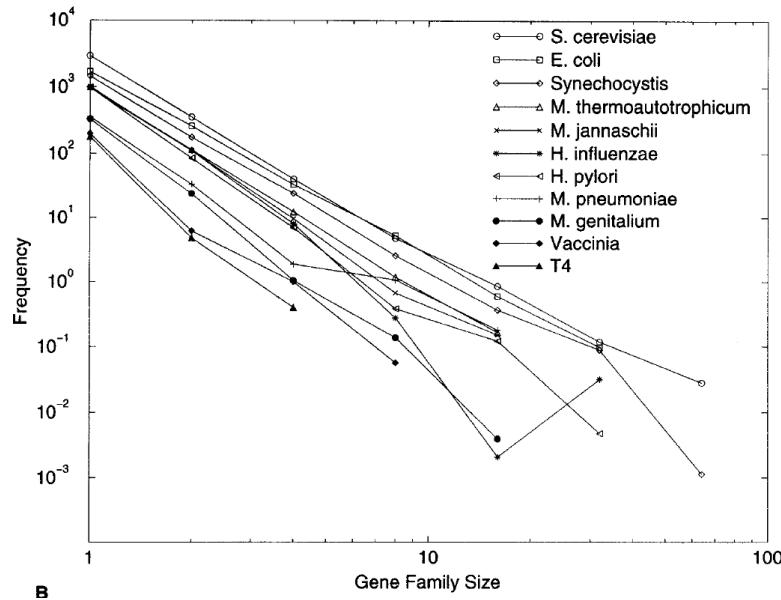
Comencemos considerando dos estructuras expresadas por genes homólogos, con una similaridad relevante. Su similaridad irá disminuyendo, y diremos por tanto que ambas estructuras divergen a lo largo del tiempo, a medida que los distintos eventos evolutivos se van acumulando en cada una de ellas. Si las *proteínas* son *ortólogas* ha habido un evento de especiación y dicha divergencia es debida esencialmente a la deriva génica. En ese sentido, *la acumulación de mutaciones es proporcional al tiempo que ha transcurrido desde la especiación*, y esa proporcionalidad nos permite decir que el modo en que las proteínas disminuyen su similaridad es gradual, es decir, sin brusquedades.

En cambio si las *proteínas* son *parálogas*, ha existido una duplicación génica y, tras este evento, los genes que no se transforman en pseudo-genes experimentan una aceleración en el ritmo de sustitución debido a una reducción en la *presión selectiva sobre una de las copias* y, quizás más importante, cambios en la regulación que modifican fuertemente la expresión génica. Estos genes duplicados divergerán por tanto en secuencia, estructura y eventualmente en función biológica. Sin embargo, en muchos casos las actividades bioquímicas se conservan (no necesariamente la función biológica) y, a pesar de que el ritmo de sustitución ha aumentado, podemos considerar en muchos casos que la divergencia es también aproximadamente gradual.

En general por tanto, el que la divergencia sea consistente con el reloj molecular, implica que pasado un tiempo  $t$  encontraríamos grupos de genes relacionados entre sí más fuertemente que con otros grupos de genes. Llamaremos a cada uno de estos grupos fuertemente relacionados *familias de genes*. Estas familias se obtienen “cortando” el árbol filogenético a un determinado umbral que, como hemos

apuntado, estará relacionado con el tiempo de divergencia. A este “corte” le llamaremos en adelante una *partición del espacio de genes*, ya que lo partimos *en distintas familias*. Y para cada una de las familias de proteínas, no existe discusión sobre su *origen monofilético*, es decir, sobre la *existencia de un ancestro común a todas las proteínas que pertenecen a la familia*.

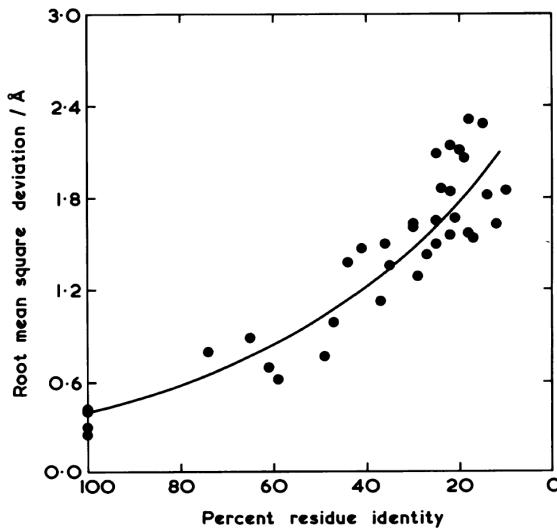
Una propiedad interesante de la partición que contribuyó a determinar el origen monofilético de las familias de genes es la siguiente. Para cada una de las familias de genes, podemos contar el número de genes que contiene. Y de este ejercicio, podemos conocer la probabilidad de que una familia contenga un determinado número de genes, es decir, su *distribución de probabilidad*. Como vemos en la Figura 12.4 [17], se encontró que era una distribución muy apuntada hacia la derecha (tengamos en cuenta que la figura está en escala logarítmica), donde las familias no tienen un tamaño típico. Esta distribución sigue una ley de potencias y también es conocida como *distribución libre de escala* [35]. Estas distribuciones son importantes porque se han propuesto varios modelos que las generan que pueden ser compatibles con los escenarios evolutivos que consideramos, además de ser distribuciones encontradas frecuentemente en sistemas muy distintos, no solamente biológicos. En el capítulo de biología de sistemas recomendamos al lector dirigirse a la sección de redes complejas donde se explica la distribución y alguno de los modelos evolutivos asociados (Sección 19.2). Pero de momento nos basta entender que estos modelos son compatibles con el escenario en el que la duplicación génica junto con la divergencia gradual es dominante. Además, el hecho de que sea libre de escala, nos dice que *el escenario evolutivo dominante no ha cambiado a lo largo de la escala temporal*.



**Figura 12.4:** Histograma del número de familias con un tamaño determinado. El tamaño viene dado por el número de genes que contiene dicha familia. La gráfica muestra el histograma para distintos organismos. Aunque la pendiente de las curvas cambia no lo hace el tipo de distribución, que se ajusta muy bien siempre a una ley de potencias. Reproducido con permiso del editor.

Esta observación abrió la puerta a preguntarse si este tipo de partición se encontraría también si hiciéramos el mismo ejercicio en el espacio de estructura de proteínas. Gracias a un artículo seminal debido a Chothia y Lesk [6] hace ya más de 25 años, y que abrió la puerta a comprender cuantitativamente las relaciones entre secuencia y estructura, sugería que así debería ser. En este artículo *observaron que la raíz de la desviación cuadrática media entre globinas (RMSD, ver eq:ProtEvolucion:RMSD) divergía regularmente con el número de sustituciones* (ver Figura 12.5). Esta observación nos invitaría a pensar

que es posible construir también “familias” de estructuras a partir de medidas de similaridad entre ellas, es decir, a establecer una clasificación. Esta fue la propuesta natural a la luz de las observaciones y por tanto la primera que vamos a desarrollar. Posteriormente vamos a ver otros escenarios en donde se sugiere la posible dominancia de otros mecanismos evolutivos como consecuencia de otras observaciones, también interesantes.



**Figura 12.5:** Relación entre la divergencia en secuencia, cuantificada mediante su porcentaje de identidad, y la divergencia en estructura cuantificada con la raíz de la desviación cuadrática media de sus posiciones espaciales. Ver las secciones de evolución en secuencia y alineamiento estructural para más detalles sobre estas medidas. Se observa que la divergencia en estructura para cada par de proteínas es proporcional a su divergencia en secuencia y por extensión al tiempo de divergencia con respecto a su ancestro común. Figura reproducida con permiso del editor.

### 12.3.1. Clasificación de estructura de proteínas

Supongamos que las estructuras divergen gradual y proporcionalmente con el tiempo y que sabemos construir una medida que, teniendo en cuenta dos estructuras cualesquiera,  $a$  y  $b$ , nos de una distancia indicativa de su disimilaridad estructural  $d(a, b)$ . Esto lo haríamos con un alineamiento de estructuras, que se puede consultar en la Capítulo 10. Si estas condiciones se cumplen, podemos deducir un resultado interesante. Imaginemos que consideramos dos proteínas homólogas  $a$ ,  $b$  y una tercera proteína  $c$  que considerarmos como referencia fuera del grupo. Si la acumulación de mutaciones, y por tanto la divergencia entre  $a$  y  $b$ , es constante a lo largo del tiempo, la siguiente relación se debe verificar aproximadamente  $d(a, c) \approx d(b, c)$ . Que se cumpla esta relación implica que estamos en situación de construir un árbol sin ambigüedades. El que podamos construir un árbol significa que podríamos además determinar un umbral por debajo del cual (si utilizamos como medida una distancia o disimilaridad, por encima si es una similaridad) las proteínas están globalmente relacionadas desde el punto de vista estructural formando conjuntos bien separados, es decir, podríamos construir una clasificación. A cada uno de estos *conjuntos de estructuras con una similaridad global sustancial* se les denomina *fold* (plegamiento).

No debemos confundir el plegamiento de una única proteína con este concepto, ya que el término en inglés es el mismo. Aquí de algún modo nos estamos refiriendo a los motivos estructurales que comparten todas las proteínas que están dentro de uno de estos conjuntos que llamamos *folds*. Es decir, es una

especie de idea “platónica” a través de la cual se representa a todas las proteínas de dicho conjunto tras reducir su variabilidad a la estructura mínima común. Y notamos también que, aunque esperamos que estos *folds* estén relacionados con las familias obtenidas mediante el análisis de las secuencias de genes, no tienen por qué ser necesariamente las mismas. De hecho, veremos más adelante que existen argumentos para esperar diferencias entre los conjuntos que se obtienen por análisis estructural respecto de los obtenidos por análisis de secuencias.

Las observaciones que hemos indicado motivaron la posibilidad de que se pudieran definir *folds*, y por extensión la búsqueda de *métodos para la creación de clasificaciones estructurales de proteínas*. Las clasificaciones más relevantes son las siguientes:

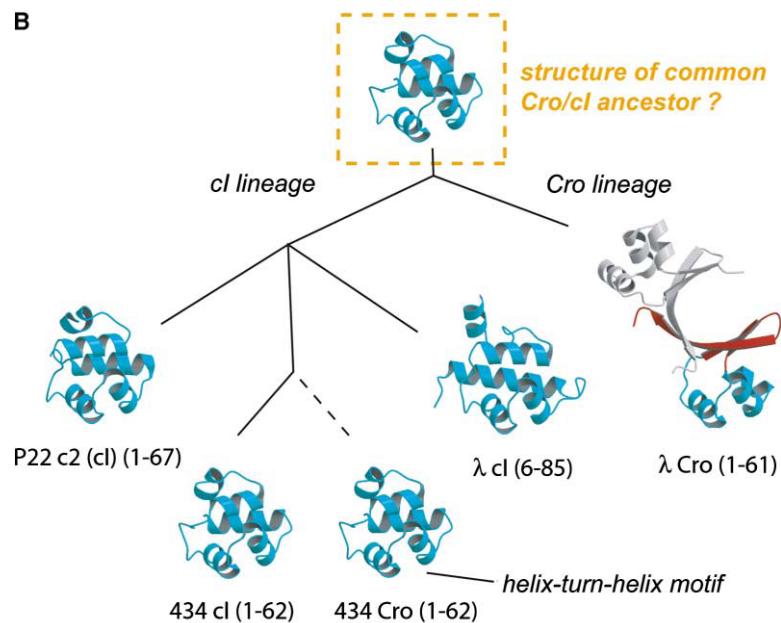
- *SCOP (Structural Classification Of Protein structures)* [22]. Ésta ha sido típicamente la clasificación de referencia. Es una clasificación en principio manual pero que, dado el elevado número de proteínas que hay que clasificar anualmente, es de esperar que utilicen herramientas automáticas de preprocessamiento (si bien es cierto que su actualización es la más lenta). Su éxito reside precisamente en el análisis experto de las estructuras, lo cual es al mismo tiempo una fuente de subjetividad. Se ha observado [27] que utilizan criterios más allá de los puramente estructurales para su clasificación, con un fuerte énfasis en las relaciones evolutivas. Por este motivo sí es de esperar que exista un acuerdo razonable entre los *folds* definidos en esta clasificación y los obtenidos por análisis de las secuencias pero la consistencia estructural de los grupos es discutible.
- *CATH (Class, Architecture, Topology, Homologous superfamily)* [24]. Esta clasificación es semi-automática. Realizan un preprocessamiento que incluye alineamientos estructurales automáticos junto con un algoritmo de aglomeración automática (en inglés algoritmo de *clustering*) de las nuevas estructuras, que son posteriormente analizadas por expertos. El punto fuerte de esta clasificación es su descomposición en dominios, que son bastante consistentes con otros métodos y con las evaluaciones manuales por expertos, y recientemente la incorporación de abundante información sobre la función de los dominios. Las siglas del nombre de la clasificación se refieren a los distintos niveles en los que clasifican las estructuras. Lo que llamamos *fold* en SCOP sería el equivalente al nivel de *Topology* en CATH.
- *FSSP (Fold Classification based on Structure-Structure Alignment of Proteins)* [16]. Esta clasificación es completamente automática y por tanto es la que encontraremos más actualizada. Se basa en la comparación estructural entre proteínas utilizando el algoritmo de alineamiento estructural DALI (que se explica en el Capítulo 10), junto con un umbral de significatividad. En el caso de DALI, al tratarse de un Z-score (Ecuación 10.5, Sección 3.3.3), el umbral utilizado es una similaridad mayor de dos. Aunque en este caso la clasificación se realiza de manera más objetiva, aún existe cierta arbitrariedad en la definición del umbral.

Existen dos críticas fundamentales a las dos primeras clasificaciones que está motivando que replanteen sus esquemas de clasificación [7]. El primero es sobre la definición de lo que se entiende por un *fold*. En la clasificación de SCOP, se dice que *dos proteínas comparten el mismo fold si tienen el mismo conjunto de estructuras secundarias principales con la misma conectividad*. Dilucidar cuáles son estructuras secundarias principales o simples embellecimientos es una tarea que debe ser definida de manera objetiva, lo cual no es sencillo si la clasificación no es automática.

Por otra parte, ambas clasificaciones establecen una jerarquía entre los distintos niveles que definen. Esto está motivado por el hecho de que, si podemos construir un árbol, podríamos en principio “cortarlo” en distintos umbrales de modo que los conjuntos estén unos dentro de otros. El problema surge principalmente porque en uno de los niveles más bajos que definen, el *nivel de Superfamilia*, se asume que las proteínas, además de estar relacionadas estructuralmente compartiendo el mismo *fold*, son homólogas.

Pero vamos a ver que, al no ser la clasificación basada exclusivamente en la estructura, existirán casos en los que no será posible clasificar sin ambigüedad (ver Tabla 12.1 para un resumen). Estos casos podrían surgir por ejemplo cuando encontramos convergencia funcional o si existe una aceleración en el ritmo de acumulación de mutaciones, veamos ambos casos. Entendemos por *convergencia funcional* el proceso evolutivo por el cual dos proteínas presentan la misma función (y en particular para el caso que nos ocupa presentan algún tipo de similaridad estructural, bien local o global) a pesar de no ser posible el reconocer un ancestro común para ambas. Por tanto, *proteínas con orígenes evolutivos distintos convergen estructuralmente debido a que los requerimientos funcionales han hecho que el proceso selectivo encuentre soluciones similares*. El problema con la clasificación surgirá al encontrar estructuras similares como para estar en el mismo *fold* (por ejemplo debido a que existe convergencia funcional) pero una de ellas es homóloga a proteínas con un *fold* muy distinto, pues la jerarquía nos exigirá automáticamente una separación en distintos *folds* para respetar el requerimiento de homología, que se considera que prevalece.

Otro ejemplo problemático es el de proteínas homólogas con *folds* distintos. Si tenemos dos proteínas ortólogas, en principio esperamos que conserven la misma función en ambos organismos y que exista una presión selectiva sobre la estructura que nos permita observar que ambas tienen una similaridad estructural global relevante. Pero si ambas proteínas son parálogas, sabemos que ha existido una duplicación génica y por tanto existe una relajación en la presión selectiva sobre una de ellas, que podría acumular mutaciones a un ritmo más elevado hasta el punto de que no podemos ya considerarla evolución gradual. Por tanto, como se ha observado que pocos cambios en la secuencia permiten obtener plegamientos significativamente distintos, asumir a priori que proteínas homólogas deben tener plegamientos similares puede suponer un problema a la hora de clasificar estructuralmente [1, 29] (ver Figura 12.6). A continuación vamos a ver algunas aproximaciones computacionales para investigar qué mecanismos evolutivos son relevantes y, por extensión, si la clasificación estructural está o no justificada.



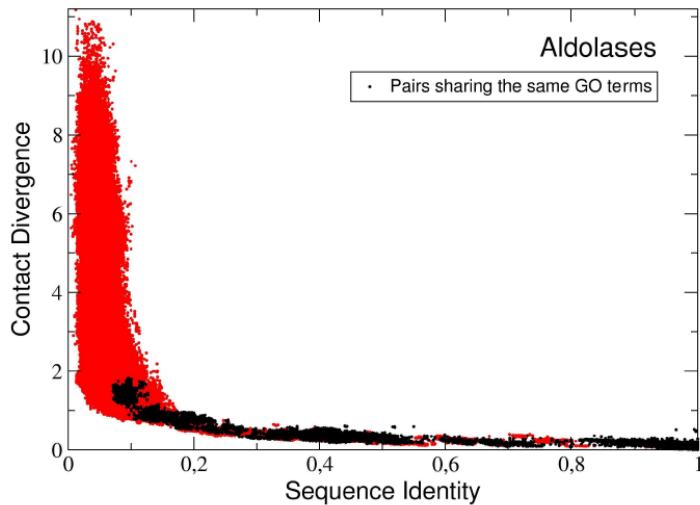
**Figura 12.6:** Divergencia estructural en la superfamilia Cro/Cl, un grupo de factores de transcripción del fago lambda. A pesar de su elevada divergencia su homología ha sido demostrada [29], si bien existe incertidumbre en la reconstrucción. Observamos que la divergencia estructural es muy notable para  $\lambda$  Cro, que forma además un dímero. Este tipo de ejemplos suponen un reto para las clasificaciones estructurales jerárquicas como CATH y SCOP.

Categoría	Ejemplo	Definición/Propiedades	Problemas
Clase estructural	$\alpha/\beta$	Composición global de los elementos de estructura secundaria Mismos elementos de estructura secundaria <i>principales</i> con la misma conectividad. Asunción a priori de relación evolutiva monofilética	
Fold	barril TIM		Casos de convergencia funcional
Superfamilia	Aldolasas	Similaridad en secuencia reconocible	Aceleraciones evolutivas con cambios estructurales que dan lugar a estructuras que se considerarían folds distintos
Familia	Aldolasas clase I	Similaridad en secuencia significativa	Aceleraciones evolutivas con cambios estructurales que dan lugar a estructuras más parecidas a las de otras familias (o folds)
Grupos de ortólogos	2-keto-3deoxy-6-phosphogluconato aldolasa	Relaciones ortólogas dentro del conjunto de especies, conservación de la actividad bioquímica y, a menudo, la función biológica	Típicamente bien resuelto

**Tabla 12.1:** Ejemplo de clasificación jerárquica de proteínas. Para cada categoría se muestra un ejemplo y cómo se determina la pertenencia de una proteína a dicha categoría. La jerarquía implica que las propiedades de los niveles superiores se observan en los niveles inferiores, lo que puede inducir algunos problemas en los casos señalados en la última columna. Tabla adaptada de [19].

### 12.3.2. Cuantificando la divergencia estructural

La pregunta que surge a nivel bioinformático es cómo de fundamentada es la hipótesis de que la partición que obtendríamos en el espacio de estructura de proteínas es consistente con el que encontramos a nivel de secuencias de genes. En primer lugar queremos saber si la divergencia estructural es proporcional a la divergencia evolutiva para proteínas cuyas estructuras no son tan parecidas como en el caso del trabajo de Chothia y Lesk. En su trabajo, utilizaron el *RMSD* pues sus proteínas estaban muy relacionadas evolutivamente y se podían superponer sin problemas, pero esta medida puede generar sesgos importantes si las proteínas son muy divergentes, para lo cual debemos acudir a otras medidas más complejas. Por ejemplo, podemos utilizar la llamada *divergencia de contactos*, que es tratada en detalle en la Sección 10.5. Esta medida utiliza el llamado *solapamiento de contactos*, del inglés *contact overlap*, que recordamos mide la *fracción de residuos alineados entre dos proteínas que se encuentran a una distancia menor de un determinado umbral*. Y lo que realiza es una transformación de el solapamiento de contactos análogamente a como se transforma la identidad en secuencia para obtener una medida de divergencia evolutiva.



**Figura 12.7:** Divergencia estructural frente a divergencia en secuencia para la numerosa superfamilia de las aldolatas. Frente a la medida utilizada en la Figura 12.5, esta medida permite cuantificar divergencias más importantes y podemos observar el efecto de la función en dicha divergencia. Aquellos pares que comparten la misma función están marcados en negro, y vemos que están muy constreñidos estructuralmente. Sin embargo existen pares con distinta función y elevada similaridad estructural.

En la Figura 12.7 podemos observar la relación entre la divergencia de contactos y la similaridad en secuencia para un conjunto de proteínas homólogas de una superfamilia de CATH muy grande, en donde se pueden encontrar divergencias estructurales importantes. Vemos que *la divergencia de contactos está correlacionada linealmente con la divergencia evolutiva para valores bajos de esta variable*, lo que sugiere que crece también linealmente con el tiempo. Sin embargo, para similaridades en secuencia bajas, observamos una explosión en la divergencia de contactos que ya no se correlaciona con la similaridad en secuencia. Esta explosión se relaciona en la gráfica con cambios en la función, si bien lo contrario no es necesariamente cierto, dado que hay proteínas con distinta función y similaridad estructural significativa. Sin embargo, aquellas *proteínas con la misma función están claramente constreñidas desde el punto de vista estructural*. La figura también sugiere que proteínas con distintas funciones han sufrido aceleraciones evolutivas en el ritmo de divergencia estructural, que podrían estar relacionadas con la

presencia de inserciones y delecciones.

Así pues, vemos que *la clasificación estructural podría estar justificada hasta cierto umbral de distancia estructural*, pero deberíamos de tener problemas en clasificar si incorporamos además algún criterio tal como la información en secuencia (por extensión la información evolutiva) o la función. Podríamos tener problemas también si eventos evolutivos más dramáticos, como las inserciones y delecciones, tuvieran un efecto sobre las estructuras que impidieran la clasificación. Antes de aproximarnos a los métodos computacionales orientados a examinar si es posible una partición compatible con la divergencia estructural gradual, vamos a discutir ejemplos de estos eventos más dramáticos.

## 12.4. Evolución estructural mediante ensamblaje de módulos

Como hemos mencionado, las proteínas están formadas por dominios que se pueden considerar autónomos desde el punto de vista estructural, funcional y evolutivo. En algunas ocasiones en los genes eucarióticos las proteínas corresponden a varios exones, donde cada dominio estructural correspondería a un exón, lo que sugiere que *las proteínas multidominio están formadas a través de un proceso de ensamblaje de exones*. De hecho, proteínas con funciones nuevas se forman frecuentemente por el ensamblaje de dominios preexistentes [1, 29].

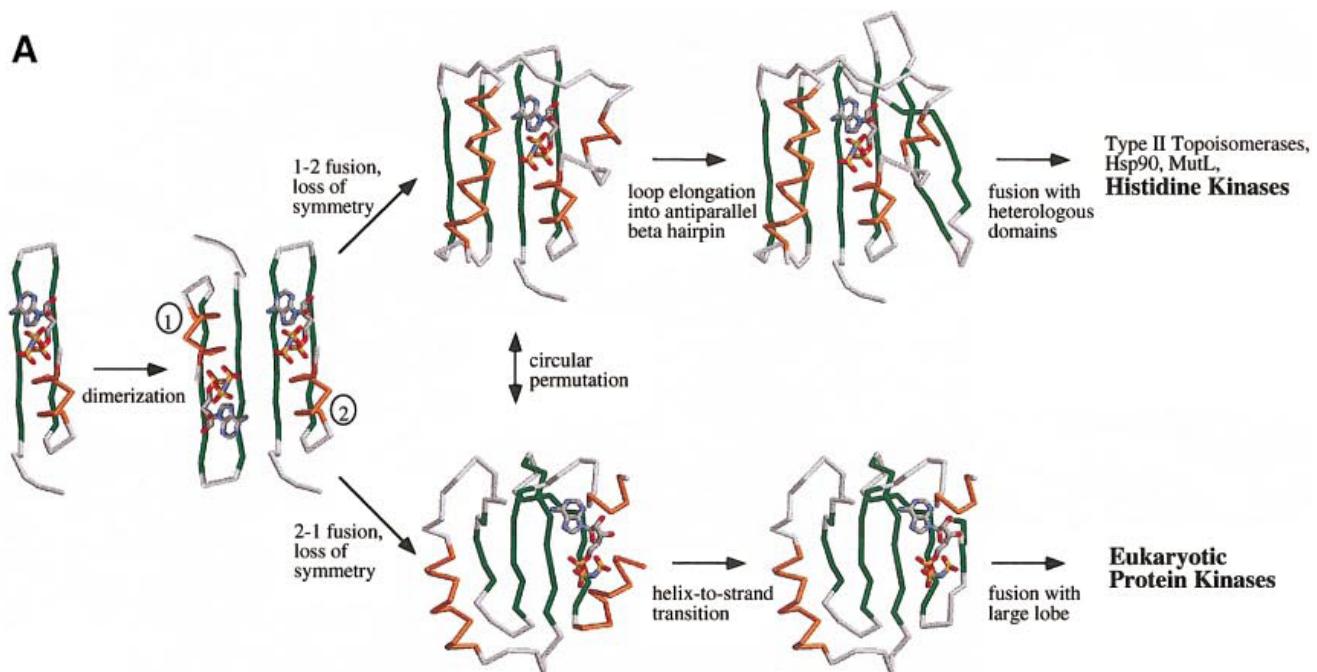
Del mismo modo se ha propuesto [21, 23] que los primeros dominios se han formado a través del ensamblaje de pequeños fragmentos polipeptídicos (del orden de 20 aminoácidos) que, sin ser globulares, tienen una estructura secundaria bien definida. La aparente periodicidad de los elementos de estructura secundaria apoyaría esta teoría [13, 20]. A estos elementos por debajo del nivel de dominio les llamamos *estructuras supersecundarias*. Se han propuesto varios mecanismos para definir estos fragmentos:

### 12.4.1. Longitud típica y posible origen evolutivo común: péptidos ancestrales

Si consideramos una cadena de monómeros, podemos calcular la *probabilidad  $P_\delta(l)$  de que dos monómeros entren en contacto* (estén a menos de una distancia espacial  $\delta$  que fijaremos) dado que hay  $l$  monómeros que los separan. Cuando estén muy cerca el uno del otro, la probabilidad será muy baja pues existe cierta rigidez intrínseca que impedirá a los monómeros entrar en contacto. Y si están muy lejos, los efectos entrópicos impedirán que entren en contacto con una probabilidad razonable. Así que existirá una *longitud óptima para la cual entrarán en contacto con máxima probabilidad* [4], que está relacionada con la denominada *longitud de persistencia*. Esta longitud óptima se puede estimar teóricamente mediante la teoría de polímeros [10], y dependerá de la naturaleza de los monómeros. Por ejemplo, para las prolinas esperamos tener que la probabilidad se maximiza para valores de  $l$  alrededor de 200 monómeros y para las glicinas de tan sólo 4. Pero en general, como en las proteínas tenemos cantidades bajas de ambas esperamos tener unos valores de  $l$  de entre 20 y 30 monómeros.

Si recorremos cadenas de proteínas contando el número de monómeros que separan cualquier par de contactos entre monómeros, encontraremos que la distribución de tamaños tiene un *máximo* precisamente alrededor de los 25 monómeros. Este tamaño está por encima de la longitud típica de un elemento de estructura secundaria pero por debajo de la longitud de un dominio estructural. Además, estos fragmentos cerrados no se encuentran ocasionalmente aquí y allá en las estructuras sino que, al recorrerlas, los encontramos uno tras otro [3], lo que nos lleva a pensar sobre un posible origen evolutivo.

Por ejemplo, si pensamos en un escenario en el que aún no existieran proteínas con funciones complejas, es razonable pensar en estos módulos como antecesores de las proteínas actuales, pues presentan una longitud y estabilidad termodinámica mínima para empezar a realizar procesos catalíticos rudimentarios

**A**

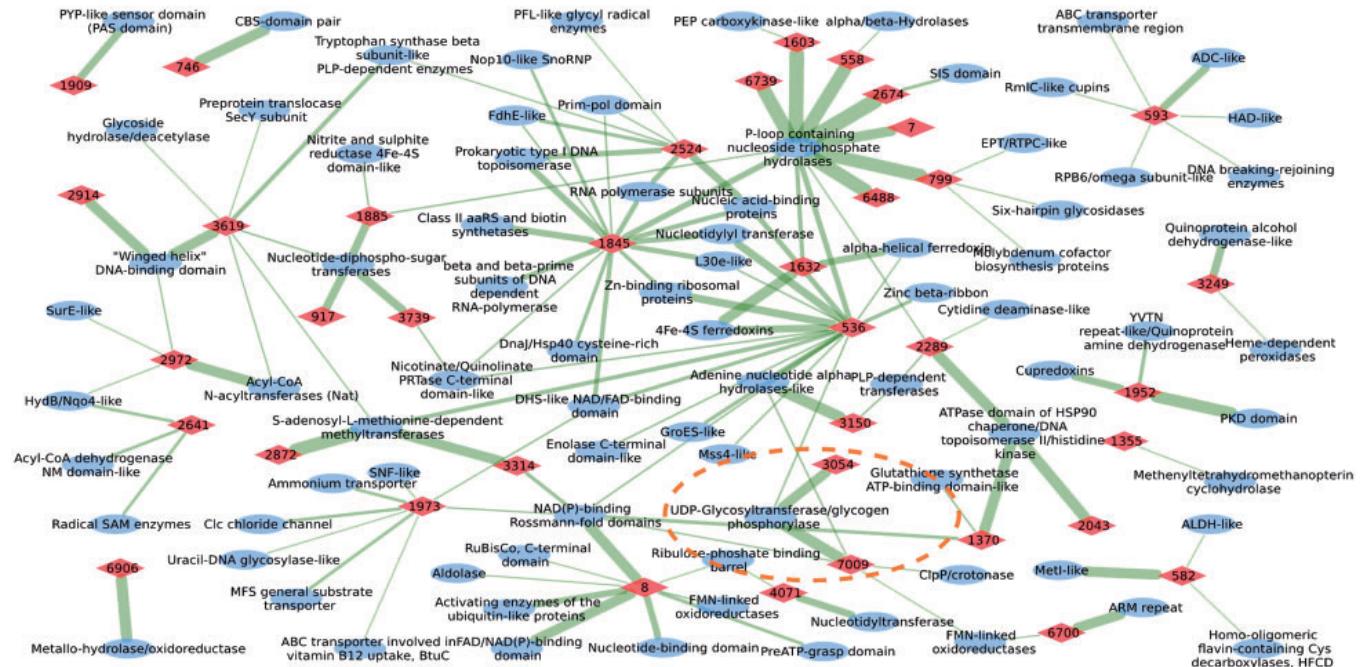
**Figura 12.8:** Trayectoria evolutiva especulativa descrita en [21], para explicar las similaridades locales encontradas entre los dominios de unión a ATP en histidina kinasas bacterianas y kinasas eucarióticas. El fragmento ancestral inicial, de tipo  $\alpha\beta\beta$ , se habría dimerizado y fusionado en orden inverso en cada trayectoria. A pesar de no tener una elevada similaridad en secuencia entre ambas, sus correspondientes elementos  $\alpha\beta\beta$  se alinean estructuralmente con una resolución menor a 2 Å. Figura reproducida con permiso del editor.

de manera semiautónoma. Por este motivo *se ha propuesto que algunas de las repeticiones de fragmentos cerrados de esta longitud encontrados sistemáticamente en las proteínas que podemos hoy observar, podrían ser las reliquias de péptidos ancestrales* a partir de los cuales se formaron estructuras más complejas por duplicación y fusión de los (también pequeños) genes asociados. En la Figura 12.8 mostramos un ejemplo de la trayectoria evolutiva de dos proteínas bajo este escenario especulativo [21]. Pero, ¿cómo compaginar este escenario con los eventos evolutivos dominantes que conocemos? ¿cómo se relaciona la función de estos fragmentos ancestrales con la función de las proteínas actuales? Pues bien, se postula también que *la formación de dominios estructurales más largos, de entre 100 y 150 aminoácidos, podría ser compatible con un momento posterior en el que se encuentra un nuevo escenario de estabilidad debido a la circularización del DNA ancestral*, ya que el tamaño óptimo de cierre del DNA (determinado de nuevo por su flexibilidad) es de alrededor de 400 pares, consistente con el tamaño típico de los dominios actuales.

#### 12.4.2. Relación con la función y búsqueda por recurrencia

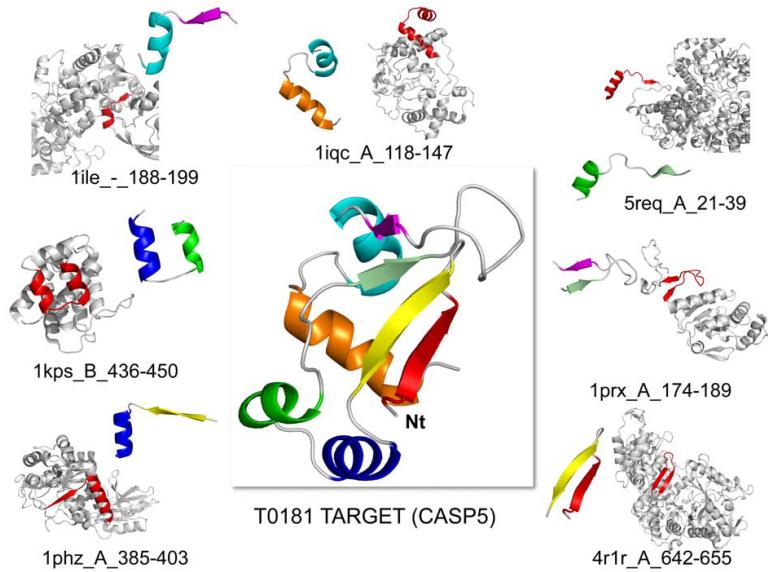
Pero, independientemente de su origen, existe un creciente interés por la búsqueda de estos fragmentos ante su posible relevancia funcional [9, 32, 34]. Si hablamos de relevancia funcional, tenemos que hablar también de la secuencia. Hemos visto que existen proteínas homólogas que, aun conservando la misma función, tienen una similaridad en secuencia indistinguible de lo esperado por azar [26]. Por tanto, dado que el número de residuos es aún más pequeño para un fragmento que para una proteína, la probabilidad de encontrar similaridades significativas en una secuencia tan corta es muy baja.

Un primer paso para abordar el problema lo encontramos en la base de datos de *BLOCKS* [14]. En esta base de datos se extraen fragmentos conservados de grupos clasificados por INTERPRO [2]. Curiosamente, la mayoría de estos fragmentos ~ 75 % tienen tamaños de entre 10 y 40 aminoácidos. Con estos bloques en secuencia, podríamos dividir una proteína en regiones en función de su correspondencia con los bloques. Se puede observar que existen correspondencias claras a lo largo de una proteína [11], con otras regiones sin asignación que podrían tener un rol únicamente de soporte a las regiones funcionales (ver Figura 12.9)



**Figura 12.9:** Cada superfamilia (elipses azules) está conectada a otra superfamilia si comparten algún fragmento (rombos naranjas). Los fragmentos se han obtenido a partir de perfiles construidos considerando proteomas completos de archaea junto con la hipótesis de la existencia de péptidos ancestrales, y posteriormente se han buscado similaridades en el espacio de estructuras [12]. Figura reproducida con permiso del editor.

Así que se han propuesto distintas aproximaciones para extraer fragmentos. La mayoría de las aproximaciones se basan en la búsqueda de motivos recurrentes a partir de una clasificación previa [9, 32, 33], acompañado de algún método para extraer estructuras consenso como técnicas de reducción de la dimensionalidad (Subsección 11.4.1), o algún criterio más teórico como el mencionado tamaño típico a partir de la longitud de persistencia. Cada autor se refiere a los fragmentos con nombres diferentes, como *legos*, *Smotifs*, etc. Sorprendentemente, *con tan sólo 70 legos es posible cubrir el 90 % de los plegamientos conocidos* [32] lo que nos lleva a pensar en su utilidad en el modelado de proteínas. Un ejemplo de cómo se puede cubrir una proteína desconocida con fragmentos lo representamos en la Figura 12.10. Además, se observa que aquellos fragmentos más frecuentes tienen propiedades características como un mayor número de contactos internos, mientras que otros más raros contienen *loops* más largos. Se ha observado además que cuando se identifica un nuevo *fold*, no se identifican nuevos fragmentos sino una nueva combinación de los mismos [9].



**Figura 12.10:** Cobertura de una proteína problema en la competición de CASP 5 mediante fragmentos encontrados en otras proteínas [9]. Hay que notar que en este caso algunos de los fragmentos no cumplen con el requerimiento de ser un loop cerrado, como se asume en algunas de las hipótesis evolutivas discutidas. Figura reproducida con permiso de los autores.

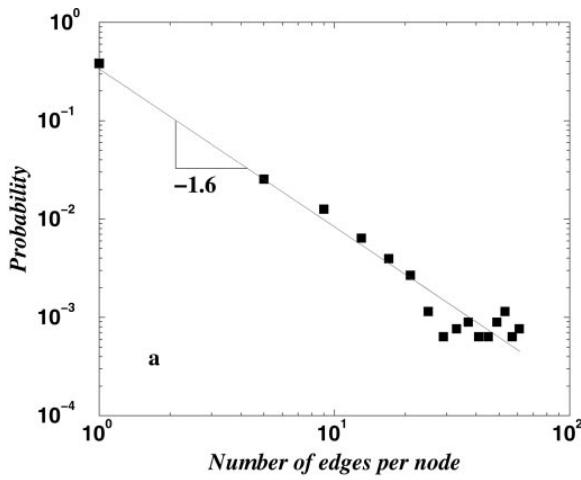
## 12.5. Divergencia estructural Vs. ensamblaje de módulos: consecuencias para la clasificación estructural y la modelización de estructura de proteínas

Si estos fragmentos pueden tener un rol claro funcional cabe preguntarse si la clasificación estructural tiene sentido ya que, *si toda las proteínas conocidas pueden ser construidas por un conjunto relativamente pequeño de fragmentos, deberíamos encontrar que muchos de los folds que pretendemos construir tienen similaridades locales entre sí*. Y esto es precisamente lo que se ha encontrado cuando se han comparado distintos *folds*: es posible relacionar muchos de ellos dentro de un umbral de similaridad significativo [27, 31], atribuibles a fragmentos de estructura supersecundaria. Si esto es así, a priori deberíamos de pensar que no es posible clasificar proteínas pues, como hemos explicado anteriormente, no vamos a encontrar ningún umbral dentro del cual los grupos sean realmente disjuntos. Esto será así a no ser que haya algún argumento desde el punto de vista evolutivo que justificara un “salto” en la similaridad y, a pesar de compartir localmente ciertos fragmentos, se pudiera decir que existe aún una separación clara entre los grupos debidas a similaridades globales. Este salto evolutivo estaría justificado por el hecho de que cualquier combinación de fragmentos ensamblados tiene una probabilidad baja de ser termodinámicamente estable. Si no fuese así, un escenario en el que el espacio es totalmente continuo y no es posible clasificar sería lo que esperaríamos. Así pues, *deberían existir ciertas combinaciones favorecidas, y estas combinaciones formarían los folds*, y las similaridades locales serían debidas o bien a que existen ciertos fragmentos ancestrales compartidos, o a mecanismos evolutivos más dramáticos como grandes inserciones correspondientes a fragmentos completos.

¿Es posible testar computacionalmente este escenario? Se han sugerido algunas aproximaciones para buscar de manera objetiva un salto en la similaridad de las estructuras de proteínas, con el objeto de determinar un umbral intrínseco en el cual es posible clasificar las estructuras. Como hemos señalado anteriormente, el disponer de clases de equivalencia estructurales con características comunes desde

el punto de vista evolutivo, funcional o ambos, sería de gran utilidad a la hora de modelar nuevas estructuras, ya que estas clases se pueden utilizar como plantillas a partir de las cuales construir los modelos.

Una de estas aproximaciones [8] se basa en la *teoría de redes complejas*, como ya hemos comentado anteriormente. Esencialmente el proceso consiste en *construir una red uniendo aquellas proteínas que sean más parecidas estructuralmente*, lo que generará en un primer momento grupos disjuntos. Al ir reduciendo la similaridad, algunos de los conjuntos irán creciendo y, eventualmente, algunos de estos grupos se irán uniendo entre sí. De nuevo surge la pregunta, ¿hasta cuándo juntar y, por tanto, en qué momento cortar el árbol que se va formando? En particular nos interesará el conjunto más grande ya que, si alcanza cierto tamaño crítico, nos puede indicar la existencia de una transición. En el trabajo que mencionamos se observó que existe una transición brusca del tamaño de este conjunto gigante, que se denomina percolación. La *percolación* es lo que ocurre por ejemplo cuando hacemos un café en una cafetera italiana. Mientras calentamos el agua, unas pequeñas burbujas de vapor atravesarán el café condensándose y formando pequeñas burbujas, que serían similares a nuestros conjuntos. A medida que sigamos calentando, el vapor encontrará ciertos caminos preferentes y algunos de estos conjuntos crecerán más que los demás hasta un punto en el cual el vapor percola, y atraviesa el café sin dificultad (momento en el cual el café empieza a fluir en el recipiente). Sabemos que, entre la formación de las primeras burbujas y el momento en el que el vapor fluye, existe un punto crítico en la transición que está bien descrito por la distribución de probabilidad del tamaño de las burbujas [28]. Y esta distribución es precisamente la misma que comentábamos en la Figura 12.4, una distribución libre de escala. Por tanto si, al construir nuestra red, encontramos un punto crítico que se pueda describir de manera similar, estaríamos ante una transición de fases similar a la del café pero en el universo de estructuras de proteínas. Este punto lo encontraríamos monitorizando también el tamaño del conjunto más grande con la esperanza de observar una transición clara. En este trabajo, se observó efectivamente una transición en el conjunto gigante. Y en el *punto de la transición*, además se obtuvo una *distribución libre de escala para el número de estructuras similares a cada una de las estructuras consideradas*, como se observa en la Figura 12.11.

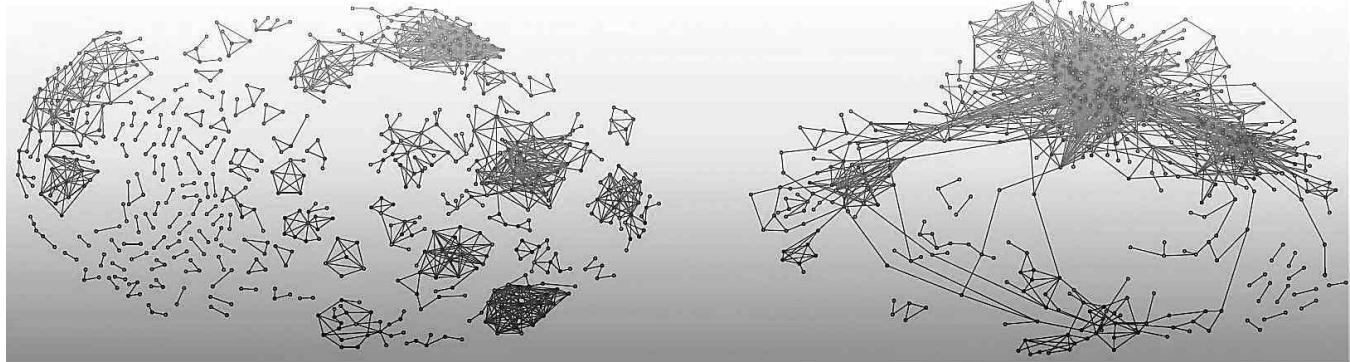


**Figura 12.11:** Distribución de probabilidad del número de relaciones de similaridad significativa que presenta cada estructura en la transición de percolación [8]. La distribución es consistente con la encontrada en la Figura 12.4. Figura reproducida con permiso del editor.

Observar esta transición es determinante en nuestra comprensión de la estructura del universo de proteínas, y ha sido posteriormente respaldada por otras aproximaciones. Por ejemplo, y como decíamos

anteriormente, si dos proteínas están relacionadas entre sí porque su distancia  $d(a, b)$  es pequeña, esperaríamos que al compararlas con una tercera proteína se cumpliera el que  $d(a, c) \approx d(b, c)$ . Si esta relación no se cumple decimos que la *transitividad* está comprometida, es decir, el hecho de que estando  $a$  relacionada con  $b$  y estando  $b$  relacionada con  $c$  resulte que  $a$  y  $c$  no están relacionadas. El que la transitividad se respete permite formalmente definir clases de equivalencia, que es nuestro *proxy* a los conjuntos de estructuras de proteínas. Pues bien, basándose en esta definición de *clase de equivalencia*, se puede medir la violación de transitividad en un proceso de aglomeración que permite detectar también una transición consistente con el resultado anterior [27].

Esta transición es importante también porque nos indicaría que existe una *dualidad entre una concepción discreta y continua del espacio de estructura de proteínas* [30]. Para *similaridades altas*, tendríamos estructuras relacionadas preferentemente por duplicación génica y posterior divergencia. Para los lectores interesados, un modelo computacional que respalda esta hipótesis utilizando modelos sencillos de plegamiento de proteínas (como los explicados en el capítulo de plegamiento de proteínas (Capítulo 11) se describe en [36]. Encontraríamos grupos disjuntos que constituirían una clasificación objetiva de las estructuras. Para *baja similaridad* estructural (más allá del punto crítico), tendríamos en cambio que están relacionadas al nivel de fragmentos de estructura super-secundaria, y el mecanismo evolutivo que lo explicaría podría ser, o bien una consecuencia del modelo en el que hemos discutido la posibilidad de que existan fragmentos ancestrales, o bien por algún mecanismo de transferencia horizontal. Ambos escenarios son representados en la Figura 12.12.



**Figura 12.12:** Representación artística de las relaciones de similaridad (enlaces) entre proteínas (nodos) para un umbral de significatividad alto (izquierda) compatible con un espacio discreto y por tanto con una clasificación estructural, y un umbral de significatividad más bajo (derecha) en el que las similaridades son locales y la representación es continua, solo compatible con una red y no con un árbol (detalles en [27]). Figura reproducida con permiso de los autores.

Sin embargo, desde el punto de vista de la modelización, el reciente interés por los fragmentos de estructura supersecundaria nos proporciona algunas novedades a tener en cuenta. Ya que si encontramos fragmentos de estructuras similares entre distintos *folds* que conservan la misma función, podemos estudiar la evolución en secuencia que han sufrido estos fragmentos conectándolos a través de una red neutral [11]. Una *red neutral* es una red que se construye relacionando secuencias que están separadas por mutaciones puntuales, y el lector interesado en profundizar en este campo encontrará un ejemplo sobre evolución neutral de RNA en la Subsección 14.5.2. El relacionar a través de la secuencia estas redes condicionado a que comparten la misma estructura y función, nos abrirá muchas puertas para entender la evolución, ya que recorreremos distintos *folds* y organismos. Además, la estructura de la red nos muestra que hay algunas secuencias más conectadas que otras, lo que indica que son “sumideros”

desde el punto de vista evolutivo.

Cuando tengamos una secuencia cuya estructura desconocemos, ¿qué aproximación deberíamos seguir entonces? Pues haciendo uso de la dualidad que se observa en el espacio de estructura de proteínas, probablemente hay que apostar por *una aproximación que combine el uso de una clasificación estructural para intentar aproximarnos a una estructura con parecido global, y de una base de datos de fragmentos que nos permita refinarla*. Por tanto, el éxito de la modelización pasa sin duda por entender en profundidad los mecanismos evolutivos que subyacen, lo que abrirá puertas a nuevas aproximaciones que están aún por escribir.

## Agradecimientos

El autor agradece la lectura crítica del manuscrito a Julián Echave, y la aportación de parte del material a Ugo Bastolla.



## 12.6. Bibliografía

- [1] P. A. Alexander, Y. He, Y. Chen, J. Orban, and P. N. Bryan. A minimal sequence code for switching protein structure and function. *Proceedings of the National Academy of Sciences*, 106(50):21149–21154, Dec. 2009.
- [2] R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher, L. Cerutti, F. Corpet, M. D. R. Croning, R. Durbin, L. Falquet, W. Fleischmann, J. Gouzy, H. Hermjakob, N. Hulo, I. Jonassen, D. Kahn, A. Kanapin, Y. Karavidopoulou, R. Lopez, B. Marx, N. J. Mulder, T. M. Oinn, M. Pagni, F. Servant, C. J. A. Sigrist, and E. M. Zdobnov. The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Research*, 29(1):37–40, Jan. 2001.
- [3] I. N. Berezovsky, A. Y. Grosberg, and E. N. Trifonov. Closed loops of nearly standard size: common basic element of protein structure. *FEBS Letters*, 466(2-3):283–286, Jan. 2000.
- [4] I. N. Berezovsky and E. N. Trifonov. Loop fold nature of globular proteins. *Protein Engineering*, 14(6):403–407, June 2001.
- [5] J.-Q. Chen, Y. Wu, H. Yang, J. Bergelson, M. Kreitman, and D. Tian. Variation in the ratio of nucleotide substitution and indel rates across genomes in mammals and bacteria. *Molecular Biology and Evolution*, 26(7):1523–1531, July 2009.
- [6] C. Chothia and A. M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO Journal*, 5(4):823–826, Apr. 1986. PMID: 3709526 PMCID: PMC1166865.
- [7] A. L. Cuff, I. Sillitoe, T. Lewis, O. C. Redfern, R. Garratt, J. Thornton, and C. A. Orengo. The CATH classification revisited—architectures reviewed and new ways to characterize structural divergence in superfamilies. *Nucleic Acids Research*, 37(Database):D310–D314, Jan. 2009.
- [8] N. V. Dokholyan, B. Shakhnovich, and E. I. Shakhnovich. Expanding protein universe and its origin from the biological big bang. *Proceedings of the National Academy of Sciences*, 99(22):14132–14136, Oct. 2002.
- [9] N. Fernandez-Fuentes, J. M. Dybas, and A. Fiser. Structural characteristics of novel protein folds. *PLoS Comput Biol*, 6(4):e1000750, Apr. 2010.
- [10] P. J. Flory and M. Volkenstein. Statistical mechanics of chain molecules. *Biopolymers*, 8(5):699–700, 1969.
- [11] Z. M. Frenkel and E. N. Trifonov. From protein sequence space to elementary protein modules. *Gene*, 408(1-2):64–71, Jan. 2008.
- [12] A. Gonçarenc and I. N. Berezovsky. Prototypes of elementary functional loops unravel evolutionary connections between protein functions. *Bioinformatics*, 26(18):i497–i503, Sept. 2010.
- [13] N. V. Grishin. Fold change in evolution of protein structures. *Journal of Structural Biology*, 134(2-3):167–185, May 2001.
- [14] S. Henikoff, J. G. Henikoff, and S. Pietrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, June 1999.
- [15] T. A. Holland, S. Veretnik, I. N. Shindyalov, and P. E. Bourne. Partitioning protein structures into domains: Why is it so difficult? *Journal of Molecular Biology*, 361(3):562–590, Aug. 2006.
- [16] L. Holm and C. Sander. Dali/FSSP classification of three-dimensional protein folds. *Nucleic Acids Research*, 25(1):231–234, Jan. 1997.
- [17] M. A. Huynen and E. v. Nimwegen. The frequency distribution of gene family sizes in complete genomes. *Molecular Biology and Evolution*, 15(5):583–589, May 1998.
- [18] C. Kimchi-Sarfaty, J. M. Oh, I.-W. Kim, Z. E. Sauna, A. M. Calcagno, S. V. Ambudkar, and M. M. Gottesman. A silent polymorphism in the MDR1 gene changes substrate specificity. *Science*, 315(5811):525–528, Jan. 2007.
- [19] E. V. Koonin, Y. I. Wolf, and G. P. Karev. The structure of the protein universe and genome evolution. *Nature*, 420(6912):218–223, Nov. 2002.
- [20] J. Lee and M. Blaber. Experimental support for the evolution of symmetric protein architecture from a simple peptide motif. *Proceedings of the National Academy of Sciences*, 108(1):126–130, Jan. 2011.

- [21] A. N. Lupas, C. P. Ponting, and R. B. Russell. On the evolution of protein folds: Are similar motifs in different protein folds the result of convergence, insertion, or relics of an ancient peptide world? *Journal of Structural Biology*, 134(2-3):191–203, May 2001.
- [22] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, Apr. 1995. PMID: 7723011.
- [23] S. OHNO. *Evolution by gene duplication*. Springer-Verlag, 1970.
- [24] C. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, and J. Thornton. CATH - a hierachic classification of protein domain structures. *Structure*, 5(8):1093–1109, Aug. 1997.
- [25] C. Pál, B. Papp, and M. J. Lercher. An integrated view of protein evolution. *Nature Reviews Genetics*, 7(5):337–348, May 2006.
- [26] A. Pascual-García, D. Abia, R. Méndez, G. S. Nido, and U. Bastolla. Quantifying the evolutionary divergence of protein structures: The role of function change and function conservation. *Proteins: Structure, Function, and Bioinformatics*, 78(1):181–196, 2010.
- [27] A. Pascual-García, D. Abia, Á. R. Ortiz, and U. Bastolla. Cross-over between discrete and continuous protein structure space: Insights into automatic classification and networks of protein structures. *PLOS Computational Biology*, 5(3):e1000331, Mar. 2009.
- [28] F. Radicchi and S. Fortunato. Explosive percolation in scale-free networks. *Physical Review Letters*, 103(16):168701, Oct. 2009.
- [29] C. G. Roessler, B. M. Hall, W. J. Anderson, W. M. Ingram, S. A. Roberts, W. R. Montfort, and M. H. J. Cordes. Transitive homology-guided structural studies lead to discovery of cro proteins with 40% sequence identity but different folds. *Proceedings of the National Academy of Sciences*, 105(7):2343–2348, Feb. 2008.
- [30] R. I. Sadreyev, B.-H. Kim, and N. V. Grishin. Discrete-continuous duality of protein structure space. *Current Opinion in Structural Biology*, 19(3):321–328, June 2009.
- [31] J. Skolnick, A. K. Arakaki, S. Y. Lee, and M. Brylinski. The continuity of protein structure space is an intrinsic property of proteins. *Proceedings of the National Academy of Sciences*, 106(37):15690–15695, Sept. 2009.
- [32] J. D. Szustakowski, S. Kasif, and Z. Weng. Less is more: towards an optimal universal description of protein folds. *Bioinformatics*, 21(Suppl 2):ii66–ii71, Oct. 2005.
- [33] C.-H. Tai, V. Sam, J.-F. Gibrat, J. Garnier, P. J. Munson, and B. Lee. Protein domain assignment from the recurrence of locally similar structures. *Proteins: Structure, Function, and Bioinformatics*, 79(3):853–866, 2011.
- [34] E. N. Trifonov and Z. M. Frenkel. Evolution of protein modularity. *Current Opinion in Structural Biology*, 19(3):335–340, June 2009.
- [35] X. F. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *IEEE Circuits and Systems Magazine*, 3(1):6 – 20, 2003.
- [36] K. B. Zeldovich, P. Chen, B. E. Shakhnovich, and E. I. Shakhnovich. A first-principles model of early evolution: Emergence of gene families, species, and preferred protein folds. *PLoS Comput Biol*, 3(7):e139, July 2007.

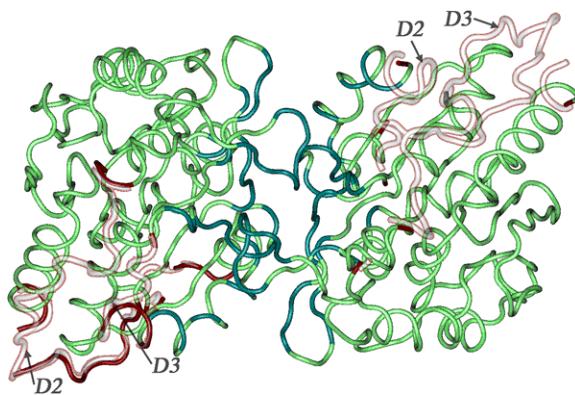
# Capítulo 13

## Proteínas desordenadas

*Inmaculada Yruela*

### 13.1. Introducción

En general se tiene la idea, acuñada durante el s. XX, de que existe una estrecha relación entre la estructura y la función de una proteína (*secuencia de aminoácidos → estructura 3D → función*), y que una condición indispensable para que una proteína desarrolle una función es que tenga en su estado nativo una estructura 3D. Sin embargo, en los últimos 20 años los estudios han revelado que hay un número elevado de proteínas que no adoptan estructuras tridimensionales definidas y realizan importantes funciones biológicas en su estado nativo. La posible utilidad de tales regiones fue sugerida por primera vez hace 70 años por Linus Pauling, quién especuló sobre su flexibilidad en la producción de anticuerpos. A nivel estructural, los primeros indicios de regiones desestructuradas en proteínas surgieron cuando tan sólo se habían determinado 20 estructuras de proteínas por rayos-X, en las que aparecían regiones no discernibles debido a su pobre densidad electrónica, y que sin embargo tenían una relevante función.



**Figura 13.1:** Estructura de la chaperona hsp31 (pdb 1PV2). Las regiones estructuradas se muestran en verde, y las regiones desestructuradas en blanco (regiones no definidas en pdb 1PV2). El posible orden que se muestra de las regiones en blanco está simulado en base a la estructura del pdb 1ONS (misma proteína en estado monomérico sin desorden) [7].

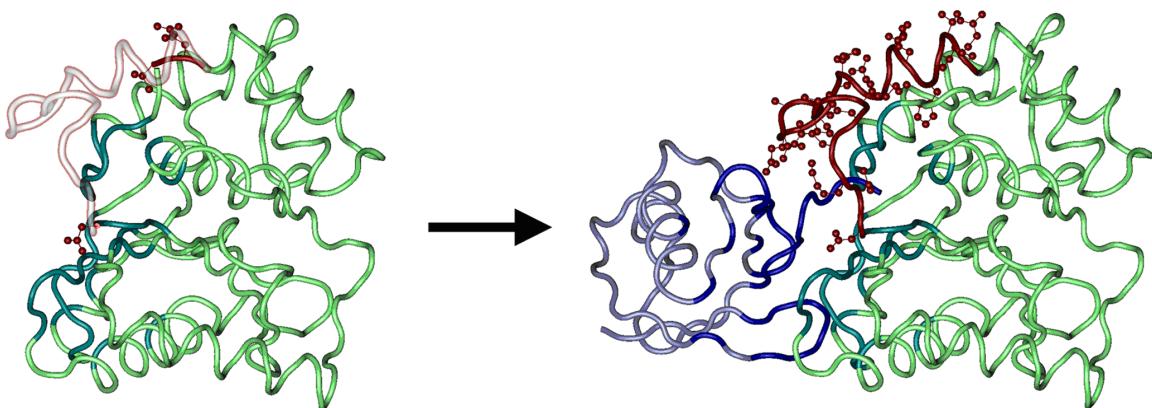
Esta clase de proteínas cuestionaba en parte el paradigma central de la biología molecular formulado

por Francis Crick (1958) que postula que a cada secuencia de aminoácidos le corresponde una estructura tridimensional. La pérdida de densidad electrónica en las estructuras de rayos-X puede surgir por un fallo al resolver el problema de la fase, por defectos del cristal o por sucesos de eliminación proteolítica accidental durante el proceso de purificación de la proteína. Sin embargo, una explicación común para la falta de densidad electrónica es que el átomo, residuo, cadena lateral o segmento no observado no disperse los X-rayos de forma coherente, debido a la variación en su posición de una proteína a la siguiente o próxima, es decir, los átomos no observados son desordenados.

En 1978, el mismo año en que el desorden funcional fue definido por cristalográfia de rayos-X, la técnica de resonancia paramagnética electrónica (NMR) reveló que la cola de histona H5, altamente cargada era desordenada y podía ser clasificada como una proteína desordenada o desestructurada [2]. Actualmente la literatura contiene numerosos datos de regiones desordenadas que son esenciales para la función de las proteínas. Ante este escenario creo que sería conveniente revisar el concepto de desorden en las proteínas y hablar mejor de ductilidad de las mismas en vez de desorden y de zonas dúctiles en vez de regiones desordenadas. Si bien al comienzo llamó la atención el carácter desordenado respecto al orden conocido en las proteínas globulares, los estudios recientes muestran la función positiva de estas regiones altamente moldeables, y por tanto el término ductilidad, aún no establecido para referirse a esta característica, resulta más apropiado.

Las proteínas que carecen de estructuras definidas se conocen actualmente por el nombre de *Proteínas Intrínsecamente Desordenadas* (PIDs) y están presentes en todos los organismos vivos. Las PIDs pueden contener regiones desordenadas o dúctiles, y ser así parcialmente desordenadas, o carecer de un plegamiento estructurado en su conjunto, siendo por tanto completamente desordenadas de forma aislada. Desde un punto de vista termodinámico el desorden en una proteína se define como un estado estructural random coil. El desorden puede encontrarse en bucles flexibles o giros, dominios, unión entre dominios o en proteínas completas. Las PIDs no pueden ser descritas por una sola conformación pues adoptan múltiples estructuras. Deben representarse como un conjunto de éstas, algunas compactas, otras extendidas, de estabilidad similar y que se intercambian a una gran velocidad, más de un millón de veces por segundo. La presencia de regiones desordenadas confiere flexibilidad, lo que es una ventaja para el reconocimiento de múltiples moléculas (ARN, ADN, otras proteínas, pequeños ligandos o moléculas). Las PIDs suelen desempeñar funciones que dependen de la unión a otras moléculas, pueden unirse a diversas dianas moleculares e incluso pueden adoptar estructuras diferentes en los distintos complejos finales. Pueden permitir transiciones entre diferentes estados conformacionales o estructurales. Los procesos de unión de las proteínas desordenadas están caracterizados por una baja afinidad, es decir, las uniones son generalmente débiles, pero en cambio son altamente específicas.

Si se quieren ampliar conocimientos sobre este tipo de proteínas se recomienda leer el reciente monográfico de la revista Chemical Reviews dedicado a las PIDs [1].



**Figura 13.2:** Transición desorden-orden en la formación del complejo ubiquitina C-terminal ubiquitina hidrolasa. En la figura de la izquierda se muestra el monómero ubiquitina hidrolasa (1UCH) con la región desordenada marcada en blanco y los aminoácidos que flanquean esta región en rojo. En la figura de la derecha se muestra el complejo formado entre la ubiquitina hidrolasa y la ubiquitina con los mismos aminoácidos marcados en rojo [7].

### 13.2. Desorden en proteínas

Las PIDs se pueden caracterizar mediante diversos métodos experimentales, cada uno con sus propios puntos fuertes y débiles. Sin embargo, esta caracterización resulta en la mayoría de los casos parcial. A continuación se describen las principales técnicas experimentales usadas.

La cristalografía y difracción de rayos-X, como se mencionó anteriormente, no puede detectar regiones desestructuradas en proteínas por su falta de densidad electrónica. La mayor incertidumbre de esta técnica es que, sin experimentos adicionales, no se puede definir con total exactitud si una región con falta de densidad electrónica es un dominio PID o es el resultado de dificultades técnicas. Una metodología alternativa a la cristalografía de rayos-X convencional para estudiar proteínas PID es la dispersión de rayos X de ángulo pequeño (SAXS).

La resonancia magnética nuclear (RMN) puede resolver estructuras 3D de proteínas en solución. El hecho de que no sea necesario cristalizar la proteína para resolver su estructura 3D hace que esta técnica proporcione una estimación menos sesgada del desorden en comparación con la determinación por cristalografía y difracción de rayos-X. Bajo circunstancias favorables esta técnica proporciona información sobre la movilidad de cada residuo. Sin embargo, comparado con los resultados obtenidos con proteínas ordenadas, los datos relativos a proteínas PID son relativamente escasos. Esto indica que el estudio de proteínas PID con esta técnica también tiene dificultades. Las proteínas PID suelen formar agregados a las concentraciones necesarias para experimentos de RMN, además de presentar alta heterogeneidad con interconversiones estructurales en el orden de milisegundos que ocasionan un elevado ensanchamiento de los picos espectrales. Estas dificultades hacen que con la técnica de RMN los datos sobre desorden no sean tan abundantes.

La espectroscopía de dicroismo circular (CD) también puede proporcionar información estructural de las proteínas en solución. Los espectros de CD UV-lejano proporcionan estimaciones de estructura secundaria y pueden distinguir entre estructuras globulares ordenadas y bucles o giros flexibles en estado de glóbulo fundido (carente de estructura compacta globular). Por otro lado, el CD UV-cercano muestra picos estrechos para grupos aromáticos cuando se ordena la proteína, pero estos picos desaparecen en el estado de glóbulo fundido debido a la movilidad promedio de los átomos. La combinación del uso del CD

UV-cercano y CD UV-lejano puede distinguir si se ordena una proteína o si se encuentra desordenada. Sin embargo, este método es sólo semicuantitativo y no proporciona información sobre aminoácidos específicos. Por tanto una limitación de esta técnica es que no proporciona una clara información para las proteínas que contienen tanto regiones ordenadas como desordenadas.

La digestión con proteasas proporciona indicios de la flexibilidad de una proteína estructurada, la flexibilidad no es una mera exposición superficial, es el factor determinante para la búsqueda de sitios de corte de digestión. Estudios han demostrado que las regiones PID tienen una hipersensibilidad a las proteasas. Así, la digestión con proteasas es especialmente útil cuando se utiliza en combinación con otros métodos. Por ejemplo, la digestión con proteasas puede usarse con difracción de rayos-X, para resolver si una región con pérdida de densidad electrónica es debido a un estado de desorden. También se usa en combinación con el dicroísmo circular o la espectrometría de masas.

La espectroscopía de resonancia paramagnética electrónica (EPR) combinada con el marcaje sitio-dirigido de etiquetas de spin (SDSL) es una de las técnicas hoy en día más adecuadas para estudiar la estructura y dinámica de PIDs. La espectroscopía SDLS-EPR ha alcanzado actualmente un nivel que hace que su aplicación en este campo sea cada vez más extendido [5].

### 13.3. Predicción de desorden en proteínas

Las técnicas experimentales que se han descrito anteriormente constituyen una herramienta muy valiosa para el estudio de PIDs, aunque presentan ciertas limitaciones. Una de ella es la identificación de este tipo de proteínas en estudios post-genómicos. En el caso del proteoma humano, hasta la fecha se han identificado unas 600 proteínas total o parcialmente desestructuradas y se ha descrito su función por medio de técnicas experimentales. Pero este número sólo constituye una pequeña parte del total de proteínas estimadas. Ante este panorama un enfoque bioinformático resulta indispensable para avanzar en la identificación y caracterización de PIDs.

Los primeros métodos bioinformáticos se basaron en los primeros estudios teóricos de proteínas individuales. Estos estudios sugerían que después de ser sintetizada una cadena de aminoácidos para producir una proteína, la cadena se pliega de una manera que depende de su composición. En concreto, los aminoácidos voluminosos e hidrofóbicos (los que repelen las moléculas de agua, que de forma natural rodean a las proteínas) se sitúan en el interior de la molécula proteíca. Por el contrario los aminoácidos que se colocan en la superficie plegada de la misma suelen ser pequeños e hidrofílicos (interaccionan con las moléculas de agua circundantes).

Así, en los inicios, el planteamiento subyacente a los métodos computacionales de predicción de desorden era comparar las secuencias de aminoácidos de las proteínas que se conocían como PIDs con las que presentaban formas plegadas rígidas. Los primeros cálculos realizados por Dunker en 1997 descubrieron que las PIDs presentaban mayor número de aminoácidos hidrofílicos que las proteínas rígidas o compactas. Por tanto la relación entre aminoácidos hidrofílicos e hidrofóbicos podría predecir el grado de desestructuración o desorden de una proteína concreta.

Los métodos computacionales desarrollados hasta la fecha se basan en distintos tipos de cálculos y aproximaciones. Según esto los podemos clasificar en 3 tipos: a) métodos basados en cálculos Ab-initio, donde las predicciones se apoyan solamente sobre la información de la composición de la secuencia proteíca y utilizan técnicas como redes neuronales o clasificadores bayesianos entre otros; b) métodos basados en moldes en los que se examinan estructuras (o no estructuras) de secuencias similares; c) métodos basados en meta-predicciones que combinan las predicciones de varios métodos computacionales.

A continuación nos detendremos en analizar los métodos basados en cálculos Ab-initio. Estos métodos utilizan medidas matemáticas de la composición proteíca. Las regiones desordenadas tienen baja complejidad por lo que los primeros métodos computacionales que se desarrollaron se basaron en medidas matemáticas que distinguen entre regiones de secuencia de proteínas globulares y no globulares. Las estructuras globulares se caracterizan por ser compactas y estar determinadas por secuencias de aminoácidos de alta complejidad. Estas difieren ligeramente de las estructuras no globulares que contienen secuencias al azar aleatorio. Los algoritmos SEG [18], CAST [14] o GBA [10] son métodos en general eficaces para discriminar entre regiones globulares y no globulares de forma automática. Los análisis estadísticos realizados en secuencias de proteínas muestran que aproximadamente una cuarta parte de los aminoácidos forman parte de regiones de baja complejidad y que más de la mitad de las proteínas tienen al menos una de estas regiones.

Otras características tenidas en cuenta en los cálculos Ab-initio son la hidrofobicidad y la carga neta de la proteína. El esqueleto proteíco y las cadenas laterales de las proteínas se mueven constantemente debido al movimiento térmico y a la energía cinética de los átomos. Este movimiento es dependiente del carácter hidrofóbico del segmento proteíco. Los métodos basados en esta propiedad emplean cálculos de la distribución de factores-B en las estructuras cristalinas. Los factores-B reflejan la fluctuación de los átomos sobre sus posiciones promedio y proporcionan información importante sobre la dinámica de la proteína. Estos cálculos usan regresiones vectoriales tipo SVR (Support Vector Regression). Los enfoques computacionales para predecir el movimiento térmico son útiles para el análisis de las propiedades dinámicas de proteínas con estructuras desconocidas y por tanto para PIDs. Las regiones desordenadas son raramente hidrofóbicas y presentan una alta probabilidad de fluctuación.

La carga neta influye en la capacidad de los polipéptidos de una proteína en formar contactos estabilizantes. En las proteínas globulares existen un gran número de interacciones inter-residuos, que aportan la energía estabilizadora para superar la pérdida de entropía durante el plegamiento. Por el contrario, en PIDs las secuencias no tienen la capacidad de formar suficientes interacciones inter-residuos. Las regiones desordenadas muestran una alta carga neta.

Las redes neuronales artificiales son también usadas para desarrollar métodos de predicción de desorden en proteínas. Estos métodos predicen, a partir de una secuencia de proteína, la probabilidad de encontrar segmentos desordenados.

### 13.3.1. Métodos bioinformáticos de predicción de desorden

En la última década más de una veintena de métodos bioinformáticos se han desarrollado para predecir regiones desordenadas o desestructuradas a partir de la secuencia de una proteína (ej. DisEMBL, DISOPRED2, DRIPPRED, DISPro, FoldIndex, GlobProt2, IUPred, PONDR, RONN, SPRITZ, entre otros). La clave del éxito de estos programas bionformáticos reside en que la secuencia de aminoácidos no sólo determina la estructura tridimensional de una proteína, sino también la ausencia de la misma. A continuación se detallan (en un orden arbitrario) algunos de ellos y se muestran, en algunos casos, los resultados que proporcionan en su formato de salida. Es interesante mencionar que estos métodos computacionales se pueden utilizar directamente a través de sus propios servidores web o bien se pueden descargar en servidores locales (según permisos de licencia). Las predicciones obtenidas por cada uno de estos métodos son difíciles de comparar entre sí debido a las diferencias existentes entre los parámetros y/o variables que manejan (ej. factores-B, regiones sin coordenadas en archivos PBD, etc.). Los experimentos CASP<sup>1</sup> son una valiosa herramienta que permite obtener una validación de estos métodos.

---

<sup>1</sup>CASP. <http://predictioncenter.org>

## DisEMBL

DisEMBL<sup>2</sup> es un método basado en redes neuronales artificiales que contempla los siguientes tres criterios para definir las regiones PID. Las regiones PID pueden ser: i) bucles flexibles (*loops/coils*) tal como los define Kabsch y Sander (1983). Los aminoácidos presentes en los bucles flexibles no son frecuentes en regiones estructuradas tales como hélices  $\alpha$  (H), hélices  $3_{10}$  (G) o láminas  $\beta$  (E). Sin embargo, es importante señalar que los bucles flexibles no necesariamente son regiones desordenadas o desestructuradas, aunque el desorden sólo se encuentra en estas regiones flexibles. ii) bucles “calientes” con un alto grado de movilidad determinada a partir de factores de temperatura  $C_\alpha$  (factores-B). Este tipo de bucles son un subgrupo de los bucles flexibles pero con una alta dinámica añadida. iii) coordenadas en estructuras de rayos-X, definidas como entradas REMARK465 en archivos PDB (Protein Data Bank), sin mapa de densidad electrónica. Las regiones ausentes en archivos PDB se consideran por tanto desordenadas o desestructuradas. DisEMBL además proporciona un interfaz de tubería para predicciones a gran escala, esenciales, por ejemplo, a escala de genómica estructural [11].

*Ejemplo:*

La interfaz web es fácil de usar. En primer lugar hay que introducir la proteína problema, bien por su código (SWISS-PROT SWALL, ej. P61313) o entrada (ej. RL15\_HUMAN), o bien por la secuencia de aminoácidos, ver Figura 13.3.

```
>sp|P61313|RL15_HUMAN 60S ribosomal protein L15
MGAYKYIQELWRKKQSDVMRFLLRVRCWQYRQLSALHRAPRPTRPDKARRLGYKAKQGYV
IYRIRVRRGGRKRPVPKGATYGKPVHHGVNQLKFARSLQSVAEERAGRHCALRVLNSYW
VGEDSTYKFFEVILIDPFHKAIRRNPDQTQWITKPVHKHREMRLTSAGRKSRLGLGKGF
HHTIGGSRRAAWRRRNTLQLHRYR
```

**Figura 13.3:** Secuencia de la proteína ribosomal L15 humana.

El resultado de salida muestra: i) una gráfica que representa la probabilidad de desorden a lo largo de la secuencia para cada uno de los criterios que definen las regiones PID (bucle flexibles, bucles “calientes” o entradas REMARK465 sin coordenadas en archivos PDB), Figura 13.4. Las líneas horizontales corresponden al nivel de expectativa aleatoria para cada predicción. ii) tres salidas de texto con las secuencias en las que están marcados los aminoácidos que cumplen cada uno de los criterios.

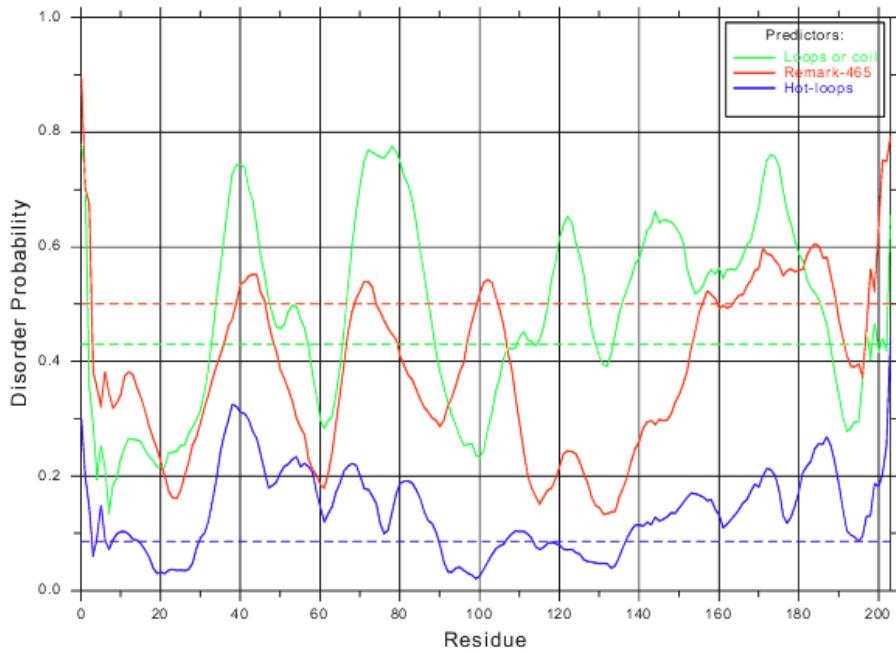
## DISOPRED2

DISOPRED2<sup>3</sup> identifica regiones PID en base a la información obtenida de 750 secuencias no redundantes con estructuras de rayos-X de alta resolución. El desorden se identifica con aquellos aminoácidos que tienen coordenadas sin mapa de densidad electrónica. Para cada proteína problema se genera un perfil de secuencia usando el motor de búsqueda PSI-BLAST contra una base de datos de secuencia filtrada. El vector de entrada para cada aminoácido se construye a partir de los perfiles de una ventana simétrica de 15 posiciones [17]. DISOPRED2 se diferencia de otros métodos en que está entrenado directamente sobre las secuencias proteicas no sobre la composición de aminoácidos.

*Ejemplo:*

<sup>2</sup>DisEMBL. <http://dis.embl.de>

<sup>3</sup>DISOPRED2. <http://bioinf.cs.ucl.ac.uk/disopred>



**Figura 13.4:** Gráfica de salida de DisEMBL para la proteína RL15.

La interfaz web es fácil de usar. En primer lugar hay que introducir la secuencia de aminoácidos de la proteína problema (Figura 13.3).

Los resultados de salida presentan diferentes formatos: i) una gráfica que representa la probabilidad de desorden a lo largo de la secuencia, Figura 13.5. ii) un archivo de texto con la secuencia donde se marcan con un asterisco los residuos desordenados. iii) un archivo de texto con todos los aminoácidos que contiene la secuencia y la probabilidad de desorden asociada.

## FoldIndex

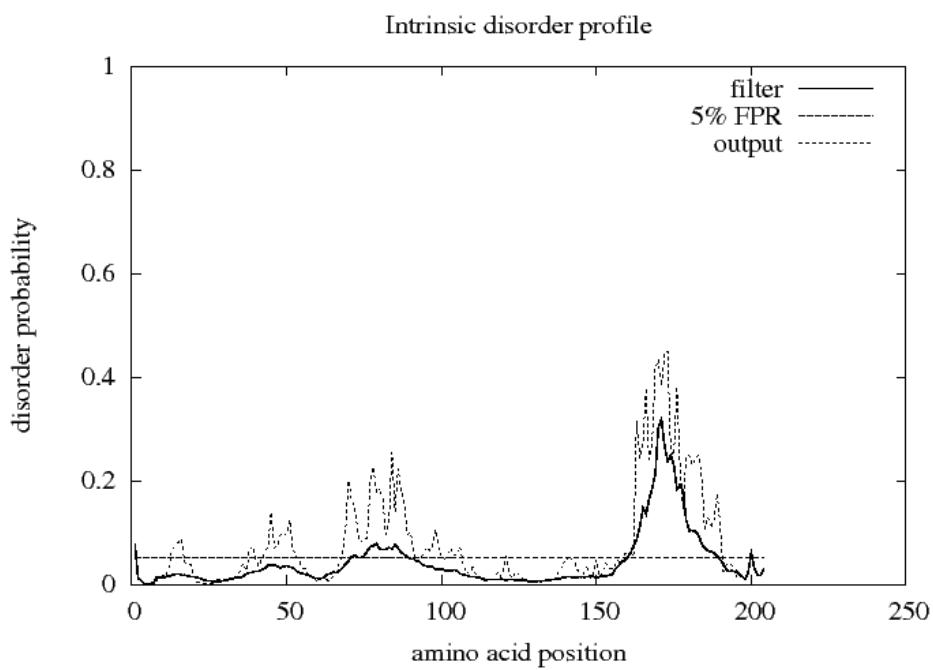
FoldIndex<sup>4</sup> predice si una secuencia de proteína es intrínsecamente desordenada en base al algoritmo propuesto por Uversky y col. (2000) que considera el promedio de la hidrofobicidad de cada residuo y la carga neta de la secuencia. FoldIndex© tiene una tasa de error comparable a la de los más sofisticados métodos de predicción. Usa ventanas correderas que permiten la identificación de grandes regiones dentro de una proteína con plegamientos diferenciables a los de la proteína completa [13].

*Ejemplo:*

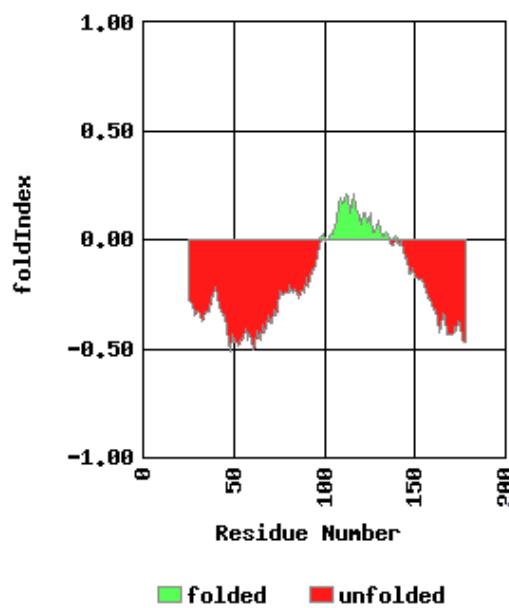
La interfaz web es fácil de usar. En primer lugar hay que introducir la secuencia de aminoácidos de la proteína problema (Figura 13.3).

En la Figura 13.6 se puede ver el tipo de resultados que proporciona este programa.

<sup>4</sup>FoldIndex. <http://bip.weizmann.ac.il/fldbin/findex>



**Figura 13.5:** Gráfica de salida de DISOPRED2 para la proteína RL15.



**Figura 13.6:** Gráfica de salida de FoldIndex para la proteína RL15.

## IUPred

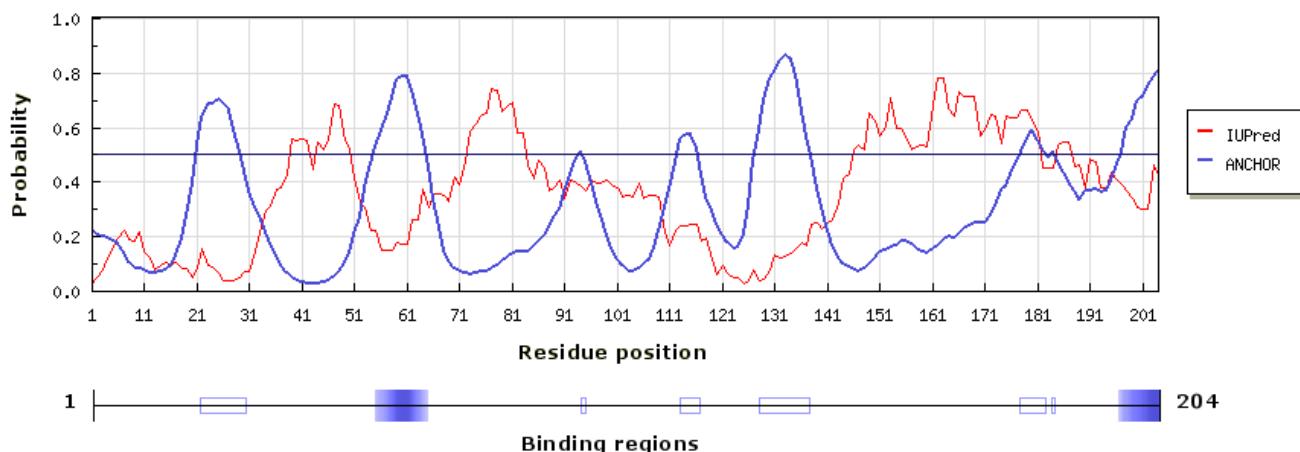
IUPred<sup>5</sup> se basa en la estimación de la capacidad de los polipéptidos en formar contactos estabilizantes. Las PIDs no tienen la capacidad de formar suficientes interacciones inter-residuos. Este método usa una expresión cuadrática para la composición de aminoácidos que tiene en cuenta que la contribución de un aminoácido (tanto ordenado como desordenado) depende no sólo de sus propiedades químicas sino también de su entorno en la secuencia, incluyendo sus potenciales de interacción. Las energías de secuencias estimadas con regiones PID claramente se desplazan hacia energías menos favorables en comparación con proteínas ordenadas [4].

IUPred, a diferencia de otros métodos, ofrece 3 tipos diferentes de predicción: i) segmentos desordenados largos ( $>30$  residuos consecutivos); ii) segmentos desordenados cortos ( $<30$  residuos consecutivos); iii) dominios estructurados. IUPred también incorpora la herramienta ANCHOR que predice regiones de interacción con otras proteínas o moléculas en el segmento de secuencia desordenada. Estas regiones funcionan como una vía de transición desorden-orden durante la interacción con una proteína globular.

Ejemplo:

La interfaz web es fácil de usar. En primer lugar hay que introducir la secuencia de aminoácidos de la proteína problema (Figura 13.3).

Los resultados de salida presentan diferentes formatos: i) una gráfica que representa la probabilidad de desorden a lo largo de la secuencia (Figura 13.7). ii) un archivo de texto con todos los aminoácidos que contiene la secuencia y la probabilidad de desorden asociada. Estos formatos son similares a los que produce DISOPRED2.



**Figura 13.7:** Gráfica de salida de IUPred para la proteína RL15.

## PONDR

PONDR®<sup>6</sup> funciona a partir de datos de secuencia primaria solamente y está basado en cálculos de redes neuronales utilizando ventanas generalmente de 21 aminoácidos. Propiedades como la composición de aminoácidos y la hidropaticidad se calculan sobre estas ventanas, y los valores son entradas del

<sup>5</sup>IUPred. <http://iupred.enzim.hu>

<sup>6</sup>PONDR®. <http://www.pondr.com>

programa. La red neuronal, que está ensayada para conjuntos específicos de secuencias ordenadas o desordenadas, devuelve un valor para el aminoácido central de la ventana. Las predicciones son refinadas sobre un ancho de ventana de 9 aminoácidos [15].

PONDR-FIT<sup>7</sup> incorpora meta-predicciones como resultado de un consenso de predicciones de desorden individuales. Se trata de un método basado en el consenso de redes neuronales artificiales que mejora la exactitud de la predicción en un rango del 3 al 20 % comparado con predicciones individuales. Un análisis de error muestra que la exactitud del método es peor cuando se analizan regiones desordenadas de corta longitud ( $L < 10aa$ ) o amino ácidos localizados en los bordes de los segmentos ordenados/desordenados [19].

## D<sup>2</sup>P<sup>2</sup>

D<sup>2</sup>P<sup>2</sup> es una base de datos de predicciones de PIDs<sup>8</sup>. Incluye las predicciones de una batería de métodos y sus variantes, tales como VL-XT, VSL2b, PrDOS, PV2, Espritz y IUPred sobre las secuencias de proteínas de 1765 genomas completos [12]. Además integra los resultados de todas las predicciones de dominios SCOP (mayoritariamente estructurados) usando la predicción de SUPERFAMILY. Esta base de datos proporciona una visión de la distribución genómica de PIDs y su historia evolutiva.

### 13.4. Composición, distribución y función de las proteínas desordenadas

Las PIDs, en general, se caracterizan por una baja complejidad en su secuencia de aminoácidos. Tienen un bajo contenido en aminoácidos de tipo hidrofóbico (Val, Leu, Ile, Met, Phe, Trp, Tyr), que suelen formar parte del esqueleto de proteínas globulares compactas, y tienen una alta proporción de aminoácidos cargados y polares (Gln, Ser, Pro, Glu, Lys, y ocasionalmente Gly y Ala). Tales regiones son muy frecuentes en proteínas reguladoras transcripcionales (factores de transcripción) que hoy se reconocen como proteínas PID. Los análisis de datos de secuencia en genomas completos indican que las PIDs son altamente prevalentes y que su proporción aumenta con la complejidad de los organismos. La composición de nucleótidos de los genes que codifican para las PIDs tienen un elevado contenido GC (guanina y citosina). En bacterias valores altos de GC resultan en un aumento del contenido de Gly, Ala, Arg y Pro, mientras que contenidos bajos de GC derivan en un enriquecimiento de Phe, Tyr, Met, Ile, Asn y Lys. El primer grupo de aminoácidos se encuentra sobrerepresentado en regiones PID, por lo que es de esperar que valores altos de GC resulten en un aumento significativo de desorden. Este tipo de correlación también se ha observado en organismos eucariotas superiores [21].

Las predicciones realizadas con diferentes métodos computacionales estiman que el 30-60 % de las proteínas en organismos eucariotas contienen segmentos desordenados de longitud superior a 30 aminoácidos. En Archaea y bacterias la prevalencia es menor (2-18 %). Estas predicciones se han realizado utilizando los proteomas completos disponibles [6]. Los porcentajes que se han calculado para los proteomas de plantas no difieren de los determinados en otros eucariotas. Sin embargo, cuando en plantas se examinan por separado los proteomas cloroplásticos, mitocondrial y nuclear se encuentran diferencias. Los proteomas cloroplástico (2-11 %) y mitocondrial (2-19 %) tienen mucho menos desorden que el nuclear, con valores similares a los presentes en Archaea y bacterias, de acuerdo con su origen filogenético. Por otra parte, es interesante señalar que cuando se examina el patrón de transferencia génica entre

<sup>7</sup>PONDR-FIT. <http://www.disprot.org/pondr-fit>

<sup>8</sup>D<sup>2</sup>P<sup>2</sup>. <http://d2p2.pro/>

el cloroplasto y el núcleo durante la evolución, nos encontramos que los genes de origen cloroplástico, que son ahora codificados por genes nucleares, han adquirido desorden. Por tanto, la dinámica evolutiva del núcleo en plantas añade segmentos de desorden, a excepción de las proteínas que son codificadas en ambos genomas, debido posiblemente a restricciones funcionales [20]. Las PIDs también se encuentran en organismos más sencillos como los virus. Los fagos, virus especializados en infectar bacterias, se adhieren a la membrana de una célula huésped mediante proteínas que se mantienen unidas al cuerpo del fago por medio de regiones conectoras flexibles.

Las PIDs desempeñan funciones importantes y básicas en la célula, la mayoría asociadas con procesos de regulación, que incluyen la transcripción, la translación, la transducción de señal, la fosforilación, la regulación del ensamblaje de multicomplejos (ej. ribosoma) donde se requieren interacciones altamente específicas y de baja afinidad, entre otros. Por tanto, las PIDs están mayoritariamente asociadas a funciones reguladoras y de señalización, que son importantes en la comunicación celular y la respuesta celular a diversos estímulos. Estas funciones adquieren un mayor protagonismo en organismos eucariotas donde la complejidad de los sistemas celulares es superior. Esta observación puede explicar la correlación positiva entre desorden y complejidad.

La diversidad funcional de las PIDs complementa a la de las proteínas estructuradas. Las regiones PID participan en interacciones proteína-proteína, en el ensamblaje de complejos multi-proteicos y en múltiples actividades de las proteínas. Las proteínas con actividad chaperona tienen alta proporción de regiones PID. Lo mismo ocurre con las regiones de unión específica a ADN, como son los elementos *cis*, o los *zinc-finger*. Intuitivamente se puede pensar que una mayor flexibilidad y capacidad de interacción entre moléculas confiere a los organismos de una ventaja evolutiva. A mayor complejidad mayor capacidad de establecer interacciones. Esta característica permite la adaptación a diferentes condiciones del entorno, haciendo que la red de interacciones sea menos sensible a cambios ambientales y continúe su normal funcionamiento, facilita también la unión a diversas dianas y a su vez el control sobre la afinidad de esa unión, ajustando de esta forma el tiempo de transmisión de la señal según las necesidades. Las características dinámicas de las proteínas desordenadas aceleran el proceso de unión, lo que puede resultar crucial en las condiciones típicas de baja concentración de proteínas involucradas en procesos de regulación. Así, es importante mencionar la hipótesis según la cual las proteínas desordenadas ayudan a la eficiente propagación de las señales celulares ya que la superficie de interacción (con otra molécula) por longitud de cadena peptídica es muy superior a la encontrada en las proteínas de estructura definida, debido de nuevo a su flexibilidad que da lugar a una cadena más expandida. Es importante matizar que la complejidad de los organismos es un fenómeno en el que intervienen múltiples parámetros, no sólo el desorden, sino también el tamaño de los genomas, la capacidad de regulación por splicing alternativo, número de interacciones potenciales, especificidad de tejido, etc. son a considerar [8, 16].

Las regiones PID también podemos encontrarlas conectando dominios o módulos estructurados, en este caso se trata de fragmentos significativamente largos ( $\geq 30$  aminoácidos) carentes de estructura. En algunos casos estas regiones participan activamente en la función de la proteína entera por estar directamente involucradas en la zona de interacción. En otros casos estas regiones, por un lado, incrementan la movilidad de los dominios estructurados, y por otro, establecen una orientación predeterminada entre ellos, modulando de forma pasiva las posibilidades de interacción de los mismos. Se ha propuesto que los segmentos desordenados de unión inter-dominio incrementan la velocidad con la que se producen grandes cambios conformacionales entre los módulos, facilitando de nuevo la transmisión de señales dentro del entorno.

Para terminar esta sección comentaremos algunos resultados recientes obtenidos de estudios bioinformáticos a nivel de genómica comparada. Estos estudios permiten clasificar las regiones desordenadas en tres tipos: a) regiones donde el desorden está conservado entre organismos pero la secuencia de

aminoácidos rápidamente cambia por procesos evolutivos (desorden flexible); b) regiones de desorden conservado con alta conservación de la secuencia (desorden conservado); c) desorden no conservado. El primer tipo estaría asociado principalmente a rutas de señalización celular y multifuncionalidad y el segundo a procesos de unión a ARN y a proteínas de tipo chaperona. Por el momento se desconoce la relevancia del tercer tipo [3].

### 13.5. Enfermedades asociadas a proteínas desordenadas

Numerosas PIDs están relacionadas con diversas enfermedades, algunas neurogenerativas (alzhéimer y parkinson), cáncer, cardiovasculares, diabetes, encefalopatías espongiformes transmisibles. Mediante estudios bioinformáticos se ha encontrado que el 79 % de las proteínas asociadas con el cáncer contienen regiones desordenadas de más de 30 aminoácidos [9]. Por el contrario, sólo el 13 % de proteínas de un conjunto con estructuras ordenadas bien definidas contenían tales regiones de desorden predichas. La presencia de desorden en varias proteínas relacionadas con el cáncer se ha observado experimentalmente. Algunos ejemplos son: la proteína p53 (que participa en la red de señalización que regula la expresión de genes), AFP (alfa-Fetoproteína que participa en la regulación de la división célula), BRCA1 (proteína de la susceptibilidad al cáncer de mama), miembros de la familia Bcl-2 (implicadas en la muerte celular programada). El ejemplo más estudiado es la proteína supresora de tumores p53 que realiza su función reguladora interaccionando con otras múltiples proteínas. Aproximadamente el 70 % de esas interacciones está mediado por sus regiones desordenadas, bien a través de mutaciones o por cualquier otro factor. Si p53 pierde su función, la célula típicamente se convierte en cancerosa.

Los análisis realizados en otras patologías han revelado resultados parecidos a los obtenidos en cáncer. Estudios bioinformáticos en los proteomas de los virus del papiloma humano indican que las proteínas de los virus considerados de alto riesgo para el desarrollo de carcinomas contienen más regiones desordenadas que las proteínas homólogas en virus no malignos.

En el caso de enfermedades cardiovasculares se ha calculado que un 61 % de las proteínas relacionadas son PIDs. Este porcentaje es próximo al calculado para proteínas de señalización (66 %) y es significativamente más alto que el promedio calculado de PIDs en organismos eucariotas (30-60 %). Este alto porcentaje de PIDs sugiere que podrían ser esenciales para la función de las proteínas que intervienen en los procesos relacionados con esta enfermedad, además de para su control y regulación. Los datos disponibles sobre PIDs en enfermedades cardiovasculares indican que hay una buena correlación entre las observaciones experimentales realizadas y los resultados de predicción.

Una de las características de la diabetes de tipo II (*diabetes mellitus*) es la formación de depósitos amiloïdes en los islotes de Langerhans del páncreas como respuesta a la disminución progresiva de la acción de la insulina. Esto produce un aumento inicial de la cantidad de hormona, aunque posteriormente decrece provocando el aumento de los niveles de glucosa en sangre. La proteína amilina es el componente principal de estos agregados amiloïdes. Se trata de una pequeña proteína totalmente desordenada que sufre un cambio conformacional y adquiere cierto grado de estructura para posteriormente dar lugar a la formación de los depósitos.

Las encefalopatías espongiformes transmisibles se producen por la acumulación de agregados de una proteína llamada Prion. Un ejemplo de este tipo de patología es la encefalopatía espongiforme bovina o enfermedad de las vacas locas. La proteína Prion consta de dos dominios, uno estructurado y otro desordenado. El primero sufre un profundo cambio conformacional que es parcialmente responsable de los fenómenos de agregación. En los procesos de interacción y unión con las proteínas Prion el dominio desestructurado juega un papel importante.

El Alzheimer está asociado a la acumulación de depósitos proteicos con diferentes características morfológicas conocidos como depósitos amiloides, placas seniles y ovillos neurofibrilares. Las placas seniles se generan por la agregación de las proteínas amiloide  $\beta$  y  $\tau$ . La proteína amiloide  $\beta$  antes de la agregación carece de estructura y sufre un proceso de compactación previo a la asociación. En el estado agregado se convierte en neurotóxica. Algo similar le ocurre a la proteína  $\tau$ . Antes de la agregación en ovillos la proteína  $\tau$  es mayoritariamente desordenada y se agrega tras sufrir un proceso de plegamiento parcial. Otras enfermedades neurodegenerativas, como las sinucleinopatías se caracterizan también por la formación de agregados fibrilares, concretamente de la proteína alfa-sinucleína. Se ha comprobado que en condiciones fisiológicas la proteína alfa-sinucleína está desordenada casi en su totalidad. Cuando varían las condiciones de pH y temperatura, la alfa-sinucleína es capaz de variar su conformación adoptando diversos grados de estructuración y de agregación. Los agregados pueden presentar morfologías muy variables; esferas, fibras y cúmulos amorfos.

Las interconexiones entre el desorden intrínseco, la señalización celular y enfermedades humanas sugieren que las enfermedades conformacionales pueden deberse no sólo al plegamiento anómalo de proteínas, sino también a errores en la identificación y la señalización, así como a fenómenos de plegamiento no natural o no nativo. La mayoría de proteínas que conocemos tienen una determinada conformación y a menudo llevan a cabo una única función. Las PID, sin embargo, son multifuncionales. Al no tener prácticamente estructura son muy flexibles lo que les permite interactuar con otras muchas proteínas del organismo, de aquí que ocupen posiciones clave dentro de las células. Entender su dinamismo, saber con qué proteínas se relacionan y saber de qué manera lo hacen es clave para poder avanzar en el diseño de fármacos específicos. En los últimos años se está realizando un importante esfuerzo para descubrir pequeñas moléculas que puedan reconocer proteínas PID y modificar su función. En este sentido las proteínas PID presentan dos problemas críticos para el desarrollo de drogas eficaces. Debido a la plasticidad en su capacidad de unión, una molécula podría unirse a la proteína diana y a varias otras cuya función no se desea alterar. También es posible que la proteína diana reconozca diversas zonas de unión en la misma proteína que modifiquen la función con resultados diferentes. Por tanto se necesitan realizar estudios en profundidad para resolver estas cuestiones.

El descubrimiento de moléculas capaces de inhibir con cierta especificidad la interacción entre las proteínas c-Myc y Max es un hecho esperanzador para avanzar en el diseño de fármacos. Ambas proteínas son factores de transcripción. En particular, c-Myc activa la expresión de hasta el 15 % de los genes humanos. La función de c-Myc afecta por tanto importantes procesos biológicos como la proliferación celular, la diferenciación y la muerte celular programada. La disfunción de c-Myc es responsable de numerosos tipos de cáncer humano. Max es capaz de homodimerizar y de interaccionar con otros factores de transcripción como c-Myc. El complejo c-Myc/Max tiene afinidad por determinadas zonas del ADN y esta unión favorece el proceso de transcripción. c-Myc y Max son proteínas desordenadas en su estado libre que adquieren estructura al unirse entre sí. Conseguir inhibir la formación del complejo y alterar su función es un objetivo importante para futuras terapias contra el cáncer.



### 13.6. Bibliografía

- [1] M. authors. Intrinsically disordered proteins (IDPs). *Chem Rev*, 114(13):6557–6948, 2014.
- [2] F. J. Aviles, G. E. Chapman, G. G. Kneale, C. Crane-Robinson, and E. M. Bradbury. The conformation of histone h5. isolation and characterisation of the globular segment. *Eur J Biochem*, 88(2):363–71, 1978.
- [3] J. Bellay, S. Han, M. Michaut, T. Kim, M. Costanzo, B. J. Andrews, C. Boone, G. D. Bader, C. L. Myers, and P. M. Kim. Bringing order to protein disorder through comparative genomics and genetic interactions. *Genome Biol*, 12(2):R14, 2011.
- [4] Z. Dosztanyi, V. Csizmok, P. Tompa, and I. Simon. Iupred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content. *Bioinformatics*, 21(16):3433–4, 2005.
- [5] M. Drescher. Epr in protein science : intrinsically disordered proteins. *Top Curr Chem*, 321:91–119, 2012.
- [6] A. K. Dunker, Z. Obradovic, P. Romero, E. C. Garner, and C. J. Brown. Intrinsic protein disorder in complete genomes. *Genome Inform Ser Workshop Genome Inform*, 11:161–71, 2000.
- [7] J. H. Fong, B. A. Shoemaker, S. O. Garbuzyntsiy, M. Y. Lobanov, O. V. Galzitskaya, and A. R. Panchenko. Intrinsic disorder in protein interactions: insights from a comprehensive structural analysis. *PLoS Comput Biol*, 5(3):e1000316, 2009.
- [8] W. L. Hsu, C. J. Oldfield, B. Xue, J. Meng, F. Huang, P. Romero, V. N. Uversky, and A. K. Dunker. Exploring the binding diversity of intrinsically disordered proteins involved in one-to-many binding. *Protein Sci*, 22(3):258–73, 2013.
- [9] L. M. Iakoucheva, C. J. Brown, J. D. Lawson, Z. Obradovic, and A. K. Dunker. Intrinsic disorder in cell-signaling and cancer-associated proteins. *J Mol Biol*, 323(3):573–84, 2002.
- [10] X. Li and T. Kahveci. A novel algorithm for identifying low-complexity regions in a protein sequence. *Bioinformatics*, 22(24):2980–7, 2006.
- [11] R. Linding, L. J. Jensen, F. Diella, P. Bork, T. J. Gibson, and R. B. Russell. Protein disorder prediction: implications for structural proteomics. *Structure*, 11(11):1453–9, 2003.
- [12] M. E. Oates, P. Romero, T. Ishida, M. Ghalwash, M. J. Mizianty, B. Xue, Z. Dosztanyi, V. N. Uversky, Z. Obradovic, L. Kurgan, A. K. Dunker, and J. Gough. D(2)p(2): database of disordered protein predictions. *Nucleic Acids Res*, 41(Database issue):D508–16, 2013.
- [13] J. Prilusky, C. E. Felder, T. Zeev-Ben-Mordehai, E. H. Rydberg, O. Man, J. S. Beckmann, I. Silman, and J. L. Sussman. Foldindex: a simple tool to predict whether a given protein sequence is intrinsically unfolded. *Bioinformatics*, 21(16):3435–8, 2005.
- [14] V. J. Promponas, A. J. Enright, S. Tsoka, D. P. Kreil, C. Leroy, S. Hamodrakas, C. Sander, and C. A. Ouzounis. Cast: an iterative algorithm for the complexity analysis of sequence tracts. complexity analysis of sequence tracts. *Bioinformatics*, 16(10):915–22, 2000.
- [15] P. Romero, Z. Obradovic, X. Li, E. C. Garner, C. J. Brown, and A. K. Dunker. Sequence complexity of disordered protein. *Proteins*, 42(1):38–48, 2001.
- [16] E. Schad, P. Tompa, and H. Hegyi. The relationship between proteome size, structural disorder and organism complexity. *Genome Biol*, 12(12):R120, 2011.
- [17] J. J. Ward, J. S. Sodhi, L. J. McGuffin, B. F. Buxton, and D. T. Jones. Prediction and functional analysis of native disorder in proteins from the three kingdoms of life. *J Mol Biol*, 337(3):635–45, 2004.
- [18] J. C. Wootton. Non-globular domains in protein sequences: automated segmentation using complexity measures. *Comput Chem*, 18(3):269–85, 1994.
- [19] B. Xue, R. L. Dunbrack, R. W. Williams, A. K. Dunker, and V. N. Uversky. Pondr-fit: a meta-predictor of intrinsically disordered amino acids. *Biochim Biophys Acta*, 1804(4):996–1010, 2010.
- [20] I. Yruela and B. Contreras-Moreira. Protein disorder in plants: a view from the chloroplast. *BMC Plant Biol*, 12:165, 2012.
- [21] I. Yruela and B. Contreras-Moreira. Genetic recombination is associated with intrinsic disorder in plant proteomes. *BMC Genomics*, 14:772, 2013.



## **Parte IV**

# **Biología estructural de ácidos nucleicos**



## Capítulo 14

# Estructura, plegamiento y evolución del RNA

*Ivan Dotu, Michael Stich y Jacobo Aguirre*

### 14.1. Introducción

La estructura de ácidos nucleicos más famosa es el DNA cromosomal, donde el DNA se encuentra en forma de dos cadenas enlazadas – la célebre doble hélice. Esta estructura formada por pares de bases complementarias permite relativamente pocas interacciones físico-químicas con otras moléculas. Ese no es el caso del RNA que en muchos contextos funcionales se encuentra en forma de cadena sencilla. Por consecuencia, puede dar lugar a la formación de enlaces con otras moléculas o permitir interacciones secundarias y terciarias consigo mismo que representan el *plegamiento espacial* de la molécula. Este plegamiento es responsable de la versatilidad funcional biológica del RNA.

El RNA, por su composición química, es muy similar al DNA y puede ejercer algunas funciones similares. No obstante, mientras el papel del DNA en los organismos actuales es –en primera aproximación– limitado a constituir el genoma, el RNA también puede ejercer un papel catalítico o regulador, como las proteínas. En este sentido *proteínas y RNA son similares a nivel estructural-funcional*. Como *el RNA es la única biomolécula que a la vez puede codificar su genoma y ejercer un papel catalítico*, se ha convertido en una molécula clave para entender la evolución temprana de la vida. En particular, se ha desarrollado la hipótesis de “Mundo RNA” en la que se supone que la vida actual fue precedida por una etapa donde el RNA realizaba tanto las funciones del DNA como las de las proteínas.

Resumiendo, la relevancia del RNA reside no sólo en codificar la información que finalmente puede ser traducida a proteínas, sino en regular y catalizar procesos celulares a través de su conformación espacial, es decir, su estructura.

### 14.2. Tipos de RNA

El RNA aparece en muchos procesos celulares. Quizá el más conocido sea el *RNA mensajero*, mRNA. En este caso, el RNA representa una parte de la información genética que codifica para una o varias proteínas. No obstante, el avance científico permite identificar y entender cada vez más procesos en los

que el papel del RNA es *no-codificante*, y la clasificación y nomenclatura de los tipos de RNA está cambiando constantemente. Un caso particular representan los virus que utilizan RNA para codificar su genoma, los llamados *virus RNA*. En la Tabla 14.1 presentamos una lista no-exhaustiva de tipos de RNA.

Nombre	Abreviatura	Función
<b>RNA en síntesis de proteínas</b>		
RNA mensajero	mRNA	Codificador de proteínas
RNA ribosomal	rRNA	Traducción
RNA de transferencia	tRNA	Traducción
<b>RNA reguladores</b>		
RNA antisentido	aRNA	Regulación de mRNA
RNA largo no codificante	long ncRNA	Varias
Micro RNA	miRNA	Regulación de genes
Small interfering RNA	siRNA	Regulación de genes
Piwi-interacting RNA	piRNA	Defensa contra transposones
Riboswitch		Regulación de un gen en el mRNA
Ribozima		Catálisis de reacciones químicas
<b>RNA en modificación postranscripcional o replicación de DNA</b>		
Espliceosoma	snRNA	Splicing
RNA pequeño nucleolar	snoRNA	Modificación de nucleótidos
Ribonucleasa P	RNase P	Maduración tRNA
Ribonucleasa MRP	RNase MRP	Maduración rRNA, replicación DNA
RNA telomerasa	telRNA	Síntesis de DNA telomérico
<b>RNA “parásito”</b>		
Virus de RNA		Codificador
Viroides		Auto-propagante
RNA satélite		Auto-propagante
Retrotransposón		Auto-propagante

**Tabla 14.1:** Tipos de RNA. Los tipos miRNA, siRNA y piRNA se pueden agrupar como RNA de interferencia, iRNA. El concepto de RNA no-codificante, ncRNA, agrupa los RNA reguladoras y los RNA en modificación postranscripcional o replicación de DNA. También existe el término fRNA, RNA funcional, que frecuentemente es utilizado como sinónimo a ncRNA.

Describimos brevemente algunos tipos de RNA relevantes. El *RNA mensajero* es el RNA codificador por excelencia. Es la transcripción del DNA que posteriormente es traducido en proteínas. El mRNA está caracterizado por su secuencia. Pero en realidad, el mRNA puede plegarse y formar estructuras secundarias que, hasta la fecha, no creemos que tengan función. Una vez en el citoplasma, la propia maquinaria de traducción tiene que incluir un mecanismo para deshacer estas estructuras.

En el proceso de traducción, el mRNA está introducido en el ribosoma. El ribosoma es un complejo macromolecular formado por varias partes y constituido aproximadamente por un tercio de proteínas y dos tercios de RNA, el *RNA ribosomal* (o ribosómico). Ese rRNA es central al proceso de traducción, y la formación de enlaces peptídicos es catalizado por rRNA. El rRNA consiste de varios miles de nucleótidos repartidos en 3 (procariotas) o 4 (eucariotas) fragmentos. El rRNA se encuentra plegado

tridimensionalmente.

El *RNA de transferencia*, tRNA, actúa conjuntamente con el ribosoma y se encarga de traducir un triplete del mRNA en su amino ácido correspondiente. Su longitud típica es de alrededor de 76 nt. Mientras la secuencia que forma el tRNA puede variar entre especies distintas, la molécula tiene una estructura secundaria (y terciaria) similar a una hoja de trébol con cuatro brazos principales, muy conservada entre especies, lo que demuestra que es la estructura espacial de la molécula la que confiere su función.

Como la molécula de RNA puede formar estructuras secundarias y terciarias complejas, similar a las proteínas, también les da la capacidad de catalizar reacciones químicas, al igual que las enzimas. Las moléculas de RNA cuya función principalmente es esa se llaman *ribozimas*, como el *hairpin ribozyme* o el *hammerhead ribozyme*. Son moléculas de varias decenas de nucleótidos.

Por último, los *riboswitches* son partes del mRNA localizadas antes (5'-UTR) o después (3'-UTR) de un gen. Un *riboswitch* cambia su conformación espacial (su forma plegada) si se le une una molécula específica y de ese modo regula la traducción y expresión del gen correspondiente.

### 14.3. Niveles estructurales del RNA

En RNA, tenemos tres tipos de niveles estructurales: la estructura primaria, secundaria y terciaria. Dicha jerarquía estructural es análoga a la descrita para proteínas (Sección 7.6)

La *estructura primaria* de una molécula de RNA está descrita por su secuencia, 5'-GAACGUUG...-3' (ver Sección 7.3). La secuencia es una entidad lineal, con una longitud dada por el número de nucleótidos que la forman y tiene un comienzo (5') y un final (3'). Un RNA de cadena sencilla sin plegar no sería más que una hélice abierta (Figura 14.2). Esa situación es altamente inestable porque las bases de una parte de la molécula pueden formar enlaces con bases de otra parte. Este proceso se llama *plegamiento* y da lugar a la estructura secundaria y estructura terciaria.

La *estructura secundaria* es un estado intermedio del plegamiento y caracterizado por la formación de pares de bases (sobre todo pares del tipo Watson-Crick) y bucles. Una estructura secundaria puede ser descrita en un plano (bidimensional).

La *estructura terciaria* representa el estado final del plegamiento que incluye posibles interacciones entre partes cercanas y/o lejanas de la molécula. La formación de enlaces va más allá de simples pares tipo Watson-Crick y la descripción de la molécula es tridimensional.

La *estructura secundaria* puede estudiarse experimentalmente utilizando métodos enzimáticos o de modificación química, que utilizan compuestos químicos como DMS, ketoal o NMIA, que reaccionan específicamente sobre los nucleótidos desapareados y la estructura terciaria a través de cristalográfia de rayos X o de resonancia magnética de la sustancia cristalizada (ver sec:Macromoleculas:MetodosExperimentalesEstructura). En la Sección 14.4 veremos el proceso de plegamiento en más detalle.

#### 14.3.1. Composición química y estructura primaria

El ácido ribonucléico (RNA, por sus siglas en inglés, *ribonucleic acid*) es una de las biomoléculas más importantes. Su composición es similar al ácido desoxirribonucléico (DNA) y su amplio abanico de funciones incluye propiedades del DNA y de las proteínas, lo que convierte a esta molécula en una de las biomoléculas más versátiles. Para distinguir moléculas de cadena simple y doble (sobre todo

en el contexto de virus de RNA), se usa la nomenclatura ssRNA (*single-stranded RNA*) y dsRNA (*double-stranded RNA*).

En el caso más simple y relevante, el RNA está formado por una cadena sencilla de azúcar (*ribosa*) y un *grupo fosfato* a la que se unen cuatro tipos de *bases nitrogenadas*, adenina, citosina, guanina y uracilo, abreviados por las primeras letras de sus nombres, A, C, G y U. Mientras A y G son purinas (con una estructura de dos anillos heterocíclicos fusionados), C y U son pirimidinas (con un anillo heterocíclico) (ver Sección 7.3). En la naturaleza existen de forma minoritaria (pero en moléculas importantes como el tRNA) *modificaciones de las bases canónicas* de las cuales mencionamos algunas: pseudouridina ( $\Psi$  o P), dihidouridina (D), inosina (I), 7-metilguanosina (m<sup>7</sup>G o 7). Para ver una lista más completa con sus abreviaturas [65] y para otras notaciones (p. ej. Y pirimidina, R purina), ver las recomendaciones de la International Union of Biochemistry<sup>1</sup>.

La unión formada por la base y la ribosa se llama *nucleósido*, y si añadimos además el grupo fosfato, se llama *nucleótido*. Una molécula de RNA está completamente caracterizada por la secuencia de las bases. La secuencia de la molécula coincide con su *estructura primaria*.

Por el posicionamiento relativo de base, ribosa y grupo fosfato, una secuencia no es idéntica a su secuencia inversa. Por lo tanto, una cadena sencilla de RNA tiene un comienzo, denominado 5', y un final, 3'. En la Figura 14.1 se muestra de forma esquemática la secuencia 5'-CGAU-3', la posición de la ribosa, del grupo fosfato y de la base en el RNA. La unión entre grupo fosfato y la ribosa es llamado *esqueleto azúcar-fosfato*. Los nucleótidos forman una cadena al tener enlaces fosfodiéster entre ellos. Cada grupo fosfato tiene una carga negativa neta. Para estabilizar la cadena, el medio suele tener iones positivos (p. ej. potasio, magnesio, sodio).

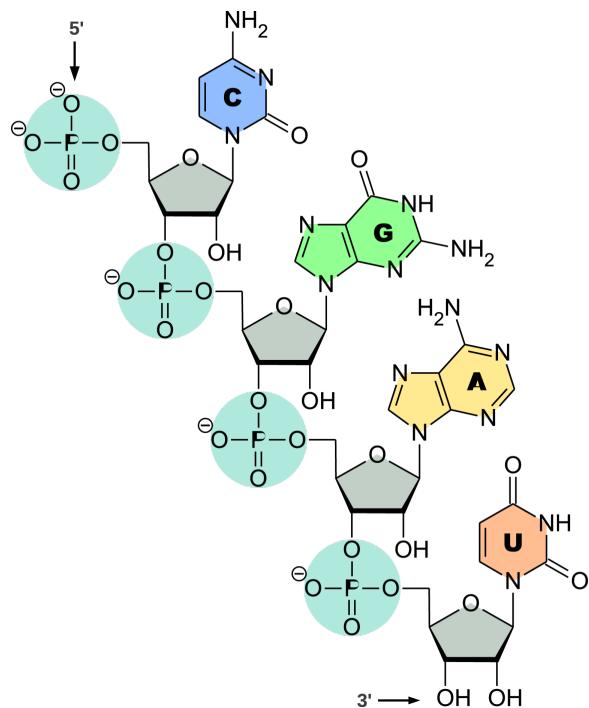
El RNA, igual que el DNA, es un *polinucleótido*, y puede ser interpretado como un polímero lineal y aperiódico, cuyos elementos, los monómeros, son los nucleótidos. El enlace covalente que proporciona la conexión entre los monómeros es un *enlace fosfodiéster* que forma una unión éster entre el grupo OH del carbono 3' de la ribosa en el monómero anterior con el ácido fosfórico y otra unión éster entre el ácido fosfórico y el carbono 5' de la ribosa en el monómero posterior, resultando en el enlace fosfodiéster 3'-5' que incorpora todo el grupo fosfato. Entre un enlace y el siguiente se forma un ángulo y en consecuencia una cadena simple de RNA gira alrededor de su eje central y forma una *hélice dextrógira* (como se muestra en la Figura 14.2). Si además el RNA forma pares de bases (véase Sección 14.3), la hélice resultante se llama A-RNA (RNA-11), que tiene 11 nucleótidos para formar una giro entero.

Hay dos diferencias fundamentales entre RNA y DNA a nivel de la composición (Figura 14.2). Mientras en el RNA el azúcar es la ribosa, en el DNA es la desoxirribosa, que se diferencia de la ribosa en tener un H en la posición 2' de la pentosa, donde el RNA tiene un OH, lo que confiere una mayor estabilidad al DNA en comparación al RNA. La segunda diferencia importante es la base uracilo que en el DNA está reemplazado por la timina (también una pirimidina). Además, la doble hélice estándar de RNA se encuentra en forma A, la del DNA en forma B.

#### 14.3.2. Estructura secundaria

Las bases de una secuencia de RNA pueden formar *enlaces de hidrógeno* entre sí, formando un par de bases, siendo los más importantes los pares de bases tipo Watson-Crick (o *canónicos*), G-C, C-G, A-U y U-A, pero también los pares *wobble* G-U y U-G. En la Figura 14.3(a) se ve que el par G-C tiene tres enlaces de hidrógeno, y los pares A-U y G-U dos. Dadas esas posibilidades de emparejamiento,

<sup>1</sup>Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences. <http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html>

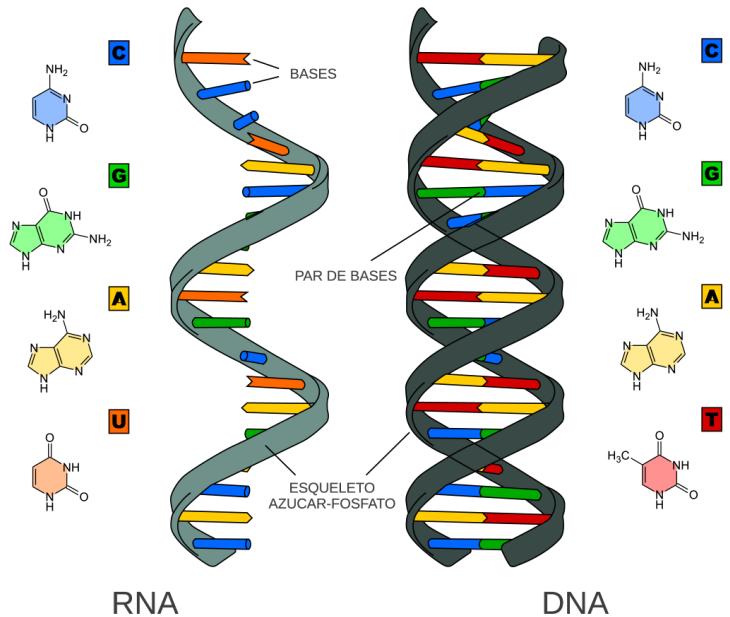


**Figura 14.1:** Composición química del RNA. Se muestra de forma esquemática la secuencia 5'-CGAU-3', la posición de la ribosa (en gris), el grupo fosfato (azul) y de la base en el RNA (colores distintos según la base). Se distinguen claramente las purinas (dos anillos) de las pirimidinas (un anillo). El enlace entre nucleótidos es un enlace fosfodiéster. Las cargas negativas están compensadas por iones positivos en el medio (p. ej. K<sup>+</sup>, Mg<sup>2+</sup>, Na<sup>+</sup>).

dos nucleótidos elegidos al azar forman un par (son compatibles) con una probabilidad del 37.5 %. No obstante, en secuencias reales –y por lo tanto no aleatorias– el número de bases que forman pares en la estructura secundaria suele ser más alto, p. ej., en un tRNA, alrededor del 60 %. La formación de pares de bases está esquemáticamente dibujado en la Figura 14.3(b). En general, se forman zonas con varios pares de bases seguidos, formando así un *apilamiento* o *stack*. La longitud de un *stack* se mide en número de pares de bases consecutivos, bp, por su siglas en inglés *base pairs*. Si un *stack* es suficientemente largo, forma localmente una estructura de doble hélice, también llamado *dúplex*.

Una observación fundamental en el plegamiento del RNA es que si la molécula forma enlaces consigo mismo, necesariamente tiene que formar por lo menos un bucle para acercar los nucleótidos suficientemente (véase la Figura 14.3(b)). Ese bucle se llama bucle terminal aunque es más común llamarlo *bucle horquilla* o *hairpin loop*. Una molécula necesita espacio para realizar este giro, lo que implica que hay por lo menos 2 o 3 nucleótidos dentro del bucle que no pueden participar en la formación de un par de bases. Este argumento también prohíbe la formación de pares de bases entre bases vecinas. En una estructura secundaria un nucleótido o está sin aparear o forma parte de un sólo par. Las bases sin aparear que no se encuentran en un bucle, son llamadas exteriores.

Para describir una estructura secundaria, se suele utilizar puntos y paréntesis. Un punto “.” refleja una base sin aparear, un paréntesis “(” una base apareada con una base hacia el 3' de la secuencia, y un paréntesis “)” una base con una pareja hacia el 5' (existen otras notaciones, pero aquí nos limitamos a esta). El número de paréntesis abiertos y cerrados tiene que ser idéntico y obviamente sólo pueden cerrarse paréntesis que hayan sido abiertos anteriormente. A muchos efectos, además, se considera una estructura secundaria ser libre de *pseudonudos* (*pseudoknots*), es decir, pares entrelazados están



**Figura 14.2:** RNA y DNA. El RNA (izquierda) tiene la base uracilo en vez de la base timina del DNA (derecha). El RNA contiene ribosa, el DNA desoxirribosa y por lo tanto el esqueleto azúcar-fosfato tiene propiedades físico-químicas distintas. Mientras el DNA típicamente está presente como cadena doble, el RNA suele aparecer como cadena sencilla. La imagen es esquemática ya que en realidad existen un surco mayor y menor en la doble hélice. Fuente: Wikipedia.

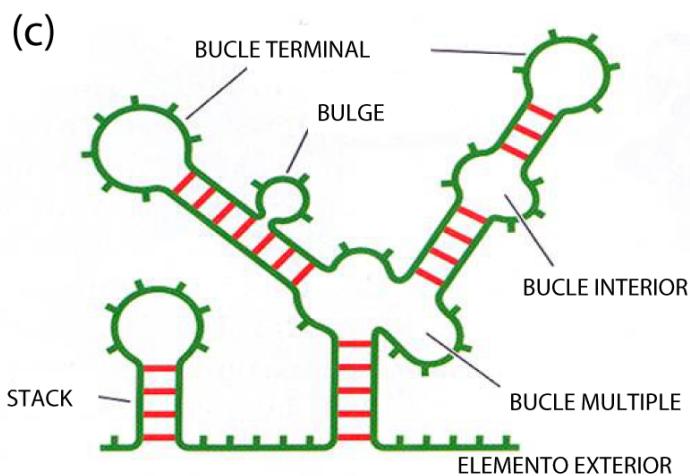
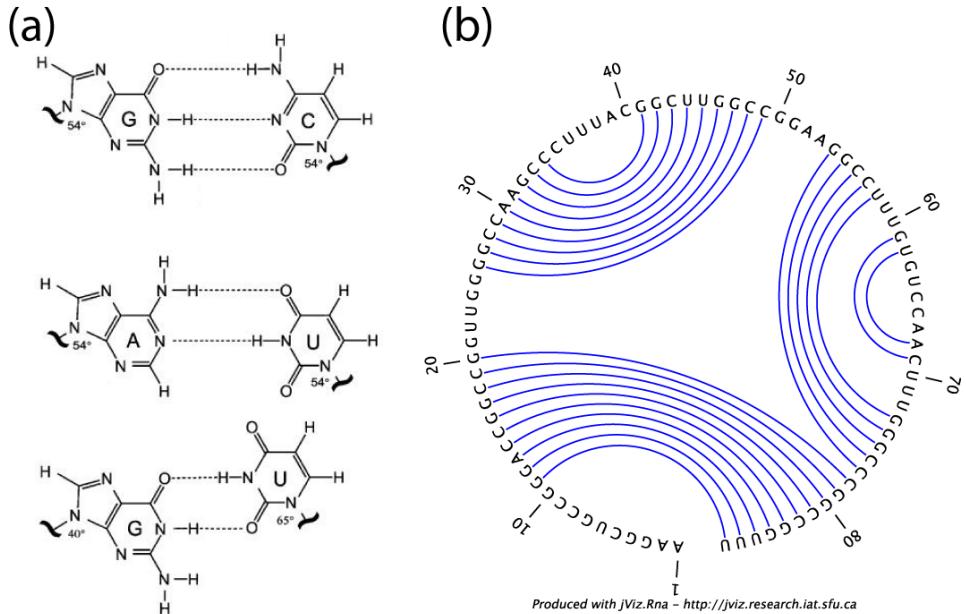
prohibidos (las líneas en la Figura 14.3(b) no se pueden cruzar.)

En la Figura 14.3(c), presentamos los elementos fundamentales de estructuras secundarias. Además del bucle terminal, limitado por un sólo par de bases, también existen *bulges* y bucles interiores, flanqueados por pares de bases distintas. Un bucle donde confluyen más de dos *stacks*, se llama *bucle múltiple*. Existen varios esquemas para clasificar estructuras secundarias en términos de esos elementos estructurales. Es obvio que la estructura secundaria más simple está formada por un *stack* y un bucle terminal. Esa estructura se llama *stem-loop* aunque también es llamada *hairpin* en ciertos contextos.

### 14.3.3. Estructura terciaria

La formación local de pequeñas hélices y bucles es central en el plegamiento del RNA, pero en muchos casos, sobre todo para moléculas grandes, no describe la estructura definitiva tridimensional (terciaria) satisfactoriamente. Describimos tres tipos de interacciones que van más allá de la estructura secundaria: formación de pares de bases no-canónicas, apilamiento de hélices, y pseudonudos.

Analizando estructuras reales, se ha encontrado muchos pares de bases que no son pares canónicos (o *wobble*, en este contexto): ej. en rRNA sólo dos tercios de todos los pares de bases reales son pares canónicos y *wobble* [70]. También se puede dar la situación que lo que parece un bucle interior, en realidad es una hélice con pares no-canónicas (ej. el *E loop*). Para explicar estos resultados tenemos que considerar la extensión tridimensional y la posición real (Figura 14.1 y Figura 14.2) de todos los componentes que forman el RNA, es decir, el grupo fosfato, la ribosa y la base nitrogenada. En general, un nucleótido tiene tres lados de interacción: El lado “Watson-Crick” (WC) es el lado estándar, pero también se observan enlaces vía el lado “Hoogsteen” o el lado de la ribosa (*sugar edge* en inglés).



**Figura 14.3:** Estructuras secundarias. (a) Pares de bases Watson-Crick G-C (arriba), Watson-Crick A-U (centro) y *wobble* G-U (abajo). Figura basada en la Fig. 1 de [74]. (b) Representación circular de la estructura secundaria. Las líneas curvadas indican pares de bases. (c) Elementos de estructuras secundarias. Los pares de bases (líneas rojas) se pueden apilar y forman *stacks*. Las bases sin aparear aparecen como parte de elementos externos o dentro de bucles. Entre los bucles se distinguen los bucles terminales (en inglés, *hairpin*), con un par que cierra el bucle, los *bulges* y bucles interiores, que tienen dos pares que cierran el bucle, y los bucles múltiples, que tienen más que dos pares que cierran el bucle. *Bulges* son bucles con bases sin aparear en un solo lado de la cadena, bucles interiores tienen bases sin aparear en los dos lados.

Cada uno de los seis posibles tipos de enlace entre dos nucleótidos existe en dos variantes, según la orientación (*cis* o *trans*) de los enlaces entre base y azúcar. Como resultado hay 12 combinaciones posibles de formación de pares de bases entre dos nucleótidos en estructuras terciarias. Los pares canónicos G-C, C-G, A-U, U-A, y *wobble* todos pertenecen a un sólo grupo: WC-WC en *cis*. Pero existen datos experimentales de todas las clases [70]. Además, también se dan enlaces de hidrógeno entre 3 o 4 nucleótidos a la vez, llamados triplexes y cuadruplexes.

Otro efecto observado en estructuras terciarias es el apilamiento coaxial de diferentes hélices, en inglés *coaxial* o *helical stacking*. Es energéticamente favorable que diferentes hélices se colocan de forma paralela y apilada a pesar de que eso aparentemente distorsiona la estructura secundaria. Más abajo veremos un ejemplo en el tRNA.

Los *pseudonudos* son otro elemento importante de estructuras terciarias: Se dan cuando nucleótidos sin aparear en la estructura secundaria forman pares entrelazándose con hélices de la estructura secundaria, violando la condición dada arriba. En ese caso, las líneas que describen los pares se cruzan en la representación circular de estructuras secundarias (Figura 14.3(b)).

Otros motivos estructurales terciarios importantes son *kissing hairpins* (las bases de dos bucles terminales distintos forman pares), el motivo *A-minor*, el receptor tetrabucle, o el *ribose zipper*. Para más información, ver [10, 59].

#### 14.3.4. Ejemplo estructura tRNA

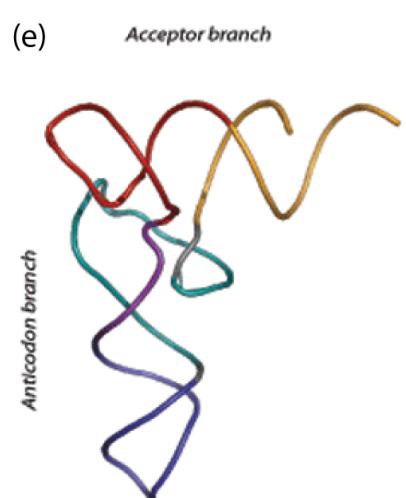
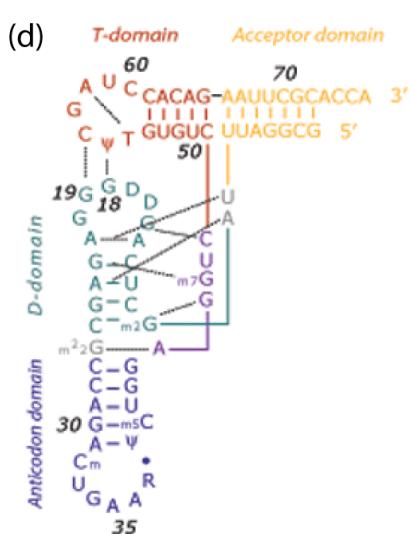
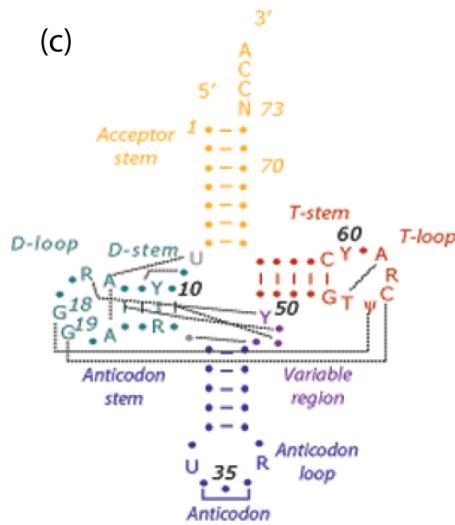
El tRNA es una molécula central para la traducción de la información genética, con una estructura muy conservada entre todos los organismos vivos. En la Figura 14.4(a) vemos una secuencia de tRNA. Además de las bases canónicas, vemos algunas bases modificadas, típico para el tRNA. En (b) está la estructura secundaria en notación punto-paréntesis. En (c) mostramos la estructura secundaria del RNA como imagen planar: las líneas cortas indican un par de base Watson-Crick o *wobble*. La estructura se conoce como hoja de trébol y está formada por cuatro *stacks*, tres bucles terminales y un bucle múltiple. Pero de hecho, el diagrama incluye ya interacciones terciarias (indicadas por las líneas largas). Para apreciar mejor la configuración espacial de la estructura, la dibujamos en (d) desde el lado. Las líneas largas de color indican por donde sigue la secuencia, las líneas negras indican interacciones terciarias. Finalmente, en (e) la representación tridimensional del tRNA basada en datos experimentales con su forma real de “L”. Tanto las hélices T y *Acceptor*, como D y *Anticodon* están apiladas, siendo ejemplos de apilamiento coaxial. También existe una interacción bucle-bucle de los nucleótidos 18 y 19 con 55 y 56, y una interacción terciaria AT en el bucle T. Mientras la estructura secundaria considera como apareados sólo 41 de 76 nt, de hecho 72 nt participan en interacciones terciarias (incluyendo apilamiento coaxial).

#### 14.3.5. Arquitectura del RNA

Métodos modernos ofrecen una vista mucho más completa y detallada de moléculas de RNA largas, de complejos RNA-proteínas, virus RNA, etc. Sin renunciar a la descripción clásica como estructura secundaria o terciaria, es útil describir y clasificar moléculas largas a través de los motivos estructurales que la forman, donde un motivo puede ser una simple hélice o elementos más complejos como *ribose zipper* o hélices apiladas. Esta clasificación en motivos estructurales o funcionales (muchas veces recurrentes) y la identificación de esos motivos como los portadores evolucionados de la función biológica es un campo de investigación en rápido desarrollo.

(a) GCGGAAUUAU CUCAGDDGGGA GAGCCMCCAGABUGAAR . P?UGGAG7UCU CUG UGT PCGAUCCACAGAAUUCGCCACCA

(b) (((((((..((((.....))))).((((.....)))))).....((((.....))))))))....



**Figura 14.4:** Representaciones de secuencia, estructura secundaria y terciaria de un genérico tRNA<sup>Phe</sup>. (a) Secuencia. (b) Estructura secundaria en notación de puntos y paréntesis, sin interacciones terciarias. (c) La estructura secundaria dibujada en el plano, con líneas discontinuas indicando interacciones terciarias. (d) La estructura dibujada desde otro ángulo para poder ver mejor la forma real de “L” de la molécula. (e) Estructura terciaria tridimensional. Datos obtenidos de [50].

## 14.4. Plegamiento de RNA

### 14.4.1. Aspectos generales del plegamiento de RNA

El RNA de cadena sencilla puede formar enlaces consigo mismo, formando pares de bases entre nucleótidos compatibles. En general, la formación de enlaces es energéticamente favorable y de forma natural aparece una molécula con múltiples enlaces consigo mismo, representando una estructura molecular tridimensional que tiene propiedades físico-químicas distintas a la secuencia sin plegar, confiriendo funciones biológicas muy bien definidas en el entorno celular. En consecuencia, este llamado *plegamiento* de la molécula es uno de los procesos más importantes en este contexto y la predicción fiable de una estructura de RNA partiendo de su secuencia es el llamado *folding problem*. Para más información ver [60].

En el plegamiento, se habla de tres tipos de estructuras: la estructura primaria es la secuencia sin plegar (se representa por una secuencia lineal de letras con alfabeto ‘ACGU’). La estructura secundaria refleja el apareamiento local de bases compatibles (Watson-Crick y *wobble*) y que siempre puede ser representado por una imagen bidimensional. Finalmente, la estructura terciaria representa la configuración espacial actual de la molécula. Frecuentemente, y sobre todo para moléculas largas, pueden estar presentes interacciones terciarias que son aquellas enlaces que van más allá de las reglas de apareamiento de bases Watson-Crick (véase Sección 14.3).

#### Estructura secundaria de energía libre mínima

El plegamiento de una molécula RNA es un proceso que depende de muchos factores (temperatura, concentración de iones, presencia de otras moléculas de RNA, etc.), pero el mecanismo fundamental es que nucleótidos complementarios pueden dar lugar a la formación de pares de bases. Es importante notar que para una secuencia dada, existe un gran número de estructuras compatibles diferentes, con un número de enlaces distintos y en posiciones distintas, entonces ¿cuál es la estructura real?

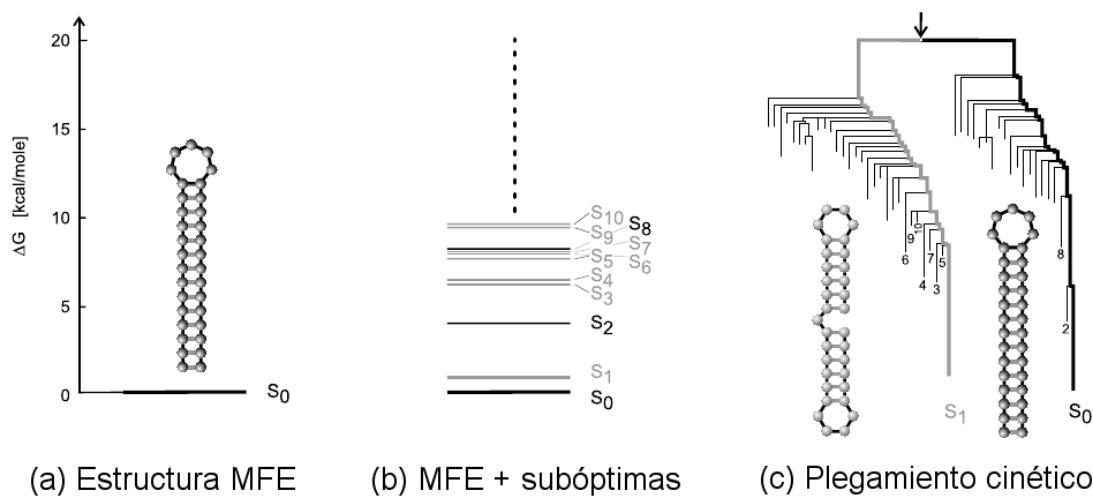
Por *estructuras compatibles* entendemos todas las estructuras que se pueden formar bajo las reglas de formación de pares mencionadas en la Sección 14.3 (ej. Watson-Crick plus *wobble*, bucles terminales de tamaño mínimo 3, bases que participan como mucho en un par, etc.). Pero, p. ej., un par G-C con tres enlaces de hidrógeno entre G y C implica una bajada de energía libre más grande que un par A-U o G-U. Por lo tanto, a las distintas estructuras compatibles están asociadas diferentes energías libres. En primera aproximación, la estructura secundaria observada es aquella que proporciona la bajada de energía libre total más grande, llamada *estructura MFE*, por *minimum free energy*. Por lo tanto, el problema de plegamiento es en gran medida un problema de minimización de energía.

Experimentalmente, podemos determinar las contribuciones energéticas de las distintas partes de la estructura: en general, los enlaces estabilizan, y los bucles desestabilizan la estructura. No obstante, es la formación de apilamientos la que representa la mayor parte de la bajada en energía libre en el plegamiento. Aunque permitida y teóricamente relacionada con una bajada de energía libre, la formación de pares sueltos no es muy estable en un entorno celular y poco frecuente en moléculas naturales.

#### Plegamiento termodinámico y cinético de la estructura secundaria

Hemos abogado por el principio de *minimización de energía libre* para explicar la estructura secundaria. Ese argumento implica que consideramos el conjunto de estructuras en el equilibrio termodinámico

(tiempo de plegamiento  $t \rightarrow \infty$ ), y además a temperatura cero, ya que no consideramos las estructuras con mayor energía, las llamadas subóptimas. Si la temperatura es distinta a cero, la probabilidad de encontrar la estructura MFE o cualquier otra se puede calcular a través de la función de partición que sigue una distribución de Boltzmann. Además, en realidad, el plegamiento no dura infinitamente, y por lo tanto el *estado real* de la molécula plegada no tiene por qué ser idéntico ni al de mínima energía ni a uno de los estados cercanos. De hecho, en muchos casos, las estructuras encontradas en secuencias naturales, no corresponden al estado de mínima energía. Eso no sólo es debido a factores externos (ej. presencia de otras moléculas) sino al proceso cinético del plegamiento mismo. En la Figura 14.5 comparamos las diferentes nociones de estructura secundaria. Vemos que la estructura MFE  $S_0$  y la siguiente estructura en energía  $S_1$  pertenecen a dos cuencas distintas de estructuras, separadas por una barrera alta de energía libre. Por lo tanto la estructura observada puede ser la  $S_1$  si la secuencia al plegarse se queda atrapada en la cuenca correspondiente.



**Figura 14.5:** Comparación entre la estructura MFE (a), el conjunto de todas las estructuras, incluyendo las subóptimas (b) y el conjunto desde el punto de vista cinético (c). Figura basada en la Fig. 15 de [60].

### Plegamiento de estructura terciaria

La formación de pares de bases tipo Watson-Crick y por lo tanto la formación de la estructura secundaria es un proceso rápido que ocurre en una escala de milisegundos. Las interacciones que dan lugar a los motivos estructurales terciarios (Sección 14.3) actúan a una escala de tiempo más lenta. Hay evidencia experimental de que la estructura secundaria del RNA se forma independientemente de la estructura terciaria [7]. Con estos datos y con datos de resonancia magnética [6], se cree que *el RNA se pliega de forma jerárquica* [15], si bien es cierta que existen excepciones [77, 78].

Por lo tanto, la estructura secundaria es un estado intermedio y forma una subestructura de la estructura final. En general, una gran parte de la energía libre de la estructura terciaria final corresponde a la estructura secundaria. Por lo tanto, estudiar el plegamiento de secuencia a estructura secundaria es el primer paso a entender el proceso global.

#### 14.4.2. Predicción de estructuras de RNA

Desde el punto de vista bioinformático, la estructura del RNA se divide en tres niveles: primaria, secundaria y terciaria (ver Sección 14.3). Como la estructura primaria es la secuencia misma, el problema de predicción de estructuras de RNA se refiere a la predicción de estructuras secundarias y terciarias, resultantes del plegamiento de la secuencia (Subsección 14.4.1).

La estructura secundaria del RNA se refiere a la colección de pares de bases de los que se compone. Por motivos de complejidad computacional, se requiere que la estructura secundaria sea planar (ver Figura 14.3(b)) y que cada nucleótido sólo pueda formar parte de un par de bases.

Formalmente, una *estructura secundaria*  $S$  sobre una secuencia de RNA  $s_1, \dots, s_n$  se define como un conjunto ordenado de pares que se corresponde con las posiciones de pares de bases, y que satisface las siguientes restricciones:

1. Watson-Crick o GU *wobble* pares: Si  $(i, j)$  pertenece a  $S$ , el par de bases  $(s_i, s_j)$  tiene que ser uno de los siguientes pares de bases canónicos: (A,U), (U,A), (G,C), (C,G), (G,U), (U,G).
2. Umbral (tamaño mínimo de un bucle): Si  $(i, j)$  pertenece a  $S$ , entonces  $j - i > \theta$  (donde  $\theta$  toma el valor 3).
3. Prohibición de pseudonudos: Si  $(i, j)$  y  $(k, l)$  pertenecen a  $S$ , no se puede dar el caso en que  $i < k < j < l$ .
4. Prohibición de tripletes: Si  $(i, j)$  y  $(i, k)$  pertenecen a  $S$ , entonces  $j == k$ .

La *estructura terciaria* se refiere a la estructura tridimensional de una molécula de RNA, es decir, a las posiciones de cada uno de sus nucleótidos en tres dimensiones (o a una colección de coordenadas tridimensionales) y engloba todas las interacciones terciarias (Sección 14.3 y Subsección 14.4.1).

Es importante mencionar que la mayoría de algoritmos de predicción de estructura de RNA se centran en las posiciones de sus nucleótidos, aunque en algunos casos se encuentran algoritmos que tratan de predecir la posición de todos sus átomos.

#### Estructuras secundarias

La predicción de estructura secundaria de RNA es uno de los problemas más recalcitrantes de la biología computacional de RNA. Muchos algoritmos basados en distintos conceptos computacionales y/o físicos han sido desarrollados para tratar de resolverlo. Esta sección está dedicada a explicar los más utilizados a lo largo de la historia.

En el apartado anterior hemos definido la estructura secundaria, en términos generales, como una colección de pares de bases, sujeta a una serie de restricciones. Siguiendo esta definición, nos encontramos con el primer algoritmo relevante, desarrollado por Nussinov [48] en los años 70. Este algoritmo pionero trata simplemente de *maximizar el número de pares de bases sin violar ninguna de las restricciones*. Se trata de un algoritmo de programación dinámica, definido por las siguientes recursiones:

$$D_{i,j} = \max \begin{cases} \max_{i < k < j} D_{i,k} + D_{k+1,j} \\ D_{i+1,j-1} + w_{i,j} \\ D_{i+1,j} \\ D_{i,j-1} \end{cases} \quad (14.1)$$

donde  $w_{i,j} = 1$  si las posiciones  $i$  y  $j$  corresponden a los pares de bases canónicos (A,U), (U,A), (G,C), (C,G), (G,U), (U,G), o 0 en caso contrario.

Este algoritmo fue posteriormente refinado y es ahora conocido como el *algoritmo Nussinov-Jacobson* [47]. Las recursiones simplificadas son las siguientes:

$$D_{i,j} = \max \begin{cases} D_{i,k} + D_{k+1,j} & \text{donde } i \leq k \leq j \\ D_{i+1,j-1} + w_{i,j}, \end{cases}$$

con la siguiente inicialización:

$$D_{i,j} = \max \begin{cases} D_{i,i} = 0 & \forall i = 1..n \\ D_{i,i-1} = 0 & \forall i = 2..n \end{cases}$$

donde  $n$  es la longitud de la secuencia. Este algoritmo no genera necesariamente la estructura más estable, y, de hecho, su eficiencia cuando se compara con estructuras reales de RNA es muy baja. Sin embargo, es importante explicarlo ya que supuso un gran paso adelante en el área de predicción de estructura secundaria.

El siguiente grupo de algoritmos está basado en conceptos termodinámicos, más concretamente en *buscar la estructura con mínima energía libre*. El primer concepto es el *modelo del vecino más cercano* (*Nearest Neighbor* en inglés). En vez de atribuir energías a pares de bases, asigna energías a ‘pares de pares’ de bases. Las contribuciones de las combinaciones distintas han sido determinadas mediante técnicas experimentales [45, 76].

En un siguiente nivel, nos encontramos con una serie de motivos estructurales que aportan una energía libre a la molécula completa: bucle terminal, *bulge*, bucle interior y bucle múltiple (comparar con la Figura 14.3(c)). Dicha energía libre ( $G^\circ$ ) está compuesta por dos términos: *entalpía* ( $H^\circ$ ) y *entropía* ( $S^\circ$ ), relacionados por la siguiente ecuación<sup>2</sup>:

$$G^\circ = H^\circ - TS^\circ \quad (14.2)$$

donde  $T$  es la temperatura.

Así pues, nos encontramos en la situación en la que queremos calcular la estructura secundaria que minimiza la energía libre. El primer algoritmo para predecir la estructura de mínima energía libre fue desarrollado por Zuker [80]. Al igual que en el caso anterior, se trata de un algoritmo de programación dinámica definido por una serie de recursiones. Estas recursiones, son, sin embargo, un tanto más complejas, y requieren una introducción más detallada.

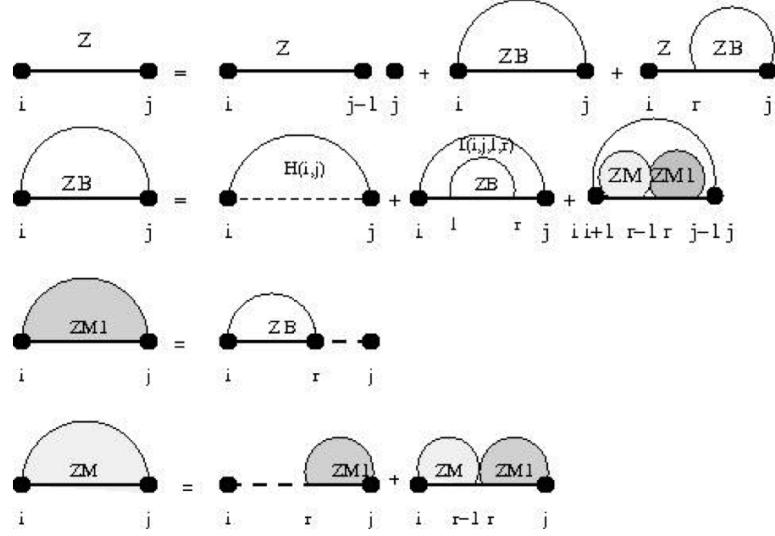
Por lo tanto, es más adecuado explicar las recursiones para calcular la función de partición. La *función de partición* es un concepto de física estadística que describe las propiedades de un sistema en equilibrio termodinámico. Es sabido que un sistema en equilibrio termodinámico con una reserva de temperatura tiene probabilidades  $p$  de ocupar un estado con energía  $E$  multiplicado por su correspondiente factor de Boltzmann. Sea  $\mathcal{X} = \{X_1, \dots, X_n\}$  un sistema de estados, donde el estado  $X_i$  tiene una energía  $E_i$ . El sistema sigue una *distribución de Boltzmann* con temperatura  $T$  si y solo si

$$Pr[X_i] = \exp(-\beta E_i)/Z \quad (14.3)$$

donde  $Z = \sum_i \exp(-\beta E_i)$  y  $\beta = (RT)^{-1}$ .

---

<sup>2</sup>Simplificando la ecuación correcta:  $\Delta G^\circ = \Delta H^\circ - T\Delta S^\circ$ .



**Figura 14.6:** Representación gráfica de las cuatro matrices que incorpora el algoritmo de Zuker.

De esta forma podemos definir la función de partición  $Z$  de una molécula de RNA (en función de la temperatura):

$$Z = \sum_{P \in \Omega} \exp(-E(P)/RT) \quad (14.4)$$

donde:

- $P$  es una estructura determinada,
- $\Omega$  es el espacio de estructuras,
- $E(P)$  es la energía de la estructura,
- $R$  es la constante de gases,
- $T$  es la temperatura absoluta en grados Kelvin,
- $\exp(-E(P)/RT)$  se conoce como el factor de Boltzmann.

Así pues, dada una secuencia de RNA  $a_1, \dots, a_n$ , para todo  $1 \leq i \leq j \leq n$ , la función de partición  $Z_{i,j}$  se define como  $\sum_S e^{-E(S)/RT}$ , donde la suma se calcula sobre todas las posibles estructuras secundarias  $S$  de  $a[i, j]$ ,  $E(S)$  es la energía libre mínima de la estructura  $S$ .

Ahora estamos preparados para presentar las recursiones correspondientes al algoritmo de Zuker<sup>3</sup> para calcular la función de partición de una molécula de RNA. Dicho algoritmo se basa en el uso de cuatro matrices, cuyas definiciones son (ver también la Figura 14.6):

- $Z_{i,j}$ : función de partición sobre todas las estructuras secundarias en  $a[i, j]$ .
- $ZB_{i,j}$ : función de partición sobre todas las estructuras secundarias en  $a[i, j]$ , que contienen el par de bases  $(i, j)$ .
- $ZM_{i,j}$ : función de partición sobre todas las estructuras secundarias en  $a[i, j]$ , sujetas a la restricción de que  $a[i, j]$  es parte de un multiloop y tiene *como mínimo* un componente.

<sup>3</sup>En realidad, las recursiones para la función de partición reciben el nombre de recursiones de McCaskill [46], si bien el algoritmo para predecir la estructura de mínima energía libre es conocido como el algoritmo de Zuker [80].

- $ZM1_{i,j}$ : función de partición sobre todas las estructuras secundarias en  $a[i,j]$ , sujetas a la restricción de que  $a[i,j]$  es parte de un multiloop y tiene *exactamente* un componente. Además, se requiere que  $i$  forme un par de bases en el intervalo  $[i,j]$ ; es decir,  $(i,r)$  es un par de bases, para algún  $i < r \leq j$ .

y cuyas recursiones son las que siguen:

$$\begin{aligned}
Z_{i,j} &= \begin{cases} 1 & \text{si } j - i \leq \theta \\ Z_{i,j-1} + ZB_{i,j} + \sum_{k=i+1}^{j-\theta-1} Z_{i,k-1} \cdot ZB_{k,j} & \text{en caso contrario} \end{cases} \\
ZM1_{i,j} &= \begin{cases} 0 & \text{si } j - i \leq \theta \\ \sum_{k=i+\theta+1}^j \exp\left(-\frac{c \cdot (j-k)}{RT}\right) \cdot ZB_{i,k} & \text{en caso contrario} \end{cases} \\
ZM_{i,j} &= \begin{cases} 0 & \text{si } i \leq j \text{ y } j - i \leq \theta \\ \sum_{k=i}^{j-\theta-1} \exp\left(-\frac{b + c \cdot (k-i)}{RT}\right) ZM1_{k,j} + \\ \sum_{k=i}^{j-\theta-2} \exp\left(-\frac{b}{RT}\right) \cdot ZM_{i,k} \cdot ZM1_{k+1,j} & \text{en caso contrario} \end{cases} \\
ZB_{i,j} &= \begin{cases} 0 & \text{si } j - i \leq \theta \\ Z_{i,j}(S) + Z_{i,j}(H) + Z_{i,j}(B) + Z_{i,j}(I) + Z_{i,j}(M) & \text{en caso contrario} \end{cases}
\end{aligned}$$

donde:

$$\begin{aligned}
Z_{i,j}(S) &= \exp\left(-\frac{E(i, i+1, ; j-1, j, T)}{RT}\right) \cdot ZB_{i+1,j-1} \\
Z_{i,j}(H) &= \exp\left(-\frac{H(j-i-1, T)}{RT}\right) \\
Z_{i,j}(LB) &= \sum_{k=i+3}^{j-\theta-2} \exp\left(-\frac{B(k-i-1, T)}{RT}\right) \cdot ZB_{k,j-1} \\
Z_{i,j}(RB) &= \sum_{k=i+\theta+2}^{j-3} \exp\left(-\frac{B(j-k-1, T)}{RT}\right) \cdot ZB_{i+1,k} \\
Z_{i,j}(I) &= \sum_{\ell-i-2 \leq 2}^{j-\theta-3} \sum_{j-r-1+\ell-i-2 \leq 30}^{j-2} \exp\left(-\frac{I((\ell-i-1)+(j-r-1))}{RT}\right) \cdot ZB_{\ell,r} \\
Z_{i,j}(M) &= \exp\left(-\frac{a+2b}{RT}\right) \cdot \sum_{k=i+\theta+3}^{j-\theta-2} ZM_{i+1,k-1} \cdot ZM1_{k,j-1}
\end{aligned}$$

donde:

- $E(i, i+1, ; j, j-1, T)$  es la energía correspondiente al apilamiento de los pares de bases  $(i, j)$  y  $(i+1, j-1)$  a temperatura T,
- $H(l, T)$  es la energía de un hairpin de longitud  $l$  a temperatura T,
- $B(l, T)$  es la energía de un bulge de longitud  $l$  a temperatura T,

- $I(l, T)$  es la energía de un bucle interno de longitud  $l$  a temperatura  $T$ ,
- $a$  es una penalización energética por empezar un bucle múltiple,
- $b$  es una penalización energética por el numero de componentes de un bucle múltiple.

Teniendo en cuenta estas recursiones para el cálculo de la función de partición de una molécula de RNA, podemos inferir fácilmente las recursiones para calcular la estructura MFE: basta con cambiar los máximos por sumas y las sumas por multiplicaciones, y reemplazar los factores de Boltzmann por energías.

También es posible calcular la probabilidad de que dos nucleótidos formen un par de bases siguiendo el *algoritmo de McCaskill* [46]. Este algoritmo computa las probabilidades de pares de bases siguiendo la fórmula:

$$p(i, j) = \frac{\sum_{S : (i, j) \in S} \exp(-E(S)/RT)}{\sum_S \exp(-E(S)/RT)}$$

Es decir, los probabilidades de pares de bases de la distribución de Boltzmann, donde el numerador comprende la suma de todas las estructuras secundarias en las que ocurre el par de bases  $(i, j)$ , y en el denominador todas las posibles estructuras secundarias de RNA dada su secuencia.

Este tipo de algoritmo es posiblemente el más extendido, conocido y utilizado. Actualmente existen 3 implementaciones distintas que se diferencian en detalles como las energías libres o el tratamiento de *dangles* (nucleótidos sin par al final de una hélice) y apilamiento coaxial: UNAFold de Zucker [44], RNAfold de Hofacker [31] (dentro del software Vienna Package) y RNAstructure de Mathews [56].

Finalmente, discutiremos otro enfoque importante dentro del área de la predicción de estructura secundaria de RNA, la llamada *estructura de máxima esperanza de precisión* (o *Maximum Expected Accurate* en inglés). El concepto de estructura de máxima esperanza de precisión fue introducido por Do [21] y su definición es la siguiente: dada un secuencia  $a_1, \dots, a_n$  de RNA, la estructura de máxima esperanza de precisión  $S$  es aquella que maximiza la siguiente formula:

$$\sum_{(i,j) \in S} 2\alpha \cdot p(i, j) + \sum_{i \text{ desapareado}} \beta q_i$$

donde

$$p(i, j) = \text{probabilidad del par de bases } (i, j)$$

y

$$q_i = 1 - \sum_{i < j} p(i, j) - \sum_{j < i} p(j, i).$$

La estructura de máxima esperanza de precisión puede ser calculada con un algoritmo de programación dinámica similar al de Nussinov-Jacobson con probabilidades obtenidas a través de una gramática estocástica de contexto libre como en [40], o con probabilidades obtenidas usando el algoritmo de McCaskill como en [42].

## Pseudonudos

La predicción de RNA pseudonudos representa un paso entre la predicción de estructuras secundarias con una complejidad relativamente baja y la predicción de estructuras terciarias con una complejidad alta. A nivel práctico, también interacciones del tipo *kissing loop* pueden ser considerados como pseudonudos. Se puede demostrar que el problema de encontrar pseudonudos arbitrarios es NP-duro. Por lo tanto hay que reducir la complejidad del problema.

Podemos distinguir entre algoritmos heurísticos y recursivos. Además, tanto programas de predicción de estructuras secundarias como terciarias pueden incluir pseudonudos. Recomendamos estudiar las programadas de predicción antes de utilizarlos, se puede consultar una lista actualizada de programas en Wikipedia<sup>4</sup>

Empezamos con las *recursiones*. Un método muy eficaz se basa en el *algoritmo de Reeder y Giegerich* que escala como  $O(n^4)$  en tiempo y  $O(n^2)$  en espacio (Programa PKnotsRG [51]). Utiliza una recursión del tipo Zuker y se limita a los pseudonudos más simples, con un sólo posible entrelazamiento entre dos *stacks* y sin *bulges*. A pesar de estas restricciones, ese algoritmo incluye al tipo de pseudonudos más frecuente y estable y es un método rápido.

Otro algoritmo recursivo es el de Rivas y Eddy [57]. Incluye la posibilidad de *kissing hairpin* y *three-knot* y escala como  $O(n^6)$  en tiempo y  $O(n^4)$  en espacio. Mencionamos los algoritmos de Akutsu [3] y de Chen, Condon y Jabbari [13] como métodos con complejidad y prestaciones intermedias.

Presentamos dos métodos de predicción de pseudonudos basado en heurísticas. Estos algoritmos son de interés porque son rápidos y pueden alcanzar una buena precisión [11]. Uno se llama *ILM* (*Iterated Loop Matching*) [58] y utiliza una iteración tipo Nussinov como plegamiento herártico. Otro método es *HotKnots* [55], basado en elegir elementos de la estructura secundaria con energía libre relativamente baja, para sucesivamente añadir elementos hacia la estructura final.

Otro algoritmo heurístico es *Probknot* [8]. En este caso, en vez de usar la energía libre se basa en la estructura de máxima esperanza de precisión. Usando las probabilidades de dos nucleótidos de formar un par de base, la estructura de máxima esperanza de precisión puede ser extendida a un problema de apareamiento máximo, donde los nodos del grafo son nucleótidos y las aristas tienen un peso igual a la probabilidad. En este contexto, resolver el problema de apareamiento máximo resulta en un conjunto de pares de bases que puede contener pseudonudos. Aunque el problema de apareamiento máximo se puede resolver en tiempo polinómico, *Probknot* usa un algoritmo más rápido y más eficiente.

## Estructuras terciarias

En la literatura, es posible encontrar métodos que hablan de estructura terciaria de RNA, pero en realidad sólo tienen en cuenta estructuras secundarias con pseudonudos o con tripletes de bases y pares de bases no-canónicos. Este es el caso de [35] donde se presenta un algoritmo capaz de encontrar la estructura secundaria de mínima energía libre con tripletes y pares de base no-canónicos. Para ello, utiliza unas complejas recursiones que derivan del concepto de *2-diagrams* sin cruces.

Sin embargo, existe algún método para calcular la estructura terciaria de una molécula de RNA. El más conocido pertenece a las llamadas MC-tools de Major [49]. MC-fold primero encuentra estructuras secundarias que maximicen la frecuencia de cuartetos de nucleótidos que aparecen en las bases de datos de estructuras reales de RNA. Estos cuartetos pueden ser o no ser pares de bases apilados. Esta estructura secundaria es transformada en terciaria gracias a MC-Sym. MC-Sym tiene en cuenta todos los átomos y se basa en especificar los llamados *dihedral angles* que definen las posiciones del esqueleto azúcar-fosfato y de las bases. Estos ángulos están restringidos a sus valores más frecuentes, calculados a partir de estructuras reales.

Muy recientemente encontramos un nuevo método para el cálculo de la estructura terciaria de RNA en [54]. Este método también utiliza MC-Sym, aunque sustituye MC-Fold por RNA-MoIP. RNA-MoIP

---

<sup>4</sup>List of RNA structure prediction software. [http://en.wikipedia.org/wiki/List\\_of\\_RNA\\_structure\\_prediction\\_software](http://en.wikipedia.org/wiki/List_of_RNA_structure_prediction_software)

refina estructuras secundarias utilizando programación entera para acomodar estructuras terciarias, en muchas ocasiones, reemplazando pares de bases canónicos por no-canónicos.

Finalmente, cabe reseñar el reciente esfuerzo por impulsar el diseño de algoritmos de predicción de estructura terciaria de RNA fomentado por la competición conocida como RNA-Puzzles<sup>5</sup>.

#### 14.4.3. Otros problemas relacionados con el plegamiento de RNA

Hay muchos problemas y herramientas relacionadas con la estructura del RNA. Dejando de lado el área de las herramientas para la búsqueda de RNA no codificante, describiremos a continuación los ejemplos más relevantes.

#### Hibridización de RNA

Existen varios algoritmos para determinar la estructura secundaria de mínima energía de hibridización de RNA (o de RNA con DNA). Este problema tiene aplicaciones en el estudio de *micro-arrays* y en el estudio de ciertos procesos biológicos como el sistema CRISPR en bacterias o el proceso de splicing de RNA con exclusividad mutua de casetes como en el caso del gen Dscam en *Drosophila* [30].

Para calcular la mínima energía libre de hibridización se usa un algoritmo similar al de Zuker, en el cual las dos cadenas se juntan y se tiene un tratamiento especial de las estructuras en torno a la posición de juntura. Hay varias herramientas que resuelven este problema, como RNACofold del Vienna Package o RNAhybrid dentro de UNAFold. También existen otras herramientas como RNA duplex o RNAup (ambas dentro del Vienna Package). RNA duplex calcula todas las posibles estructuras locales de hibridización entre dos cadenas de RNA sin considerar estructuras intra-moleculares, y RNAup calcula la mejor estructura local de hibridización teniendo en cuenta que la cadena de mayor longitud forma una estructura secundaria que hay que romper antes de formarse la hibridización.

#### RNAbor

RNAbor es una herramienta desarrollado por el Clote Lab [26] en la que se pueden calcular la probabilidad de encontrar estructuras que difieren en  $k$  pares de bases con respecto a una estructura inicial. Dicha estructura inicial puede ser tanto la estructura de mínima energía libre como la estructura vacía (ningún par de bases) o cualquier otra estructura aleatoria. Esto se consigue usando un algoritmo de programación dinámica similar al de McCaskill en el cual se va concentrando la probabilidad de cada valor  $k$  simultáneamente.

Tanto RNAbor como su reciente versión FFTbor [63] en la que se mejora la velocidad usando la transformada de Fourier pueden utilizarse para detectar riboswitches, para estudiar el espacio de plegamiento o incluso para aproximar la velocidad de plegamiento de RNA.

#### RNAmutants

RNAmutants es otra herramienta desarrollada por el Clote Lab [75]. De una forma similar al caso anterior, RNAmutants calcula la probabilidad (y la estructura de mínima energía libre) para todas las

---

<sup>5</sup>RNA-Puzzles. <http://paradise-ibmc.u-strasbg.fr/rnапuzzles>

secuencias que difieren en  $k$  nucleótidos ( $k$  nucleótidos que cambian de valor, no que se añaden o se eliminan) de la secuencia original.

RNAmutants puede ser utilizado para estudiar el espacio mutacional de una secuencia de RNA, identificando posiciones más débiles o más robustas ante posibles mutaciones. RNAmutants también sirve para comparar la robustez de diferentes familias de RNA.

## Mapeo secuencia a estructura secundaria

Si queremos conocer para una secuencia dada el conjunto de estructuras compatibles, también queremos saber cual el conjunto total de estructuras para todas las secuencias. Existe un nivel de degeneración entre la secuencia y la estructura secundaria. Mientras hay  $4^n$  secuencias de longitud  $n$ , hay un número más bajo de estructuras secundarias,  $S(n)$ . Grüner et al. derivaron aproximaciones para ese número,  $S(n) \approx 0,7131 \times n^{-3/2} (2,2888)^n$ , si se permite la formación de pares sueltos, y  $S(n) \approx 1,4848 \times n^{-3/2} (1,849)^n$ , si se prohíbe la formación de pares sueltos [32].

Destacamos varias propiedades importantes del mapeo entre secuencia y estructura secundaria: (a) Degeneración: como hay más secuencias que estructuras, sobre todo para  $n$  grande, hay secuencias distintas que pliegan en la misma estructura. (b) Distribución de estructuras: existen pocas estructuras que son el resultado del plegamiento de muchas secuencias (estructuras frecuentes) mientras hay muchas estructuras que son estructuras raras. Es una distribución ancha con una cola larga. (c) Estructuras frecuentes están distribuidos de forma homogénea en el espacio de las secuencias: de una secuencia aleatoria en pocas mutaciones se puede obtener una secuencia que pliega en una estructura frecuente. (d) Redes neutrales: Las secuencias que pliegan en estructuras frecuentes forman una red extensa en el espacio de secuencias de tal manera que si la longitud de la molécula es suficientemente larga, esa red neutral percola el espacio de secuencias.

En la misma línea del cálculo del número de estructuras secundarias, existe todo un campo a lo que se ha venido a llamar Combinatoria de RNA. Dentro de este campo destacamos: cálculo de el número de estructuras secundarias canónicas y saturadas [16], cálculo de la distancia entre el 5' y el 3' de una molécula de RNA [17], etc.

## Comparación de estructuras

Mientras la diferencia entre secuencias se puede cuantificar con la *distancia Hamming* (número de nucleótidos distintos), para comparar estructuras existen una variedad de métodos distintos. Aunque se puede definir una distancia Hamming entre estructuras basado en el alfabeto punto-paréntesis, no se considera una medida muy apta para estructuras, y se prefiere la *distancia base-pair* que es el número de pares que uno tiene que abrir y cerrar para convertir una estructura en otra. Además, existe la posibilidad de definir distancias basado en la formulación de árbol para una estructura, y se define una distancia a través de cambios que uno tiene que hacer para convertir una estructura en otra. Un ejemplo es la *distancia tree-edit* que junto con las otras medidas están presentadas en [60].

Un concepto similar es el de agrupar estructuras según su similitud, reduciendo la descripción de la estructura secundaria. Una posible definición es el *Shape* que es una representación de estructura secundaria en la que las hélices se colapsan, por ejemplo, (((...((...))..(((...))))..)) pasa a ser [ [ ] [ ] ] [41]. Otra posibilidad es la definición de familias de estructuras como en [66, 69].

#### 14.4.4. Biología Sintética de RNA: plegamiento inverso

La *biología sintética* es una disciplina emergente con ramificaciones que van desde la detección de moléculas dentro de la célula a la creación de genomas sintéticos y nuevas formas de vida. Grupos pioneros en este campo han obtenido resultados espectaculares como, por ejemplo, la síntesis combinatoria de redes de genes, la síntesis de genomas usando *BioBricks*, y la reacción de hibridización en cadena, en la cual monómeros estables de DNA se juntan sólo ante la exposición a un fragmento de DNA objetivo. La mayor parte de los trabajos en el campo de la biología sintética se concentran en lo que se puede considerar como genómica sintética, como por ejemplo, la regulación sintética de genes [14] y el desarrollo de bloques genéticos, gracias a los cuales, se pueden construir nuevos genomas [64].

Sin embargo, esta sección versa sobre la biología sintética de RNA. Como hemos visto en los apartados anteriores, y al igual que en el caso de las proteínas, la estructura de RNA determina en gran medida su función. Dada la gran cantidad de nuevos tipos de RNA con funciones complejas, y dada la actual variedad de algoritmos para predicción de estructura de RNA, de alineamiento estructural, de búsqueda de motivos estructurales, etc; parece claro que algunos de los más importantes avances en la biología sintética tratarán el diseño y validación experimental de nuevas estructuras sintéticas de RNA.

#### Plegamiento inverso de RNA

Dado que el cálculo de la estructura secundaria de mínima energía libre puede ser realizado de forma eficiente y que la determinación de la estructura de RNA con pseudonudos y la estructura terciaria son NP-completos [43], nos centraremos en el problema del plegamiento inverso en cuanto se refiere a la estructura secundaria.

Como la estructura secundaria de RNA es una parte esencial de la estructura terciaria y el plegamiento tiene lugar de forma jerárquica (ver Cap. 14.4), cualquier solución al problema del plegamiento inverso para estructura secundaria es, claramente, un gran paso hacia el diseño de RNA funcional.

Dada una estructura secundaria  $S$  sobre una cadena desconocida de longitud  $n$ , el *problema del plegamiento inverso* consiste en encontrar la secuencia de RNA  $a_1, \dots, a_n$  (es decir, la palabra de longitud  $n$  del alfabeto  $\sigma = A, C, G, U$ ) cuya estructura mínima de energía libre es  $S$ .

Aunque no ha sido demostrado formalmente, se cree que el problema del plegamiento inverso de RNA es NP-duro.

Existen varios algoritmos para resolver el problema del plegamiento inverso de RNA: `RNAinverse` [34], `RNA-SSD` [5], `INFO-RNA` [9], `MODENA` [71], `NUPACK-DESIGN` [79], `Inv` [28]. Todos estos algoritmos pueden ser clasificados como heurísticos: comienzan con una secuencia inicial que es mejorada de forma iterativa hasta encontrar una solución o terminar por algún otro criterio como tiempo límite o máximo número de iteraciones.

El primer algoritmo que se encuentra en la literatura es `RNAinverse`, el cual forma parte del Vienna Package [31, 34]. `RNAinverse` divide la estructura objetivo  $S$  en subunidades más pequeñas e intenta encontrar la secuencia de RNA usando un camino adaptativo o algoritmo avaro. Las posiciones de la secuencia inicial son mutadas, estas mutaciones se aceptan si la función objetivo mejora. En este caso, la función objetivo es la distancia de Hamming entre la estructura de mínima energía libre de la secuencia y la estructura objetivo  $S$ . `RNAinverse` puede devolver la solución correcta, una solución aproximada o ninguna solución, dependiendo de la dificultad de la instancia.

`RNA-SSD` [5] es un algoritmo distinto y muy eficiente, si bien comparte la misma filosofía de divide y vencerás, dividiendo la estructura de forma jerárquica en subunidades. En comparación con

**RNAinverse**, RNA–SSD utiliza una inicialización más sofisticada e implementa un algoritmo de búsqueda local estocástica, en vez de un simple camino adaptativo. RNA–SSD es capaz de encontrar la secuencia correcta para estructuras de más de mil nucleótidos de longitud.

El tercer algoritmo es **INFO–RNA** [9]. La principal diferencia con los anteriores algoritmos reside en su inicialización, la cual se basa en un algoritmo de programación dinámica para elegir la secuencia  $s_1, \dots, s_n$  que es compatible con  $S$  y tiene la menor energía libre. Aunque la energía libre  $E(s_1, \dots, s_n; S)$  de la estructura objetivo  $S$  en  $s_1, \dots, s_n$  es menor o igual a la energía libre  $E(s'_1, \dots, s'_n; S)$  para todas las secuencias  $s'_1, \dots, s'_n$  que son compatibles con  $S$ , esto no significa que la estructura de mínima energía libre de  $s_1, \dots, s_n$  es la estructura objetivo  $S$ . INFO–RNA es, por lo menos, igual de eficiente que RNA–SSD, y debido a su especial inicialización, devuelve secuencias de RNA cuya energía libre es muy baja (con alto contenido de GCs). Aunque esto pueda parecer beneficioso, estas secuencias de alto contenido de GCs suele tener poco parecido con secuencias reales de RNA.

El cuarto, **MODENA** [71], es bastante diferente a todos los anterior. MODENA usa el conocido algoritmo genético NSGA2 [19] para encontrar el conjunto débil de Pareto de soluciones óptimas respecto a dos funciones objetivo: estabilidad de la estructura (mínima energía libre) y similitud (distancia entre la estructura de mínima energía libre y la estructura objetivo  $S$ ).

**NUPACK–DESIGN** [79], es el punto de partida de un pionero proyecto del laboratorio de Pierce, para diseñar moléculas de RNA que más tarde son validadas tanto *in vitro* como *in vivo*. NUPACK–DESIGN utiliza un enfoque parecido a RNA–SSD, aunque en este caso, NUPACK–DESIGN intenta encontrar secuencias con mínimo *defecto de colectividad* [20].

Dada una secuencia de RNA  $s = s_1, \dots, s_n$  con respecto a una estructura objetivo  $S$ , el *defecto de colectividad* se define como el valor esperado de nucleótidos cuya estado de emparejamiento difiere del que tienen en la estructura objetivo  $S$ . Formalmente tenemos:

$$n(s, S_0) = n - \sum_{1 \leq i, j \leq n} p_{i,j}^* \cdot I[(i, j) \in S] - \sum_{1 \leq i \leq n} p_{i,n+1}^* \cdot I[i \text{ desapareado en } S]$$

donde  $I$  es la función indicatriz y, para cada posición fija  $1 \leq i \leq n$ , se define la función de probabilidad  $p_{i,j}^*$ , para todo  $j$  en  $[1, n + 1]$ , simetrizando  $p$  para valores  $1 \leq i, j \leq n$ , y así pues definiendo  $p_{i,n+1}^* = 1 - \sum_{j > i} p_{i,j} - \sum_{j < i} p_{j,i}$ .

Finalmente, el algoritmo **Inv** [28] utiliza una rutina de búsqueda local estocástica para encontrar la secuencia cuya mínima energía libre con psuedonudos es la *3-noncrossing* estructura objetivo. Una estructura *3-noncrossing* es una estructura (posiblemente con psuedonudos) en la cual no es posible encontrar 3 (o más) pares de bases que se cruzan mutuamente. **Inv** es un algoritmo de programación dinámica de tiempo exponencial [37] y utiliza el hecho de que cada *3-noncrossing* estructura tiene una única descomposición en hélices.

En contraposición a todos los algoritmos anteriores, un nuevo enfoque utiliza programación con restricciones y es completo. **RNAifold** [29] esta implementado en **COMET** [72] y tiene la capacidad de encontrar todas las secuencias cuya estructura de mínima energía libre es la estructura objetivo.

## 14.5. Evolución de RNA

### 14.5.1. El RNA como modelo evolutivo

La evolución molecular cubre una enorme área de investigación. Esta comprende desde la química prebiótica y las preguntas sobre el origen de la vida, hasta el diseño artificial de moléculas y la selección y

evolución in vitro con sus aplicaciones en nano y biotecnología, pasando por muchos aspectos relacionados con el origen y las relaciones entre las especies, el estudio de la evolución viral y bacteriana y sus implicaciones médicas. En este texto, no pretendemos dar una visión completa de todo el campo, sino centrarnos en los enfoques teóricos del uso de las poblaciones de moléculas de RNA como un modelo para entender la evolución de replicadores simples. Y para ello, comenzamos identificando las dos propiedades que todo sistema evolutivo debe tener: (i) un *mecanismo que introduzca variabilidad genética* en la población (en nuestro caso es la mutación puntual de nucleótidos) y (ii) una *diferenciación de los genomas con respecto a su capacidad replicativa (fitness)* que permita la selección de los más aptos (mediante una presión selectiva).

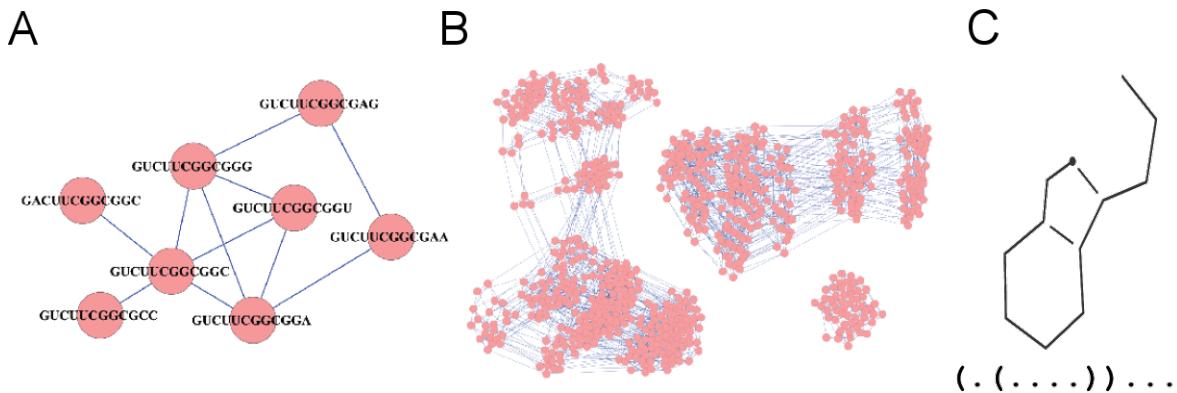
Las poblaciones de *moléculas de RNA* son un modelo muy adecuado para estudiar procesos evolutivos porque *incorporan, en una única entidad molecular, tanto genotipo como fenotipo* [25]. Mientras que los errores en el proceso de replicación introducen mutaciones en la secuencia de RNA (genotipo), la selección actúa sobre la función (fenotipo) de la molécula. Como en muchos casos la estructura espacial de la molécula es crucial para su función bioquímica, la estructura de una molécula de RNA puede considerarse como una representación del fenotipo.

No obstante, es muy difícil obtener la estructura real (es decir, tridimensional y con todos sus detalles) en la que pliega una secuencia arbitraria (ver Sección 7.7), y en la práctica conocemos con alta precisión sólo las estructuras de algunas moléculas fundamentales en la biología actual (como el tRNA, de 76 nucleótidos). Sin embargo, la estructura secundaria de una secuencia sí que se puede conocer con más precisión y también predecir con fidelidad mediante algoritmos informáticos. Por esta razón se suele utilizar *la estructura secundaria como representación mínima del fenotipo*, y por extensión, de la función bioquímica de la molécula. Como consecuencia de todo esto, conviene recordar que existen varios niveles de degeneración en el sistema: muchas secuencias pueden plegar en la misma estructura, y diferentes estructuras pueden tener la misma función bioquímica.

#### 14.5.2. Redes neutrales de RNA: concepto y propiedades básicas

La idea de la *evolución neutral* fue introducida por Kimura [39] con el fin de explicar un resultado por entonces muy sorprendente: *un gran número de mutaciones observadas en proteínas, DNA, o RNA, no tienen efecto sobre el fitness de dichas moléculas*. La neutralidad es particularmente importante en la evolución de cuasiespecies [22], poblaciones de replicadores de alta tasa de mutación que están formadas por un gran número de fenotipos diferentes -y muchísimos más genotipos- donde la alta diversidad y la exploración constante del espacio de genomas se convierte en una estrategia adaptativa. Algunos ejemplos relevantes de cuasiespecies son los virus de RNA (VIH, Ebola, Gripe, etc...) y los replicadores de alta tasa de mutación en el contexto del mundo de RNA prebiótico.

Como ya se ha comentado, el RNA es un modelo ideal para el estudio de la evolución molecular, y de hecho las secuencias de RNA plegando en sus estructuras secundarias de energía libre mínima son probablemente el modelo más utilizado en la literatura de la relación genotipo-fenotipo [61]. *Cada estructura secundaria de RNA, utilizada como aproximación del fenotipo o función biológica de la molécula, tiene asociada una red neutral de genotipos*, donde los nodos representan todas las secuencias que pliegan en dicha estructura. Dos nodos de la red están conectados por un enlace cuando sus secuencias están a *distancia de Hamming 1*, es decir, cuando difieren en un solo nucleótido [52] (véase la Figura 14.7). Para una longitud dada  $l$  de la secuencia de RNA, tenemos por lo tanto tantas redes neutrales como estructuras secundarias distintas (o fenotipos distintos). Conviene resaltar el hecho de que cada red neutral puede contener uno o más componentes aislados entre sí, es decir, que no siempre mediante mutaciones puntuales una secuencia puede recorrer la totalidad de secuencias que pliegan en la misma estructura secundaria. En este último caso, la red neutral se compone de *subredes*.



**Figura 14.7:** Construcción de una red neutral asociada a una estructura secundaria de RNA dada. (A) Esquema de la construcción: Los nodos de la red son todas las secuencias que pliegan en la misma estructura secundaria, y se conectan entre sí si están a una distancia de Hamming uno, es decir, si difieren sólo en un nucleótido. (B) Red neutral asociada a la estructura secundaria de longitud 12 (. ....) ..., mostrada en (C). Nótese que, aunque todas las secuencias de una red neutral pliegan en la misma estructura secundaria, la red neutral puede estar dividida. En este caso, existen 3 subredes aisladas de tamaños 404, 341 y 55. Figura modificada de [2].

Puesto que el RNA consta de cadenas de 4 nucleótidos distintos (cuyas bases nitrogenadas son A, G, C y U), las redes neutrales asociadas a una longitud de secuencia  $l$  son *subredes inmersiones* en la macrorred formada por todas las secuencias posibles de longitud  $l$ . Esta macrorred es regular, tiene tamaño  $4^l$ , dimensión  $l$ , y cada nodo tiene grado  $3l$ , porque cada nucleótido puede sufrir tres mutaciones distintas, y cada secuencia tiene  $l$  nucleótidos. (Véase la Sección 19.2 para una introducción básica a la teoría de redes complejas.)

Los estudios analíticos del tamaño de las redes neutrales, es decir, de la cantidad de secuencias de longitud  $l$  compatible con una estructura secundaria dada (lo que se conoce como degeneración genotípofenotípico), han revelado que *las redes neutrales de genotipos son astronómicamente grandes incluso para valores moderados de la longitud de la secuencia*. Sólo por citar un par de ejemplos, diremos que existen aproximadamente  $10^{28}$  secuencias compatibles con la estructura de una molécula de tRNA (que tiene una longitud  $l = 76$ ), mientras que las estructuras funcionales más cortas conocidas actualmente, que tienen una longitud  $l$  de unos 12 nucléotidos, pueden obtenerse a partir de más de  $10^6$  secuencias diferentes [2]. De hecho, una aproximación válida para secuencias suficientemente largas es que existen  $S_l = 1,4848 \times l^{-3/2} (1,8488)^l$  estructuras secundarias para secuencias de longitud  $l$  [62], y por lo tanto el tamaño medio de una red neutral crece con  $l$  como  $4^l / S_l = 0,673 \times l^{3/2} 2,1636^l$ , cantidad que crece enormemente con  $l$ . Este valor medio no es sin embargo representativo de la distribución real de los tamaños de redes neutrales, que es una función muy ancha sin una media bien definida y con una cola también muy ancha [62, 66]. De hecho, el espacio de secuencias de RNA de longitud  $l$  está dominado por un número relativamente pequeño de estructuras comunes que son muy abundantes y que se encuentran habitualmente en la naturaleza como motivos estructurales de moléculas de RNA funcional [23, 27]. Las redes neutrales correspondientes a dichas estructuras comunes percolan el espacio de secuencias [33, 53] y facilitan así la exploración de un gran número de estructuras alternativas. Esto es posible porque *las diferentes redes neutrales están profundamente entrelazadas*: a partir de cualquier secuencia aleatoria, y mediante pocos pasos sobre la red (o mutaciones), es posible llegar a todas las estructuras comunes [53].

### 14.5.3. Modelización matemática de la evolución sobre redes neutrales de RNA

#### Ecuaciones dinámicas

Más adelante en este capítulo, en la Subsección 14.5.5, nos centraremos en el caso general de la evolución de poblaciones de RNA sobre un paisaje de *fitness* rugoso que tiene en cuenta de forma diferenciada todas las estructuras posibles de RNA. En él, una población puede acceder a través de mutaciones a todo el espacio de secuencias de tamaño  $4^l$  en su camino hacia una *estructura objetivo*. Sin embargo, comenzaremos suponiendo que, de todas las estructuras secundarias posibles en las que pueden plegar todas las secuencias de RNA de longitud  $l$ , sólo una es funcional (su función específica dependerá del contexto, y es irrelevante ahora). Por lo tanto, el *fitness* de las secuencias de una única red es máximo, mientras que el *fitness* de todas las secuencias pertenecientes a otras redes neutrales es 0. Con esta hipótesis, la evolución de una población de RNA a través del espacio de genotipos debido a mutaciones se limita a la red neutral funcional (o a cada subred de la misma si está partida en subredes), ya que suponemos que sólo la progenie que se mantiene dentro de la red neutral sobrevive. Veamos la forma de modelizar este proceso haciendo uso de la teoría de cadenas de Markov y matrices de transición [1].

Cada nodo  $i$  en la red, que representa a un genotipo distinto, tiene un número  $n_i(t)$  de secuencias en un tiempo dado  $t$ . Hay  $i = 1, \dots, m$  nodos en la red, cada uno con un grado, o número de vecinos,  $k_i$ . La población total es  $N = \sum_i n_i(t)$ , y suponemos  $N \rightarrow \infty$  para evitar efectos estocásticos debido al tamaño finito de la población. La distribución inicial de las secuencias en la red a  $t = 0$  es  $n_i(0)$ . Las secuencias de longitud  $l$ , formadas por los 4 nucleótidos diferentes de que consta el RNA, tienen a lo sumo  $3l$  vecinos. Llamamos  $\{v_i\}$  al conjunto de los  $k_i$  vecinos del nodo  $i$ . En cada paso de tiempo, las secuencias en cada nodo se replican. Las nuevas secuencias mutan a uno de los  $3l$  vecinos más cercanos con probabilidad  $\mu$ , y se mantienen iguales a la secuencia madre con una probabilidad  $1 - \mu$ . En nuestro caso,  $0 < \mu \leq 1$ . El caso singular  $\mu = 0$  se excluye para evitar una dinámica trivial y garantizar una evolución hacia un estado de equilibrio único. Por lo tanto, con probabilidad  $k_i/(3l)$  las secuencias mutadas existen en la red neutral y en ese caso se suman a la población de los nodos vecinos correspondientes. De lo contrario, caen fuera de la red y desaparecen.

Las *ecuaciones que describen la dinámica de la población sobre la red neutral* presentada en el párrafo anterior son:

$$n_i(t+1) = (2 - \mu)n_i(t) + \frac{\mu}{3l} \sum_{j \in \{v_i\}} n_j(t) \quad (14.5)$$

y de forma matricial

$$\vec{n}(t+1) = (2 - \mu)I\vec{n}(t) + \frac{\mu}{3l}A\vec{n}(t) \quad (14.6)$$

donde  $I$  es la matriz identidad y  $A$  es la matriz de adyacencia de la red, cuyos elementos son  $A_{ij} = 1$ , si los nodos  $i$  y  $j$  están conectados y  $A_{ij} = 0$  en caso contrario (véase la Sección 19.2).

La dinámica asociada al sistema cuya evolución la define una matriz de transición  $M$  viene dada por  $\vec{n}(t+1) = M\vec{n}(t)$ . En nuestro caso particular, la matriz de transición  $M$  se define como

$$M = (2 - \mu)I + \frac{\mu}{3l}A \quad (14.7)$$

## Estudio del estado asintótico de la población

Llamemos  $\{\lambda_i\}$  al conjunto de autovalores de  $M$ , con  $\lambda_i \geq \lambda_{i+1}$ , y  $\{\vec{u}_i\}$  los autovectores correspondientes (para información general sobre autovalores y autovectores, véase cualquier libro de álgebra lineal, por ejemplo [12]). Los autovectores verifican que  $\vec{u}_i \cdot \vec{u}_j = 0$ ,  $\forall i \neq j$  y  $|\vec{u}_i| = 1$ ,  $\forall i$ . Una matriz es irreducible cuando el grafo correspondiente es conexo, y en nuestro caso cualquier par de nodos  $i$  y  $j$  de la red están conectados a través de mutaciones por definición. Si además de irreducible, una matriz cumple la condición  $M_{ii} > 0$ ,  $\forall i$ , entonces se dice que la matriz  $M$  es primitiva. La matriz de transición  $M$  presentada en la Ecuación 14.7 es efectivamente primitiva, y el teorema de Perron-Frobenius asegura que, en el intervalo de valores de  $\mu$  utilizados, el autovalor mayor de  $M$  es positivo, es mayor que el resto de autovalores, y su autovector asociado también es positivo (es decir,  $(\vec{u}_1)_i > 0$ ,  $\forall i$ ). La verificación de este teorema es importante porque nos permite dar los pasos mostrados a continuación (véase [36], por ejemplo, para profundizar en el teorema de Perron-Frobenius).

La dinámica asociada al sistema, plasmada en la Ecuación 14.6, puede ser escrita de la forma:

$$\vec{n}(t) = M^t \vec{n}(0) = \sum_{i=1}^m \lambda_i^t \alpha_i \vec{u}_i \quad (14.8)$$

donde  $\alpha_i$  es la proyección de la condición inicial sobre el autovector  $i$  de  $M$ ,  $\alpha_i = \vec{n}(0) \cdot \vec{u}_i$ .

Además, como  $\lambda_1 > |\lambda_i|$  por el teorema de Perron-Frobenius,  $\forall i > 1$ , el estado asintótico de la población es proporcional al autovector que corresponde al mayor autovalor,  $\vec{u}_1$ :

$$\lim_{t \rightarrow \infty} \left( \frac{\vec{n}(t)}{\lambda_1^t \alpha_1} \right) = \vec{u}_1 \quad (14.9)$$

mientras que el mayor autovalor  $\lambda_1$  nos da la tasa de crecimiento de la población en el equilibrio (en ausencia de reescalamiento). Si se normaliza la población  $\vec{n}(t)$  tal que  $|\vec{n}(t)| = 1$  después de cada generación, se cumple que  $\vec{n}(t) \rightarrow \vec{u}_1$  cuando  $t \rightarrow \infty$ . Estos resultados son importantes, porque afirman que, independientemente de la condición inicial, la población de secuencias tiende asintóticamente hacia una distribución constante sobre la red que es igual al primer autovector de la matriz de transición.

## Tiempo al estado asintótico

Otra cantidad dinámica relevante y con directas implicaciones biológicas es el tiempo que la población tarda en alcanzar dicho equilibrio. Para obtenerlo, partimos de la Ecuación 14.8, que describe la dinámica transitoria hacia el equilibrio a partir de la condición inicial  $\vec{n}(0)$ . La distancia  $\Delta(t)$  al estado de equilibrio se puede escribir como:

$$\Delta(t) = \left| \frac{M^t \vec{n}(0)}{\lambda_1^t \alpha_1} - \vec{u}_1 \right| = \left| \sum_{i=2}^m \vec{\Delta}_i(t) \right| = \left| \sum_{i=2}^m \frac{\alpha_i}{\alpha_1} \left( \frac{\lambda_i}{\lambda_1} \right)^t \vec{u}_i \right| \quad (14.10)$$

A fin de estimar cuántas generaciones transcurren antes de que se alcance el equilibrio, fijamos una tolerancia  $\epsilon$ , y definimos el *tiempo al equilibrio*  $t_\epsilon$  como el número de generaciones necesarias para que se cumpla que  $\Delta(t_\epsilon) < \epsilon$ . En general,  $t_\epsilon$  se puede aproximar a primer orden por:

$$t_\epsilon^1 \simeq \frac{\ln |\alpha_2/\alpha_1| - \ln \epsilon}{\ln \lambda_1/\lambda_2} \quad (14.11)$$

y en definitiva:

$$t_\epsilon \propto \left( \ln \frac{\lambda_1}{\lambda_2} \right)^{-1} \quad (14.12)$$

Esta aproximación resulta ser extraordinariamente acertada en la mayoría de los casos gracias a la supresión exponencialmente rápida de las contribuciones debidas a los términos de orden superior (puesto que  $\lambda_i \geq \lambda_{i+1}, \forall i$ ). Se puede perder precisión, sin embargo, cuando  $\lambda_3 \approx \lambda_2$ , cuando la condición inicial de  $\vec{n}(0)$  es tal que  $\alpha_3 \gg \alpha_2$ , o cuando  $\epsilon$  es tan grande que la población está lejos del equilibrio y  $\Delta(t)$  se sigue rigiendo por  $\lambda_3$  y los autovalores de orden superior.

### Influencia de la topología de red en la dinámica

Llamemos  $\{\gamma_i\}$  al conjunto de autovalores de la matriz de adyacencia  $A$ ,  $\gamma_i \geq \gamma_{i+1}$ , y  $\{\vec{w}_i\}$  el conjunto de autovectores correspondientes. De la Ecuación 14.7, obtenemos:

$$M\vec{w}_i = (2 - \mu)I\vec{w}_i + \frac{\mu}{3l}A\vec{w}_i = \left[ (2 - \mu) + \frac{\mu}{3l}\gamma_i \right] \vec{w}_i \quad (14.13)$$

Por lo tanto, los autovectores de la matriz de adyacencia  $A$  son también autovectores de la matriz de transición  $M$ ,  $\vec{u}_i \equiv \vec{w}_i, \forall i$ , lo que demuestra que el estado asintótico de la población sólo depende de la topología de la red neutral [73]. Los autovalores de ambas matrices están relacionados a través de:

$$\lambda_i = (2 - \mu) + \frac{\mu}{3l}\gamma_i \quad (14.14)$$

donde el conjunto  $\{\gamma_i\}$  no depende de la tasa de mutación  $\mu$ . En resumen, la matriz de adyacencia contiene toda la información de los estados finales, mientras que la matriz de transición nos da información cuantitativa sobre la dinámica hacia el equilibrio.

Como se ve en la Ecuación 14.14, el valor mínimo de  $\lambda_1$  se obtiene en el límite de una población evolucionando a una tasa de mutación muy alta ( $\mu \rightarrow 1$ ) y para moléculas muy grandes ( $l \rightarrow \infty$ ). En este límite, todos los autovalores de  $M$  se vuelven independientes de la topología precisa de la red y  $\lambda_i \rightarrow 1, \forall i$ . En este caso extremo todas las secuencias hijas caen fuera de la red y mueren, pero la población se mantiene constante a través de la población original.

### Obtención de un resultado biológico relevante: la tendencia a la robustez mutacional

El *grado medio de la población*  $K(t)$  en el tiempo  $t$  se define como:

$$K(t) = \frac{\vec{k} \cdot \vec{n}(t)}{\sum_i n_i(t)} \quad (14.15)$$

donde  $\vec{k}$  es el vector grado y tiene como componentes el grado de los  $i = 1, \dots, m$  nodos de la red. En el límite de  $t \rightarrow \infty$ , se obtiene el grado medio en el equilibrio:

$$K(t \rightarrow \infty) = K = \frac{\vec{k} \cdot \vec{u}_1}{\sum_i (u_1)_i} \quad (14.16)$$

Se definen como  $k_{min}$ ,  $k_{max}$  y  $\langle k \rangle = \sum_i k_i/m$  el grado mínimo, máximo, y medio de la red, respectivamente. Un cálculo muy simple (basado en la identidad entre los autovectores de la matriz de adyacencia  $A$  y la de transición  $M$ ) nos dice que el grado medio de la población en el equilibrio,  $K$ , es igual al autovalor más grande  $\gamma_1$  de la matriz de adyacencia, también conocido como *radiopectral* de la red [73].

Además, el teorema de Perron-Frobenius para grafos conectados, con pesos no negativos y simétricos, establece límites en el grado medio  $\langle k \rangle$ : Cuando  $k_{min} < k_{max}$ , es decir, en la medida que el grafo no es regular, se cumple:

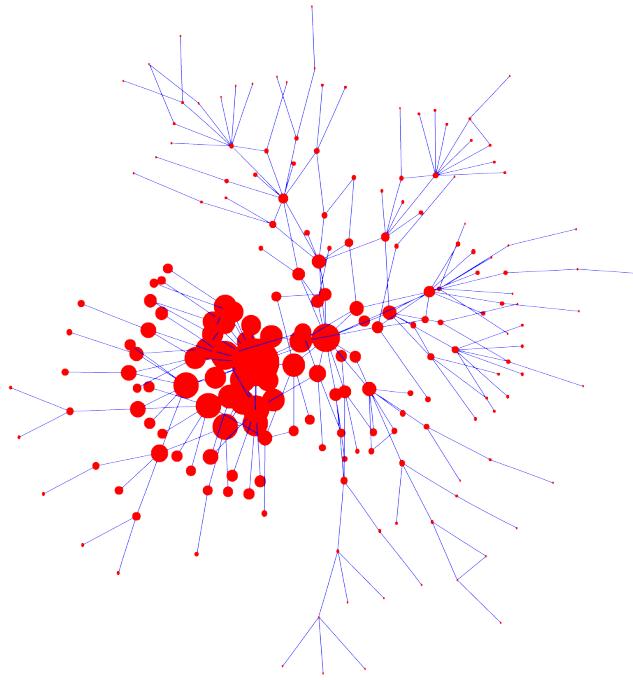
$$k_{min} < \langle k \rangle < \gamma_1 = K < k_{max}. \quad (14.17)$$

Por lo tanto, el grado medio  $K$  de la población en el equilibrio será mayor que el grado medio  $\langle k \rangle$  de la red, lo que indica que la población selecciona regiones con conectividad por encima del promedio de la red. Las implicaciones evolutivas de este resultado son importantes. El grado de cada nodo  $i$  refleja su robustez ante mutaciones, porque cuanto mayor sea dicho grado más probabilidades tendrá la secuencia de RNA de, en caso de mutar, mantener la estructura y en definitiva la función. Por lo tanto, la relación  $K > \langle k \rangle$  nos dice que una población de secuencias de RNA que evolucione sobre una red neutral tiende de forma natural a la robustez ante mutaciones, y en consecuencia a la fijación de la estructura (Véase la Figura 14.8).

#### 14.5.4. Limitaciones y líneas futuras de la modelización de la evolución sobre redes neutrales de RNA

Los enormes tamaños de las redes neutrales de RNA imposibilitan los estudios sistemáticos de este tipo de redes para RNAs de longitudes que no sean muy pequeñas, y suponen un reto importante a la bioinformática. Actualmente, sólo se han hecho estudios exhaustivos plegando la totalidad de las secuencias de RNA de longitud por debajo de unos 20 nucleótidos, donde el número de secuencias a plegar no supera los  $10^{12}$ . Por ejemplo, recientemente se estudiaron las propiedades topológicas de todas las redes neutrales asociadas a  $l = 12$  [2]. En la Tabla 14.2 se muestra una comparativa entre las propiedades topológicas de dichas redes neutrales y las de las redes de topología más habitual, las aleatorias Erdős-Renyi (ER) y las libres de escala Barabási-Albert (BA). Como queda claro en la tabla, la topología de las redes neutrales dista mucho de ser aleatoria, y de hecho el grado medio de la red  $\langle k \rangle$ , y por lo tanto la robustez ante mutaciones, crece con el tamaño de la red. Este hecho refuerza la hipótesis de que las estructuras que se ven en la naturaleza son aquellas cuyas redes neutrales son más grandes. Al argumento de que al ser más extensas, una secuencia que evoluciona aleatoriamente por el espacio de genomas encontrará dicha red más fácilmente, se une el hecho de que, una vez hallada, la mayor robustez a las mutaciones de la red mantendrá con más fidelidad a la población de secuencias dentro de sus márgenes.

Sin embargo, no hay seguridad de que los resultados obtenidos para moléculas de RNA de longitudes pequeñas sean extensibles a moléculas más largas, como puede ser el tRNA ( $l = 76$ ), y no digamos ya a los virus de RNA, cuyo material genético tiene del orden de miles de nucleótidos, o incluso más. El desafío computacional, teórico y experimental es fabuloso, y sin duda este problema requiere un enfoque distinto al utilizado hasta ahora. Además, no podemos olvidar que para desarrollar los cálculos mostrados en este apartado, hemos supuesto que existe una única red neutral funcional, mientras que el resto han sido obviadas. En la naturaleza, en muchas ocasiones varias estructuras secundarias distintas, aunque similares, pueden desarrollar la misma función, aunque su eficiencia sea distinta. Cada una de las redes neutrales asociadas a estas estructuras puede estar conectada en una red de redes, donde las



**Figura 14.8:** Tendencia a la robustez mutacional. Independientemente de la distribución inicial, una población de replicadores que evoluciona sobre una red compleja tiende en el equilibrio a las zonas más conectadas. En la figura hemos hecho evolucionar una población siguiendo la Ecuación 14.6 sobre una red neutral artificial. El tamaño de cada nodo es proporcional a la población alcanzada en el equilibrio, y muestra con claridad la tendencia natural a poblar la región con mayor grado medio de la red, y por lo tanto al refuerzo de las secuencias más robustas ante mutaciones. Figura modificada de [1].

Redes neutrales	Aleatorias (ER)	Libres de escala (BA)
$p(k)$	pico único	distr. de Poisson
$\langle k \rangle(N)$	$\sim \ln N$	constante
$C(k)$	$\sim k^{-1}$	constante ( $\frac{\langle k \rangle}{N}$ )
$C(N)$	$\sim (\ln N)^{-1}$	$\sim N^{-1}$
$\langle d \rangle(N)$	$\sim \ln N$	$\sim \ln N / \ln \langle k \rangle$
$k_{nn}(k)$	$\sim k^{0,75}$	constante ( $\langle k^2 \rangle / \langle k \rangle$ )
Asortatividad	asort. ( $r > 0$ )	no asort. ( $r \rightarrow 0$ )
		ley de potencias ( $\sim k^{-3}$ )
		constante
		constante ( $\sim N^{-0,75}$ )
		$\sim N^{-0,75}$
		$\sim \ln N / \ln \ln N$
		no trivial [4]
		no asort. ( $r \rightarrow 0$ )

**Tabla 14.2:** Comparación de la topología de las redes neutrales de longitud de secuencia  $l = 12$  con las redes aleatorias (Erdős-Renyi) y libres de escala (Barabási-Albert). Las cantidades reflejadas son la distribución de grado  $p(k)$ , el clustering en función del grado  $C(k)$  y en función del tamaño de la red  $C(N)$  respectivamente, el camino medio  $\langle d \rangle(N)$ , el grado medio de los vecinos  $k_{nn}(k)$ , y la asortatividad  $r$ . (Véase la Sección 19.2 para más información acerca de estas cantidades.)

secuencias van mutando y van saltando de unas redes a otras y variando en consecuencia su *fitness*. Si cada una de estas redes es representada por un único nodo, entonces obtenemos lo que se conoce como *redes de fenotipos*, campo de estudio abierto recientemente y que promete aportar importantes

resultados a esta línea de investigación [18].

#### 14.5.5. Evolución dirigida a una estructura objetivo y definición de distancia estructural

En los apartados anteriores hemos supuesto que una población de moléculas de RNA de longitud  $l$  se mueve estrictamente encima de una red neutral y que las moléculas que pliegan en otra estructura secundaria tienen *fitness* cero. Este *paisaje de fitness*, conocido como de *pico único (single-peak)*, es obviamente una aproximación muy simplificada. En esta sección consideramos una generalización representando un paisaje de *fitness* rugoso que tiene en cuenta de forma diferenciada todas las estructuras posibles, y donde las poblaciones evolucionan hacia una estructura objetivo.

Asumimos que una población puede acceder a través de mutaciones a todo el espacio de secuencias de tamaño  $4^l$  y que conocemos sus estructuras secundarias asociadas. Elegimos una estructura como estructura objetivo, es decir, con *fitness* óptimo. Ahora, las demás estructuras no tienen *fitness* zero, sino un *fitness* que depende de la similitud con la estructura objetivo. Para comparar estructuras entre sí, se han introducido diferentes medidas de distancia: *base-pair*, *Hamming*, y *tree-edit* (véase Sección 14.4.3). Se puede demostrar que esas medidas establecen una métrica y que podemos utilizarlas para definir una función de *fitness*. Una posibilidad es que la probabilidad de replicación de una molécula esté dada por  $p \propto \exp(-\beta d)$ , donde  $d$  es la distancia entre la estructura de la molécula y la estructura objetivo y  $\beta$  la magnitud de la presión selectiva.

Como el mapeo de secuencia a estructura es altamente complejo, hay que recurrir a métodos numéricos y estadísticos para estudiar el proceso evolutivo de una población. El esquema es el siguiente:

1. Elegir una estructura objetivo. Puede ser cualquier estructura, pero obviamente los casos más relevantes son aquellos en los que la elegida representa una estructura biológicamente relevante, p.ej. una horquilla (*hairpin*), una cabeza de martillo (*hammerhead*), tRNA, o similar. El espacio de secuencias, estructuras, la imprecisión del plegamiento y los recursos computacionales crecen fuertemente con la longitud de la molécula, por lo que muchos estudios están enfocados a moléculas cortas, de orden  $10^1$  a  $10^2$  nucleótidos.
2. Establecer el tamaño  $N$  de la población.
3. Formar una población inicial, que puede constar de secuencias al azar o elegidas expresamente.
4. Determinar las estructuras secundarias de las secuencias. Se pueden utilizar programas de libre acceso como el RNAfold del Vienna Package [34], véase Sección 14.4.2.
5. Determinar las distancias de las estructuras a la estructura objetivo.
6. Replicar la población con el método Wright-Fisher, teniendo en cuenta las probabilidades normalizadas de replicación:

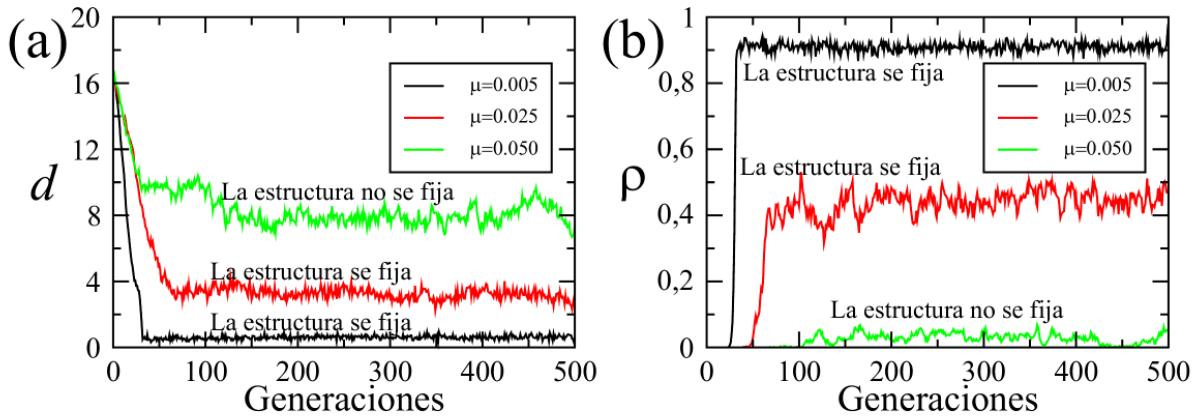
$$p(d_i) = \frac{\exp(-\beta d_i)}{\sum_{i=1}^N \exp(-\beta d_i)}. \quad (14.18)$$

Entre las posibilidades para elegir la magnitud  $\beta$ , destacamos dos particularmente relevantes: en función de la longitud de la molécula, p.ej.,  $\beta = 1/l$ , o en función de una propiedad variable, p.ej.,  $\beta = 1/d$ , siendo  $d = \sum_{i=1}^N d_i/N$  la distancia media de la población, y reflejando así una selección dependiente de la frecuencia.

7. Al crear una nueva generación de la población (que reemplaza a la anterior), hay que incluir una fuente de variabilidad, en el caso más simple mutaciones puntuales con una tasa de mutación por nucleótido  $\mu$ .

El proceso evolutivo avanza de generación en generación al repetir los pasos (4)-(7).

En el caso general, en la población inicial no hay secuencias que pliegan en la estructura objetivo, y por lo tanto la primera fase de la evolución está marcada por la búsqueda de dicha estructura objetivo. Cuando la población ya la ha encontrado por primera vez, empieza la segunda fase, en la que la estructura se fija y el número de moléculas con la estructura correcta crece en la población. Los tiempos de búsqueda dependen no sólo de la tasa de mutación, sino también de la estructura elegida [69]. Si la tasa de mutación es demasiado alta, es posible que la estructura objetivo vuelva a perderse. Una vez fijada la estructura, la población puede llegar a su estado asintótico, caracterizado por una distancia media mínima y una fracción máxima de moléculas con estructura objetivo. Como la mutación sigue operando, ese estado asintótico es estacionario sólo con respecto a cantidades medias. Además, si  $N$  es demasiado bajo, pueden darse efectos de tamaño finito.



**Figura 14.9:** Comportamiento evolutivo típico de una población de RNA para tres tasas de mutación distintas. Mostramos  $d$ , la distancia base-pair media a la estructura objetivo de la población, y  $\rho$ , la densidad de estructuras correctas en la población. El tamaño de la población es 602,  $\beta = 1$ , y la estructura objetivo es una horquilla (o *hairpin*) de longitud 35 nucleótidos.

En la Figura 14.9, mostramos un ejemplo numérico de lo que acabamos de exponer. En ella se observa el comportamiento de una población que evoluciona para tres valores distintos de la tasa de mutación. Las variables son  $d$ , la distancia base-pair media a la estructura objetivo de la población, y  $\rho$ , la densidad de estructuras correctas en la población. Si la tasa de mutación es pequeña, la población termina conteniendo un alto número de estructuras correctas, y la distancia media es baja. Si aumenta la tasa de mutación, se encuentran un número menor de estructuras correctas, y dicha distancia media aumenta. Si la tasa de mutación supera un cierto límite, la estructura objetivo no se mantiene de forma permanente en la población ( $\rho$  tiende a cero), y decimos que la estructura no se ha *fijado* en la población.

Desde que se implantó por primera vez un sistema de evolución de una población de RNA utilizando su estructura secundaria [24, 38], este modelo ha sido modificado muchas veces, aunque manteniendo su filosofía original. La función de selección puede ser exponencial, pero se han utilizado otras funciones

(lineales y no-lineales). Además, se han utilizado distintos tipos de distancia RNAEstructura: mientras la *base-pair* es la que más se asemeja a un proceso biofísico (el abrir y cerrar de enlaces), no puede ser utilizada si la población contiene moléculas de distintos tamaños, en cuyo caso se recurre a la distancia *tree-edit*. Muchas propiedades de las poblaciones en evolución sólo dependen cuantitativamente, no cualitativamente, de estas variables.

Obviamente, la función de selección puede ser ampliada para no sólo tener en cuenta la distancia a la estructura objetivo, sino también la distancia a una secuencia (p.ej. un trozo de una secuencia conservada) si la hubiera, u otro criterio de optimización, p.ej. una minimización de energía [67]. Finalmente, el marco del modelo también permite que el tamaño de la población no sea constante, para poder abarcar los llamados cuellos de botella [68].

En resumen, una población de moléculas de RNA en evolución permite estudiar procesos diversos: los tiempos de búsqueda que dependen de la tasa de mutación y el tipo de estructura, la adaptación de poblaciones en función de la fuerza de selección y mutación, o la robustez respecto a cambios de entorno (estructuras objetivo distintas, tamaños de población), entre otros muchos.



## 14.6. Bibliografía

- [1] J. Aguirre, J. Buldú, and S. Manrubia. Evolutionary dynamics on networks of selectively neutral genotypes: Effects of topology and sequence stability. *Phys. Rev. E*, 80:066112, 2009.
- [2] J. Aguirre, J. M. Buldú, M. Stich, and S. C. Manrubia. Topological structure of the space of phenotypes: The case of RNA neutral networks. *PLoS ONE*, 6:e26324, 2011.
- [3] T. Akutsu. Dynamic programming algorithms for rna secondary structure prediction with pseudoknots. *Discr. Appl. Math.*, 104:45–62, 2000.
- [4] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, 2002.
- [5] M. Andronescu, A. Fejes, F. Hutter, H. Hoos, and A. Condon. A new algorithm for rna secondary structure design. *J. Mol. Biol.*, 336:607–624, 2004.
- [6] M. Bailor, X. Sun, and H. Al-Hashimi. Topology links rna secondary structure with global conformation, dynamics, and adaptation. *Science*, 327:202–206, 2010.
- [7] A. Banerjee, J. Jaeger, and D. Turner. Thermal unfolding of a group i ribozyme: The low-temperature transition is primarily disruption of tertiary structure. *Biochemistry*, 32:153–163, 1993.
- [8] S. Bellaousov and D. Mathews. ProbKnot: fast prediction of rna secondary structure including pseudoknots. *RNA*, 16:1870–1880, 2010.
- [9] A. Busch and R. Backofen. Info-rna, a fast approach to inverse rna folding. *Bioinformatics*, 22:1823–1831, 2006.
- [10] S. Butcher and A. Pyle. The molecular interactions that stabilize rna tertiary structure: Rna motifs, patterns, and networks. *Acc. Chem. Res.*, 44:1302–1311, 2011.
- [11] S. Cao and S.-J. Chen. Predicting rna pseudoknot folding thermodynamics. *Nucl. Acids Res.*, 34:2634–2652, 2006.
- [12] M. Castellet and I. Llerena. *Algebra lineal y geometría*. Editorial Reverté, 2009.
- [13] H.-L. Chen, A. Condon, and H. Jabbari. An  $O(n(5))$  algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *J. Comput. Biol.*, 16:803–815, 2009.
- [14] Y. Chen, M. Jensen, and C. Smolke. Genetic control of mammalian T-cell proliferation with synthetic rna regulatory systems. *Proc. Natl. Acad. Sci. USA*, 107:8531–8536, 2010.
- [15] S. Cho, D. Pincus, and D. Thirumalai. Assembly mechanisms of rna pseudoknots are determined by the stabilities of constituent secondary structures. *Proc. Natl. Acad. Sci. USA*, 106:17349–17354, 2009.
- [16] P. Clote, E. Kranakis, D. Krizanc, and B. Salvy. Asymptotics of canonical and saturated rna secondary structures. *J. Bioinform. Comput. Biol.*, 7:869–893, 2009.
- [17] P. Clote, Y. Ponty, and J.-M. Steyaert. Expected distance between terminal nucleotides of rna secondary structures. *J. Math. Biol.*, 2011.
- [18] M. Cowperthwaite, E. Economou, W. Harcombe, E. Miller, and L. Meyers. The ascent of the abundant: how mutational networks constrain evolution. *PLoS Comp. Biol.*, 4:e1000110, 2008.
- [19] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, 2001.
- [20] R. Dirks, M. Lin, E. Winfree, and N. Pierce. Paradigms for computational nucleic acid design. *Nucl. Acids Res.*, 32:1392–1403, 2004.
- [21] C. Do, M. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.
- [22] M. Eigen. Selforganization of matter and evolution of biological macromolecules. *Naturwissenschaften*, 58:465–523, 1971.
- [23] W. Fontana, D. Konings, P. Stadler, and P. Schuster. Statistics of rna secondary structures. *Biopolymers*, 33:1389–1404, 1993.
- [24] W. Fontana and P. Schuster. A computer-model of evolutionary optimization. *Biophys. Chem.*, 26:123–147, 1987.
- [25] W. Fontana and P. Schuster. Shaping space: The possible and the attainable in rna genotype-phenotype mapping. *J. Theor. Biol.*, 194:491–515, 1998.

- [26] E. Freyhult, V. Moulton, and P. Clote. Boltzmann probability of rna structural neighbors and riboswitch detection. *Bioinformatics*, 23:2054–2062, 2007.
- [27] H. Gan, S. Pasquali, and T. Schlick. Exploring the repertoire of rna secondary motifs using graph theory with implications for rna design. *Nucl. Acids Res.*, 31:2926–2943, 2003.
- [28] J. Gao, L. Li, and C. Reidys. Inverse folding of rna pseudoknot structures. *Algorithms Mol. Biol.*, 5(27), 2010.
- [29] J. Garcia-Martin, P. Clote, and I. Dotu. RNAiFold: A constraint programming algorithm for rna inverse folding and molecular design. *J. Bioinform. Comput. Biol.*, 2013. in press.
- [30] B. Graveley. Mutually exclusive splicing of the insect dscam Pre-mRNA directed by competing intronic rna secondary structures. *Cell*, 123:65–73, 2005.
- [31] A. Gruber, R. Lorenz, S. Bernhart, R. Neubock, and I. Hofacker. The vienna rna websuite. *Nucl. Acids Res.*, 36:70–74, 2008.
- [32] W. Grüner, R. Giegerich, D. Strothmann, C. Reidys, J. Weber, I. Hofacker, P. Stadler, and P. Schuster. Analysis of rna sequence structure maps by exhaustive enumeration. i. neutral networks. *Monatsh. Chem.*, 127:355–374, 1996.
- [33] W. Grüner, R. Giegerich, D. Strothmann, C. Reidys, J. Weber, I. Hofacker, P. Stadler, and P. Schuster. Analysis of rna sequence structure maps by exhaustive enumeration. ii. structures of neutral networks and shape space covering. *Monatsh. Chem.*, 127:375–389, 1996.
- [34] I. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.
- [35] C. Honer zu Siederdissen, S. Bernhart, P. Stadler, and I. Hofacker. A folding algorithm for extended rna secondary structures. *Bioinformatics*, 27:129–136, 2011.
- [36] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 2nd edition, 2013.
- [37] F. Huang, W. Peng, and C. Reidys. Folding 3-noncrossing rna pseudoknot structures. *J. Comput. Biol.*, 16:1549–1575, 2009.
- [38] M. A. Huynen, D. A. M. Konings, and P. Hogeweg. Multiple coding and the evolutionary properties of RNA secondary structure. *J. Theor. Biol.*, 165:251–267, 1993.
- [39] M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217:624–626, 1968.
- [40] H. Kiryu, T. Kin, and K. Asai. Robust prediction of con-sensus secondary structures using averaged base pairing probability matrices. *Bioinformatics*, 23:434–441, 2007.
- [41] W. Lorenz, Y. Ponty, and P. Clote. Asymptotics of rna shapes. *J. Comput. Biol.*, 15:31–63, 2008.
- [42] Z. Lu, J. Gloor, and D. Mathews. Improved rna secondary structure prediction by maximizing expected pair accuracy. *RNA*, 15:1805–1813, 2009.
- [43] R. Lyngso and C. Pedersen. Rna pseudoknot prediction in energy-based models. *J. Comput. Biol.*, 7:409–427, 2000.
- [44] N. Markham and M. Zuker. UNAFold: software for nucleic acid folding and hybridization. *Methods Mol. Biol.*, 453:3–31, 2008.
- [45] D. Mathews, J. Sabina, M. Zuker, and H. Turner. Expanded sequence dependence of thermodynamic parameters provides robust prediction of rna secondary structure. *J. Mol. Biol.*, 288:911–940, 1999.
- [46] J. McCaskill. The equilibrium partition function and base pair binding probabilities for rna secondary structure. *Biopolymers*, 29:1105–1119, 1990.
- [47] R. Nussinov and A. Jacobson. Fast algorithm for predicting the secondary structure of single-stranded rna. *Proc. Natl. Acad. Sci. USA*, 77:6309–6313, 1980.
- [48] R. Nussinov, G. Pieczenik, J. Griggs, and D. Kleitman. Algorithms for loop matchings. *SIAM J. Appl. Math.*, 35:68–82, 1978.
- [49] M. Parisien and F. Major. The MC-Fold and MC-Sym pipeline infers rna structure from sequence data. *Nature*, 452:51–55, 2008.

- [50] J. Putz, B. Dupuis, M. Sissler, and C. Florentz. Mamit-trna, a database of mammalian mitochondrial trna primary and secondary structures. *RNA*, 13(8):1184–90, 2007.
- [51] J. Reeder and R. Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.
- [52] C. Reidys, C. Forst, and P. Schuster. Replication and mutation on neutral networks. *Bull. Math. Biol.*, 63:57, 2001.
- [53] C. Reidys, P. Stadler, and P. Schuster. Generic properties of combinatory maps - neutral networks of rna secondary structures. *Bull. Math. Biol.*, 59:339–397, 1997.
- [54] V. Reinharz, F. Major, and J. Waldspühl. Towards 3D structure prediction of large rna molecules: an integer programming framework to insert local 3D motifs in rna secondary structure. *Bioinformatics*, 28:207–214, 2012.
- [55] J. Ren, B. Rastegari, A. Condon, and H. Hoos. Hotknots: Heuristic prediction of rna secondary structures including pseudoknots. *RNA*, 11:1494–1504, 2005.
- [56] J. Reuter and D. Mathews. RNAstructure: software for rna secondary structure prediction and analysis. *BMC Bioinf.*, 11:129, 2010.
- [57] E. Rivas and S. Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068, 1999.
- [58] J. Ruan, G. Stormo, and W. Zhang. An iterated loop matching approach to the prediction of rna secondary structures with pseudoknots. *Bioinformatics*, 20:58–66, 2004.
- [59] T. Schlick. *Molecular Modeling and Simulation: An interdisciplinary Guide*. Springer, 2010.
- [60] P. Schuster. Prediction of rna secondary structures: from theory to models and real molecules. *Rep. Prog. Phys.*, 69:1419–1477, 2006.
- [61] P. Schuster. Prediction of RNA secondary structures: from theory to models and real molecules. *Rep. Prog. Phys.*, 69:1419, 2006.
- [62] P. Schuster, W. Fontana, P. Stadler, and I. Hofacker. From Sequences to Shapes and Back: A Case Study in RNA Secondary Structures. *Proc. Roy. Soc. (London) B*, 255:279–284, 1994.
- [63] E. Senter, S. Sheik, I. Dotu, Y. Ponty, and P. Clote. Using the fast fourier transform to accelerate the computational search for rna conformational switches. *PLoS ONE*, 2012. in press.
- [64] R. Shetty, D. Endy, and T. Knight Jr. Engineering BioBrick vectors from BioBrick parts. *J. Biol. Eng.*, 2:5, 2008.
- [65] M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch, and S. Steinberg. Compilation of trna sequences and sequences of trna genes. *Nucl. Acids Res.*, 26:148–153, 1998.
- [66] M. Stich, C. Briones, and S. Manrubia. On the structural repertoire of pools of short, random rna sequences. *J. Theor. Biol.*, 252:750–763, 2008.
- [67] M. Stich, E. Lázaro, and S. Manrubia. Phenotypic effect of mutations in evolving populations of rna molecules. *BMC Evol. Biol.*, 10:46, 2010.
- [68] M. Stich, E. Lázaro, and S. Manrubia. Variable mutation rates as an adaptive strategy in replicator populations. *PLoS ONE*, 5:e11186, 2010.
- [69] M. Stich and S. Manrubia. Motif frequency and evolutionary search times in rna populations. *J. Theor. Biol.*, 280:117–126, 2011.
- [70] J. Stombaugh, C. Zirbel, E. Westhof, and N. Leontis. Frequency and isostericity of rna base pairs. *Nucl. Acids Res.*, 37:2294–2312, 2009.
- [71] A. Taneda. Modena: a multi-objective rna inverse folding. *Adv. Appl. Bioinf. Chem.*, 4(1), 2011.
- [72] P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. MIT Press, 2005.
- [73] E. vanNimwegen, J. Crutchfield, and M. Huynen. Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. USA*, 96:9716–9720, 1999.
- [74] G. Varani and W. McClain. The gu wobble base pair. *EMBO Rep.*, 1:18–23, 2000.

- [75] J. Waldspühl, S. Devadas, B. Berger, and P. Clote. Efficient algorithms for probing the rna mutation landscape. *PLoS Comput. Biol.*, 4:e1000124, 2008.
- [76] A. Walter, D. Turner, J. Kim, M. Lytle, P. Müller, D. Mathews, and M. Zuker. Coaxial stacking of helices enhances binding of oligoribonucleotides and improves predictions of rna folding. *PNAS*, 91:9218–9222, 1994.
- [77] K. Wilkinson, E. Merino, and K. Weeks. RNA SHAPE chemistry reveals nonhierarchical interactions dominate equilibrium structural transitions in tRNA<sup>Asp</sup>. *J. Am. Chem. Soc.*, 127:4659–4667, 2005.
- [78] M. Wu and I. Tinoco Jr. Rna folding causes secondary structure rearrangement. *Proc. Natl. Acad. Sci. USA*, 95:11555–11560, 1998.
- [79] J. Zadeh, B. Wolfe, and N. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comput. Chem.*, 32:439–452, 2011.
- [80] M. Zuker and P. Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucl. Acids Res.*, 9:133–148, 1981.

## Capítulo 15

# Estructura y organización del DNA

*Davide Bau*

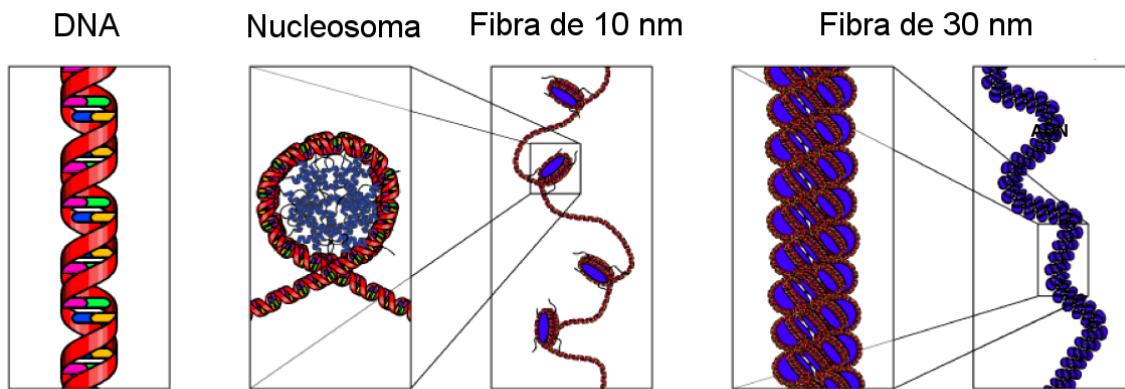
### 15.0.1. Niveles de organización del DNA en la cromatina

El DNA es un biopolímero natural cuya función es, a través de segmentos de DNA llamados genes, la de transportar la información genética usada en el desarrollo y el funcionamiento de casi todos los organismos vivos conocidos. El DNA está formado por cadenas de *nucleótidos*, que son unidades compuestas por una *base nitrogenada*, un azúcar de cinco carbonos llamado *desoxirribosa* y un *grupo fosfato* (a través del cual están unidos el uno al otro) (ver Sección 7.3 y Figura 14.2). Las cuatro bases que se encuentran en el DNA son *adenina* (abreviado A), *citosina* (C), *guanina* (G) y *timina* (T). Una molécula de DNA consiste de dos hebras de secuencias de nucleótidos que interactúan entre ellas. En cadenas opuestas, la adenina empareja con la timina a través de 2 puentes de hidrógeno, mientras que la guanina empareja con la citosina a través de 3 puentes de hidrógeno, formando la típica estructura de doble hélice. Las dos hebras que forman la *estructura de doble hélice* se orientan en direcciones opuestas entre sí y son por lo tanto en una orientación *anti-paralela*. Los enlaces de hidrógeno y apilamiento de bases (las interacciones entre las bases nitrogenadas aromáticas) son las fuerzas que más estabilizan la estructura de doble hélice del DNA [23].

Dentro de las células el DNA está organizado en estructuras llamadas *cromosomas*. Dentro de los cromosomas, el DNA se compacta enrollándose en complejos de ocho proteínas llamadas *histonas* para formar nucleosomas (Figura 15.1), las unidades básicas de empaquetamiento del DNA en eucariotas. La partícula central del *nucleosoma* se compone de aproximadamente 147 pares de bases de DNA enrolladas en torno a un octámero de histonas que consta de 2 copias de cada una de las proteínas del núcleo de las histonas H2A, H2B, H3 y H4 [16]. Las partículas centrales de un nucleosoma están conectadas por tramos de “DNA enlazante”, que puede extenderse hasta aproximadamente 80 pares de bases. Secuencias repetidas de nucleosomas compactan la cromatina para formar la *fibra de 10 nm*, que se denomina “*collar de perlas*” (Figura 15.1) [12]. Este tipo de disposición puede plegarse posteriormente para formar una *fibra de 30 nm* (Figura 15.1), una estructura aún más compacta que la fibra de 10 nm. Estudios de microscopía electrónica han demostrado que este tipo de estructura es muy dinámica y puede desplegarse para volver al estado de 10 nm. Los nucleosomas se pliegan a través de una serie sucesiva de estructuras de orden superior para formar los cromosomas. Los cromosomas no están distribuidos al azar dentro del núcleo de la célula sino que, al menos en eucariotas superiores, ocupan lugares específicos denominados “*territorios cromosómicos*”. Los territorios cromosómicos tienen forma irregular y están organizados en pequeños subdominios, partes del núcleo ocupado por

cromosomas diferentes. En general, los cromosomas ricos en genes están situados cerca del centro del núcleo, mientras que los cromosomas con pocos genes están situados cerca de la periferia del núcleo, a menudo tocando la membrana nuclear [5]. Por otra parte, los territorios cromosómicos no son compartimentos herméticamente cerrados, sino que permiten que cromosomas diferentes, en la periferia de sus territorios interactúen, por ejemplo, acercando distintos genes con un mismo promotor.

La disposición tridimensional de la cromatina es responsable de guiar las interacciones entre los genes y sus correspondientes elementos reguladores. Por lo tanto, juega un papel crucial en la determinación y en el control de las partes del DNA que están siendo transcritas. La estructura de la cromatina depende de varios factores, como la fase del ciclo celular (durante la interfase la cromatina es menos compacta para permitir la transcripción y la replicación del DNA) y si los genes están en un estado activo (transcripción) o pasivo. Los genes que están siendo transcritos, de hecho, están menos empaquetados y se encuentran junto a la RNA polimerasa. La cromatina en este estado se conoce como *eucromatina*. Los genes inactivos se asocian con proteínas estructurales y están más empaquetados, la cromatina en este estado se conoce como *heterocromatina* [6]. Modificaciones de las histonas como *metilación* y *acetilación* son también responsables de los cambios en la estructura local de la cromatina. Finalmente, durante la anafase, la cromatina está todavía más empaquetada para facilitar la segregación de los cromosomas.



**Figura 15.1:** Niveles de organización del DNA. El DNA se compacta enrollándose en complejos de ocho proteínas llamadas histonas para formar nucleosomas. Secuencias repetidas de nucleosomas forman 10 nm (“collar de perlas”), la cual puede plegarse ulteriormente para formar la fibra de 30 nm (imagen adaptada de Wikipedia).

### 15.0.2. Determinación de la estructura de dominios genómicos

La *arquitectura en tres dimensiones* (3D) de un genoma entero o de parte del mismo (un *dominio genómico*), es responsable de promover los contactos entre los genes y sus elementos reguladores, lo que les permite llevar a cabo su función [21]. El conocimiento de la organización espacial de los dominios genómicos es por tanto esencial para comprender cómo se regulan los genes en las células vivas.

El advenimiento de nuevas tecnologías de secuenciación ha permitido obtener una gran cantidad de datos sobre la secuencias genómica de diferentes especies. Proyectos como 1000 Genomes [1] o ENCODE [4] han permitido obtener un listado de variaciones genéticas (diferencias en las secuencias de DNA entre una población o entre especies), elementos funcionales y reguladores que controlan las células y la actividad de los genes.

Sin embargo, esta información por sí sola, es decir, la localización de genes y elementos funcionales a lo largo de la secuencia del genoma, no permite conocer la posición espacial relativa de los genes respecto de sus elementos reguladores, información esencial para saber cómo se regulan [21]. Para llevar a cabo la transcripción (Figura 7.1), un gen debe encontrarse próximo a moléculas específicas que la regulan, como las RNA-polimerasas, un conjunto de proteínas con carácter enzimático.

Se puede disponer, si bien parcialmente, de esta información a través de técnicas de espectroscopía, que permiten determinar la posición espacial de los genes y elementos reguladores a baja resolución. Recientemente, se han desarrollado nuevos métodos bioquímicos de medición de la frecuencia de las interacciones entre pares de *loci* situados en el mismo o en diferentes cromosomas. Estos métodos, conocidos como *métodos 3C* [7, 10, 15, 20, 24], permiten superar la baja resolución de las técnicas de espectroscopía. Los métodos basados en 3C “capturan” los *loci* que están “interactuando” (o sea, que están cerca) utilizando un método de *cross-link* con formaldehído [10]. En la práctica, estas técnicas analíticas capturan los *loci* que están cerca formando un enlace químico entre ellos (a través del formaldehído); el resultado es un “calco” de la estructura de la cromatina en forma de frecuencias de interacciones entre loci cercanos.

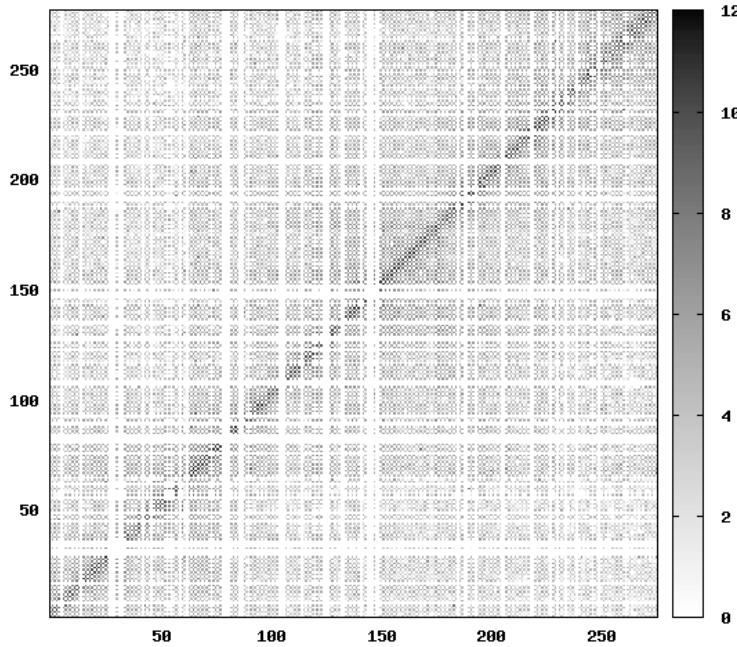
La combinación de estos métodos bioquímicos con técnicas de secuenciación de última generación ha permitido mapear en alta resolución las interacciones entre *loci* de dominios genómicos y de genomas enteros. En particular, los *métodos 5C* y *Hi-C* son métodos basados en 3C que permiten la determinación simultánea de las frecuencias de interacción de cromosomas enteros (5C) o de genomas completos (Hi-C) [8, 9]. En la Figura 15.2 se puede ver un ejemplo de matriz 5C. El método Hi-C permite abarcar regiones más grandes que el método 5C, pero a menor resolución. Al integrar estos tipos de datos con métodos computacionales (por ejemplo usando simulaciones de Monte Carlo [1]), como la Integrative Modeling Platform (IMP) [19], es posible determinar la estructura 3D de un dominio genómico a una resolución sin precedentes. Este método es similar a la determinación de la estructura de proteínas mediante espectroscopía NMR, donde, en lugar de las matrices 5C (o Hi-C), se utilizan espectros de NMR.

## Primeros modelos de dominios genómicos

Los primeros modelos de baja resolución en 3D de la cromatina se generaron utilizando métodos de física de polímeros [7, 17] y de dinámica molecular [22]. Estos métodos han permitido obtener información sobre las características físicas de la cromatina, como su compactación, densidad y flexibilidad.

Recientemente, el desarrollo de *nuevos métodos que integran datos experimentales con métodos computacionales* (principalmente explotando métodos de optimización) ha permitido llegar a una resolución más alta de la estructura 3D de los dominios genómicos [3, 11, 13, 14]. Ejemplos de estas metodologías incluyen la determinación de la arquitectura de baja resolución del *locus* de la inmunoglobulina de cadena pesada (IGH) mediante la integración de datos de espectroscopía FISH [14], la determinación del modelo 3D del genoma de levadura usando datos de interacción de cromatina en todo su genoma [11] y la determinación de la arquitectura del dominio de la  $\alpha$ -globina humana [3] mediante la integración de un amplio conjunto de datos 5C en IMP.

En la siguiente sección explicaremos como determinar la estructura 3D de cromatina utilizando IMP. La ventaja de utilizar este sistema sobre otros es su flexibilidad, IMP permite integrar datos proveniente de diferentes técnicas experimentales (lo cual, en principio, permite aumentar la resolución de los modelos generados) y representar cualquier “objeto” (una macromolécula en este caso) a diferente escalas.



**Figura 15.2:** Ejemplo de una matriz 5C. La matriz muestra el logaritmo de las interacciones, donde 0 (blanco) corresponde a baja frecuencia de interacción, y 12 (negro) corresponde a alta frecuencia de interacción.

### 15.0.3. Determinación de estructura con IMP

El *software Integrative Modeling Platform*<sup>1</sup> (IMP) traduce los datos experimentales en objetos 3D y en *restricciones espaciales* entre ellos. Una proteína, por ejemplo, puede ser representada a diferentes escalas: empleando una esfera por cada aminoácido (lo que resultaría en modelos de baja resolución) o por cada átomo (lo que resultaría en modelos de alta resolución). Hay varios tipos de restricciones que pueden ser añadidas al modelo, como distancias entre las esferas (aminoácido/átomos) o volumen total ocupado por el modelo generado.

Una vez que los datos experimentales hayan sido representados como objetos y restricciones, IMP busca un conjunto de coordenadas de esos objetos (o sea su estructura tridimensional) que mejor se adapta a las restricciones impuestas, derivadas de los datos experimentales mismos.

En todos los métodos de determinación y predicción de estructura se genera un gran *conjunto de modelos 3D compatibles con los datos experimentales*. Dado que los datos experimentales de tipo 3C son en general un promedio medido sobre una población de células, la generación de un gran conjunto de soluciones permite representar las diferentes conformaciones capturadas por los experimentos.

IMP proporciona un conjunto de herramientas para que los desarrolladores de métodos computacionales puedan convertir datos experimentales en restricciones espaciales, para implementar técnicas de optimización y análisis, y para poder construir modelos por integración de datos provenientes de diferente experimentos. En principio, con IMP es posible modelar cualquier objeto en tres dimensiones (por ejemplo una biomolécula), siempre que el objeto pueda ser representado adecuadamente en el marco de IMP (ver siguiente apartado) y que los datos experimentales disponibles sobre la estructura

<sup>1</sup>Integrative Modeling Platform. <https://integrativemodeling.org>

que se está estudiando (el objeto) puedan traducirse en restricciones espaciales entre las partículas que la representan.

## Colección de datos

El primer paso en la determinación de la estructura por IMP, como en cualquier método de modelado por integración de datos, es la *colección de datos experimentales*. Actualmente, sólo los datos 5C y Hi-C se han integrado en IMP para determinar la estructura de dominios genómicos. Por tanto, vamos a entrar en detalle en la determinación de la estructura por IMP considerando este tipo de datos como punto de partida. Sin embargo, conviene recordar que, en principio, IMP permite la integración de datos de múltiples tipos; IMP, por ejemplo, ha sido usado por la determinación de la estructura del complejo del poro nuclear (NPC), un complejo de 456 proteínas, integrando datos de microscopía, ultracentrifugación y cromatografía de afinidad, entre otros [2]. La ventaja de utilizar datos de diferentes experimentos es la posibilidad de obtener modelos de mayor resolución en comparación con los métodos de técnicas experimentales individuales (datos diferentes pueden aportar información complementaria sobre la estructura de una molécula, permitiendo revelar mas detalles que en el caso de métodos de técnicas experimentales individuales).

Los datos experimentales de tipo 3C representan la frecuencia de interacción entre fragmentos de restricción (los trozos de DNA resultante de la digestión enzimática), y pueden ser considerados como un indicador de la proximidad espacial de los fragmentos relacionados (es decir, de las partes de DNA que están espacialmente cerca) [15, 18]. La asunción que se hace es que si la frecuencia de interacción entre dos *loci* es muy alta, estos dos *loci* estarán, en media, cerca uno del otro. Hay que recordar que el término “interacción” se refiere a la formación de un *cross-link*(enlace químico) mediante formaldehído entre dos *loci* que están cerca (en la estructura 3D).

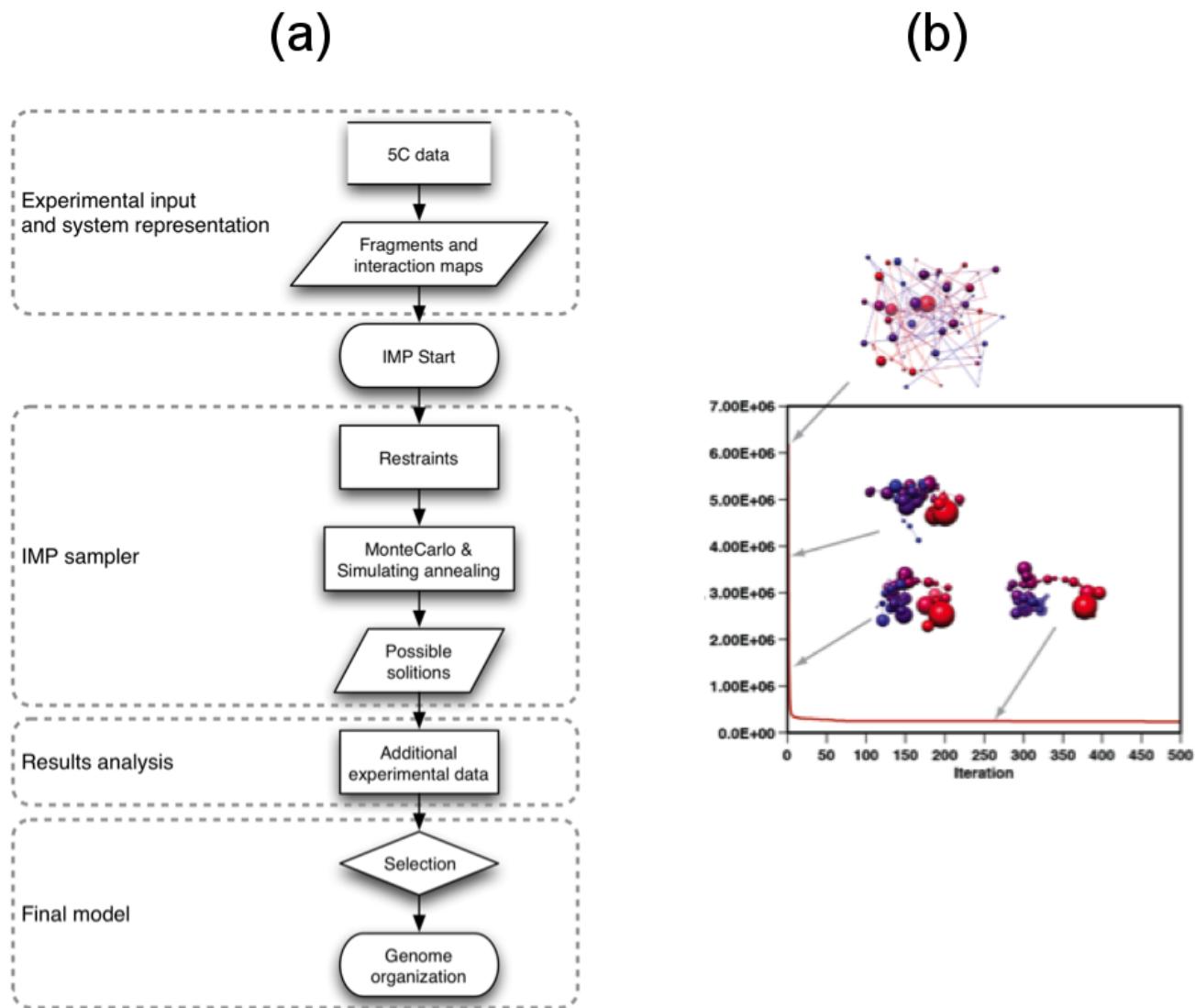
Para tener en cuenta las posibles sesgos/desviaciones en los datos experimentales, los datos son normalizados (utilizando la función “*log*”) y transformados en *Z-scores* antes de ser introducido en IMP (Sección 3.3.3).

## La estrategia de IMP

El marco conceptual de IMP se compone de cuatro etapas: la representación (asignar una partícula a cada fragmento de DNA), la puntuación (la magnitud usada para evaluar los modelos generados), la optimización (la generación de los modelos) y el análisis. Ver esquema en Figura 15.3a.

La *representación* del sistema (es decir, el conjunto de objetos que representan la macromolécula biológica bajo investigación) debe ser suficientemente detallada para caracterizar exhaustivamente los datos experimentales sin hacer la parte computacional inviable como consecuencia de una descripción del sistema demasiado detallada. Mientras una representación detallada de una molécula podría dar lugar a modelos de mayor resolución, una representación limitada permite una búsqueda más exhaustiva del espacio conformacional, siendo computacionalmente menos exigente y por tanto más adecuada para la representación de las macromoléculas biológicas, como la cromatina. IMP representa un objeto como un conjunto jerárquico de partículas (cada partícula representa un fragmento de DNA) que puede ser representado a diferente niveles de detalles, y sus propiedades (por ejemplo, su posición en el espacio). Esto permite una representación flexible del sistema a diferentes resoluciones (en la Figura 15.3b, el dominio de la  $\alpha$ -globina humana representada en IMP, en diferentes estados de la optimización).

El paso clave en cualquier método de determinación de estructura es la correcta *evaluación de los modelos generados* para identificar las soluciones que mejor representan los datos de entrada. Por



**Figura 15.3:** Protocolo de optimización de IMP. (a) Diagrama de flujo del protocolo de optimización de IMP. (b) Representación esquemática de un proceso de optimización típico para la simulación del dominio génico de la  $\alpha$ -globina humana en células K562. El modelo comienza con una configuración aleatoria y termina con una configuración óptima después de la minimización de la función-objetivo de IMP reduciendo contemporáneamente las violaciones a todas las restricciones impuestas. Los modelos son muestra de cuatro instantáneas diferentes durante la optimización. Cada fragmento de restricción se representa como una partícula de radio proporcional a su volumen excluido.

tanto, una relación cuantitativa entre la representación del sistema y el sistema biológico (por ejemplo, los datos experimentales) debe ser formulada. En IMP esto se logra mediante la asignación a cada modelo de una puntuación consistente en la suma de funciones individuales llamadas *funciones de densidad de probabilidad* (PDFs) que afectan a todas las partículas en el sistema. Una PDF es una función que describe la probabilidad de una variable (por ejemplo, las coordenadas de cada partícula del modelo) de asumir un valor dado (por ejemplo, la distancia entre dos partículas derivadas de los datos experimentales; un ejemplo es el RMSD – Ecuación 10.3). En IMP, la *puntuación* es un número que mide la calidad de un modelo durante (y al final de) la optimización; es una magnitud que se optimiza

durante el modelado. Al concluir el modelado, los modelos con mejor puntuación serán elegido para el análisis final. La puntuación final de un modelo, o la función-objetivo de IMP, consistirá entonces en la suma de cada PDF individual que afecta a todas las partículas del sistema (cuanto más baja es la puntuación, mejor es el modelo).

Una vez que el sistema ha sido debidamente representado y las restricciones entre las partículas han sido definidas, IMP busca una solución del sistema (es decir, genera un modelo 3D) reduciendo simultáneamente las violaciones a todas las restricciones impuestas; es decir, IMP genera modelos que mejor representan los datos experimentales de entrada. El espacio conformacional es explorado en términos de *optimización de energía* (puntuación) (por ejemplo, utilizando el método de Monte Carlo) y mediante la generación de un gran número de modelos.

La posición espacial de cada partícula se determina considerando una serie de osciladores armónicos (o muelles; son las PDF implementadas para este problema) aplicados entre pares de partículas, cuyo propósito es mantener las partículas a una distancia de equilibrio derivada, para cada par de partículas, de los datos experimentales. Para el modelado de cromatina se utilizan tres tipos de osciladores: (i) los osciladores armónicos ( $H_{i,j}$ ), cuyo objetivo es mantener dos partículas a una distancia específica, (ii) los osciladores armónicos inferiores ( $lbH_{i,j}$ ), que evitan que dos partículas no se acerquen más que una distancia de equilibrio dada y, (iii) los osciladores armónicos superiores ( $ubH_{i,j}$ ), que obligan a cada par de partículas a estar a una distancia menor de una distancia determinada. Las funciones exactas de estos tres osciladores armónicos son:

$$H_{i,j} = k(d_{i,j} - d_{i,j}^0)^2$$

$$lbH_{i,j} = \begin{cases} k(d_{i,j} - d_{i,j}^0)^2 & \text{si } d_{i,j} \leq d_{i,j}^0 \\ 0 & \text{si } d_{i,j} > d_{i,j}^0 \end{cases}$$

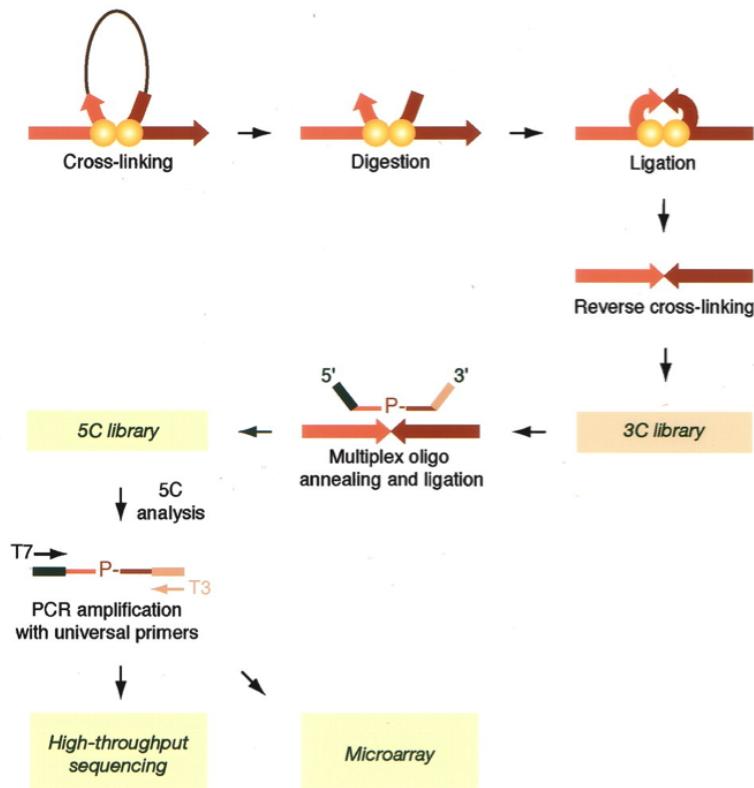
$$ubH_{i,j} = \begin{cases} k(d_{i,j} - d_{i,j}^0)^2 & \text{si } d_{i,j} \geq d_{i,j}^0 \\ 0 & \text{si } d_{i,j} < d_{i,j}^0 \end{cases}$$

donde  $d_{i,j}$  es la distancia entre las partículas  $i$  y  $j$  durante la simulación,  $d_{i,j}^0$  es la distancia de equilibrio entre las partículas  $i$  y  $j$  obtenida de los datos experimentales, y  $k$  es la constante de fuerza aplicada a los osciladores armónicos, que escala la penalización por no satisfacer una restricción impuesta (es decir,  $k$  penaliza las conformaciones en las cuales las partículas están a una distancia diferente de la distancia obtenida de los datos experimentales). El tipo de restricción ( $H_{i,j}$ ,  $lbH_{i,j}$  o  $ubH_{i,j}$ ) y la distancia de equilibrio aplicada a cada partícula se definen en base a los datos experimentales y a tres parámetros de IMP (un límite inferior y uno superior de *Z-score* y una distancia máxima entre dos fragmentos). Los valores óptimos para estos tres parámetros se determinan empíricamente [3].

Por último, el *análisis estructural* del conjunto resultante de soluciones compatibles con las restricciones impuestas (es decir, con los datos experimentales de entrada) tiene como objetivo *validar los modelos* y revelar sus aspectos importantes. No existiendo estructuras reales con la cual comparar los resultado (como en el caso de las proteínas), es importante confrontar los modelos generado con datos biológicos comprobados (por ejemplo, utilizando datos de espectroscopia FISH o introduciendo en los modelos datos sobre las actividad de los genes) para ver si las estructura 3D generadas son compatibles con la información disponible sobre la molécula estudiada. Este tipo de análisis, junto con la observación de las características físicas de los modelos (por ejemplo densidad, grado de compactación, variabilidad de la estructura), forman una parte esencial de la determinación de estructuras por integración de datos.

#### 15.0.4. Ejemplo de determinación de estructura por IMP

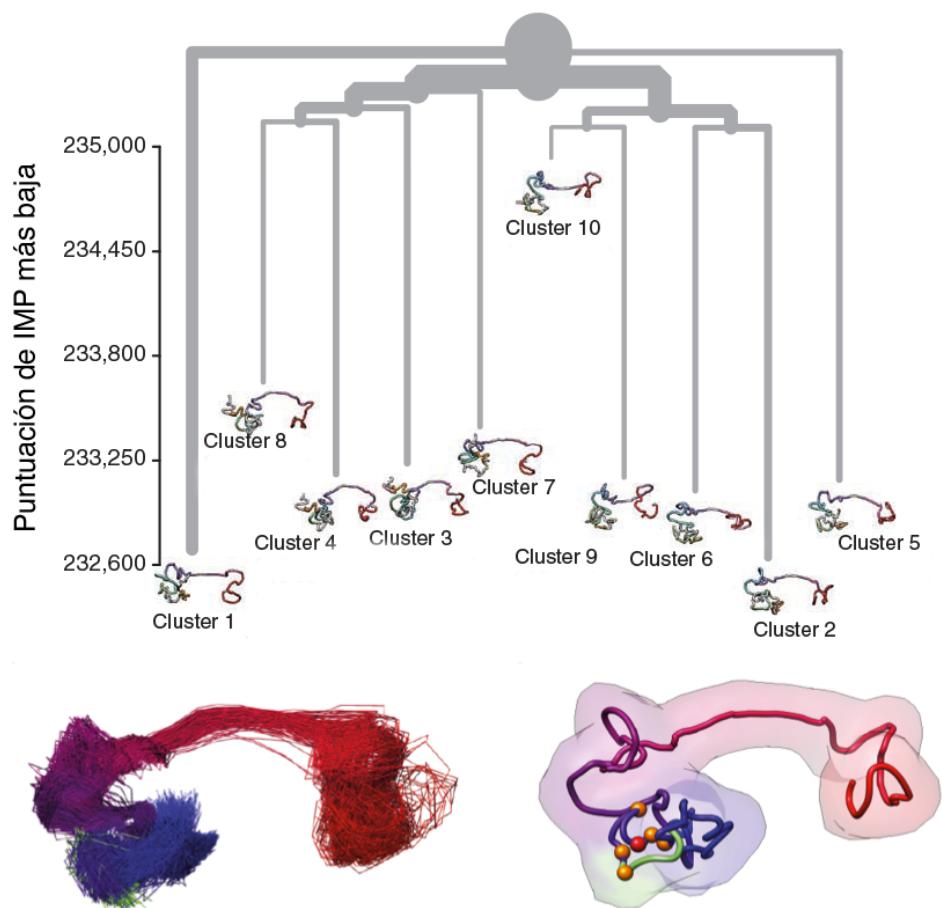
IMP fue recientemente aplicado para la determinación de la arquitectura 3D del dominio génico de la  $\alpha$ -globina humana, una región de ~500 Kb situada en el cromosoma humano 16 [3]. Un mapa de interacción completa del *locus* de la  $\alpha$ -globina se obtuvo mediante la aplicación de la técnica experimental 5C (Figura 15.4) para células cancerosas K562, donde se piensa que tienen lugar interacciones de larga distancia (medida en Kb dentro de la secuencia genómica) entre los genes de  $\alpha$ -globina y sus elementos reguladores. A cada fragmento de restricción resultante por digestión con el enzima HindIII le fue asignada una partícula de radio proporcional al número de bases de DNA en el fragmento. El HindIII es un enzima de restricción que reconoce la secuencia específica de DNA “AAGCTT”. Cuando la enzima encuentra esta secuencia de nucleótidos, produce un corte que, dependiendo de la frecuencia y de la posición de la secuencia reconocida en el DNA, genera un número de fragmentos de tamaño diferente. En este caso, el dominio  $\alpha$ -globina estuvo representado por un conjunto de 70 partículas (derivadas de la digestión por HindIII), a las cuales se asignaron 1049 restricciones.



**Figura 15.4:** Representación esquemática de la técnica 5C. Una librería 5C, generada a partir de una librería 3C, puede ser analizada por técnicas de *microarrays* o de secuenciación cuantitativa. La digestión (generalmente por el enzima de restricción HindIII) fragmenta la cromatina en un número de fragmentos que depende del número de sitios de restricción del DNA (secuencias características de nucleótidos dentro de una molécula de DNA.) (Adaptado de [9]).

De estas restricciones, que corresponden a los osciladores armónicos descritos anteriormente, 235 eran osciladores armónicos, 709 osciladores armónicos inferiores, y 105 osciladores armónicos superiores. Para realizar una búsqueda exhaustiva del espacio conformacional, se generaron un total de 50.000 simulaciones independientes. En el caso de una macromolécula biológica como la cromatina, este espacio es ingente y como consecuencia, es necesario generar un gran número de modelos tridimensionales

independientes. Cada simulación genera un modelo correspondiente a la mejor solución (en términos de la puntuación descrita anteriormente) encontrada por cada modelo generado (Figura 15.3b). Los 10.000 modelos con la puntuación de IMP más baja (es decir, los modelos que mejor representan los datos de entrada) fueron seleccionados y agrupados estructuralmente, lo que resultó en 393 grupos, con los 10 grupos principales representando el 26 % de los modelos seleccionados (Figura 15.5). Este tipo de análisis da una idea de la variabilidad de las estructuras generadas. Si se obtuviera un único grupo de estructuras, se deduciría que los modelos generados tienen poca variabilidad (estructura bien definida). El elevado número de grupos estructurales (393) encontrado en este caso, revela una alta diversidad en los modelos generados. Esta variabilidad podría ser relacionada con el tipo celular estudiado, siendo la línea celular K562 derivada de tejido canceroso. Estas células presentan un cariotipo (número de cromosomas) muy variable y distinto del normal, y un alto nivel de expresión en los genes, factores que podrían aumentar la variabilidad de la cromatina en estas células. Los modelos resultantes fueron validados indirectamente, mediante la asignación de las características estructurales locales de la cromatina publicada por el consorcio ENCODE [4] y por experimentos de FISH, que validaron el tamaño y la forma. En la Figura 15.5 los diferentes colores representan los genes y las esferas sus elementos reguladores [3].



**Figura 15.5:** Conjunto de soluciones del dominio de la  $\alpha$ -globina humana (10 grupos principales).

### **15.0.5. Conclusiones**

En este capítulo hemos introducido los diferentes niveles de organización del DNA en el núcleo celular. Para poder acomodarse en el núcleo celular, el DNA se compacta alrededor de una proteínas llamadas histonas, formando la cromatina. La cromatina es un conjunto de DNA y proteínas; la modificación de su estructura 3D permite que elementos genómicos (por ejemplo genes, promotores, y reguladores de los genes), se unan (o separan) para asegurar una correcta actividad genómica y garantizar un correcto funcionamiento de la célula. Para entender como estos procesos tienen lugar es necesario conocer la estructura 3D de la cromatina. Hemos visto una aplicación concreta que permite, a partir de datos experimentales, determinar la disposición espacial de los genes, con respecto de su elementos reguladores (es decir, su estructura 3D). Las mejoras de la técnica de secuenciación (en términos de calidad de datos producidos, tiempos experimentales y costes) permitirán obtener más datos de tipo 3C que contienen, aunque de manera indirecta, información sobre la estructura 3D de la cromatina. Integrándolos con métodos computacionales permiten revelar detalles sobre los mecanismo de plegamiento de la cromatina, con el fin último de añadir información importante para entender cómo el genoma funciona, esencial para desarrollar nuevas técnicas terapéuticas, como las terapias génicas.

## 15.1. Bibliografía

- [1] G. R. Abecasis, A. Auton, L. D. Brooks, M. A. DePristo, R. M. Durbin, R. E. Handsaker, H. M. Kang, G. T. Marth, and G. A. McVean. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.
- [2] F. Alber, S. Dokudovskaya, L. M. Veenhoff, W. Zhang, J. Kipper, D. Devos, A. Suprapto, O. Karni-Schmidt, R. Williams, B. T. Chait, M. P. Rout, and A. Sali. Determining the architectures of macromolecular assemblies. *Nature*, 450(7170):683–94, 2007.
- [3] D. Bau, A. Sanyal, B. R. Lajoie, E. Capriotti, M. Byron, J. B. Lawrence, J. Dekker, and M. A. Marti-Renom. The three-dimensional folding of the alpha-globin gene domain reveals formation of chromatin globules. *Nat Struct Mol Biol*, 18(1):107–14, 2011.
- [4] E. Birney, J. A. Stamatoyannopoulos, A. Dutta, R. Guigo, T. R. Gingeras, E. H. Margulies, Z. Weng, M. Snyder, E. T. Dermitzakis, R. E. Thurman, M. S. Kuehn, C. M. Taylor, S. Neph, C. M. Koch, S. Asthana, A. Malhotra, I. Adzhubei, J. A. Greenbaum, R. M. Andrews, P. Flicek, P. J. Boyle, H. Cao, N. P. Carter, G. K. Clelland, S. Davis, N. Day, P. Dhami, S. C. Dillon, M. O. Dorschner, H. Fiegler, P. G. Giresi, J. Goldy, M. Hawrylycz, A. Haydock, R. Humbert, K. D. James, B. E. Johnson, E. M. Johnson, T. T. Frum, E. R. Rosenzweig, N. Karnani, K. Lee, G. C. Lefebvre, P. A. Navas, F. Neri, S. C. Parker, P. J. Sabo, R. Sandstrom, A. Shafer, D. Vetric, M. Weaver, S. Wilcox, M. Yu, F. S. Collins, J. Dekker, J. D. Lieb, T. D. Tullius, G. E. Crawford, S. Sunyaev, W. S. Noble, I. Dunham, F. Denoeud, A. Reymond, P. Kapranov, J. Rozowsky, D. Zheng, R. Castelo, A. Frankish, J. Harrow, S. Ghosh, A. Sandelin, I. L. Hofacker, R. Baertsch, D. Keefe, S. Dike, J. Cheng, H. A. Hirsch, E. A. Sekinger, J. Lagarde, J. F. Abril, A. Shahab, C. Flamm, C. Fried, J. Hackermuller, J. Hertel, M. Lindemeyer, K. Missal, A. Tanzer, S. Washietl, J. Korbel, O. Emanuelsson, J. S. Pedersen, N. Holroyd, R. Taylor, D. Swarbreck, N. Matthews, M. C. Dickson, D. J. Thomas, M. T. Weirauch, J. Gilbert, et al. Identification and analysis of functional elements in 1genome by the encode pilot project. *Nature*, 447(7146):799–816, 2007.
- [5] T. Cremer and C. Cremer. Chromosome territories, nuclear architecture and gene regulation in mammalian cells. *Nat Rev Genet*, 2(4):292–301, 2001.
- [6] R. T. Dame. The role of nucleoid-associated proteins in the organization and compaction of bacterial chromatin. *Mol Microbiol*, 56(4):858–70, 2005.
- [7] J. Dekker, K. Rippe, M. Dekker, and N. Kleckner. Capturing chromosome conformation. *Science*, 295(5558):1306–11, 2002.
- [8] J. Dostie and J. Dekker. Mapping networks of physical interactions between genomic elements using 5c technology. *Nat Protoc*, 2(4):988–1002, 2007.
- [9] J. Dostie, T. A. Richmond, R. A. Arnaout, R. R. Selzer, W. L. Lee, T. A. Honan, E. D. Rubio, A. Krumm, J. Lamb, C. Nusbaum, R. D. Green, and J. Dekker. Chromosome conformation capture carbon copy (5c): a massively parallel solution for mapping interactions between genomic elements. *Genome Res*, 16(10):1299–309, 2006.
- [10] J. Dostie, Y. Zhan, and J. Dekker. Chromosome conformation capture carbon copy technology. *Curr Protoc Mol Biol*, Chapter 21:Unit 21 14, 2007.
- [11] Z. Duan, M. Andronescu, K. Schutz, S. McIlwain, Y. J. Kim, C. Lee, J. Shendure, S. Fields, C. A. Blau, and W. S. Noble. A three-dimensional model of the yeast genome. *Nature*, 465(7296):363–7, 2010.
- [12] G. Felsenfeld and M. Groudine. Controlling the double helix. *Nature*, 421(6921):448–53, 2003.
- [13] J. Fraser, M. Rousseau, S. Shenker, M. A. Ferraiuolo, Y. Hayashizaki, M. Blanchette, and J. Dostie. Chromatin conformation signatures of cellular differentiation. *Genome Biol*, 10(4):R37, 2009.
- [14] S. Jhunjhunwala, M. C. van Zelm, M. M. Peak, S. Cutchin, R. Riblet, J. J. van Dongen, F. G. Grosveld, T. A. Knoch, and C. Murre. The 3d structure of the immunoglobulin heavy-chain locus: implications for long-range genomic interactions. *Cell*, 133(2):265–79, 2008.
- [15] E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–93, 2009.
- [16] K. Luger, A. W. Mader, R. K. Richmond, D. F. Sargent, and T. J. Richmond. Crystal structure of the nucleosome core particle at 2.8 a resolution. *Nature*, 389(6648):251–60, 1997.

- [17] J. Mateos-Langerak, M. Bohn, W. de Leeuw, O. Giromus, E. M. Manders, P. J. Verschure, M. H. Indemans, H. J. Gierman, D. W. Heermann, R. van Driel, and S. Goetze. Spatially confined folding of chromatin in the interphase nucleus. *Proc Natl Acad Sci U S A*, 106(10):3812–7, 2009.
- [18] A. Miele, K. Bystricky, and J. Dekker. Yeast silent mating type loci form heterochromatic clusters through silencer protein-dependent long-range interactions. *PLoS Genet*, 5(5):e1000478, 2009.
- [19] D. Russel, K. Lasker, B. Webb, J. Velazquez-Muriel, E. Tjioe, D. Schneidman-Duhovny, B. Peterson, and A. Sali. Putting the pieces together: integrative modeling platform software for structure determination of macromolecular assemblies. *PLoS Biol*, 10(1):e1001244, 2012.
- [20] M. Simonis, P. Klous, E. Splinter, Y. Moshkin, R. Willemsen, E. de Wit, B. van Steensel, and W. de Laat. Nuclear organization of active and inactive chromatin domains uncovered by chromosome conformation capture-on-chip (4c). *Nat Genet*, 38(11):1348–54, 2006.
- [21] T. Takizawa, K. J. Meaburn, and T. Misteli. The meaning of gene positioning. *Cell*, 135(1):9–13, 2008.
- [22] G. Wedemann and J. Langowski. Computer simulation of the 30-nanometer chromatin fiber. *Biophys J*, 82(6):2847–59, 2002.
- [23] P. Yakovchuk, E. Protozanova, and M. D. Frank-Kamenetskii. Base-stacking and base-pairing contributions into thermal stability of the dna double helix. *Nucleic Acids Res*, 34(2):564–74, 2006.
- [24] Z. Zhao, G. Tavoosidana, M. Sjolinder, A. Gondor, P. Mariano, S. Wang, C. Kanduri, M. Lezcano, K. S. Sandhu, U. Singh, V. Pant, V. Tiwari, S. Kurukuti, and R. Ohlsson. Circular chromosome conformation capture (4c) uncovers extensive networks of epigenetically regulated intra- and interchromosomal interactions. *Nat Genet*, 38(11):1341–7, 2006.

## **Parte V**

# **Dinámica estructural y diseño de fármacos**



## Capítulo 16

# Diseño de fármacos asistido por ordenador

*Antonio Morreale, Almudena Perona, Javier Klett,  
Álvaro Cortés-Cabrera y Helena G. Dos Santos*

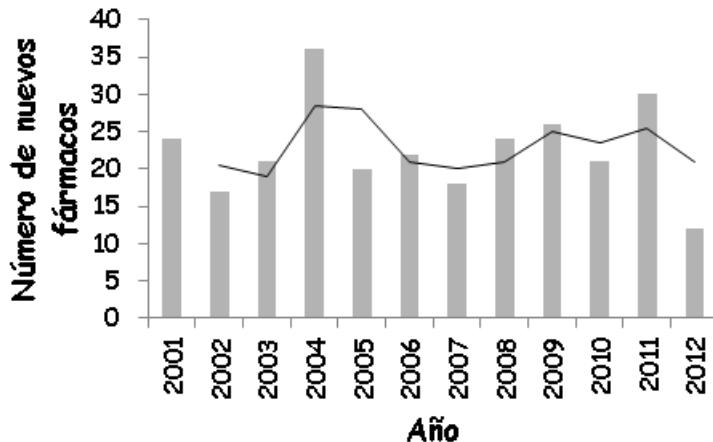
### 16.1. Introducción

De una manera muy general podemos decir que los sistemas biológicos comparten un lenguaje común de comunicación basado en las interacciones que se establecen entre distintos tipos de moléculas como proteínas, ácidos nucleicos y entidades químicas de menor tamaño denominadas genéricamente ligandos. Muchas enfermedades provienen precisamente de un mal funcionamiento de estas interacciones por lo que conocer cómo se producen nos va a permitir interferir en ellas a través del uso de ligandos que disminuyan, o incluso anulen, los efectos de la enfermedad. También se puede dar el caso de que lo que busquemos sean activadores de dichas interacciones.

La búsqueda y posterior comercialización de un nuevo fármaco es un proceso que requiere de un tremendo esfuerzo en investigación y desarrollo así como de una enorme inversión económica. Bajo el paradigma de que con el mayor número posible de candidatos a estudiar sería más probable encontrar nuevos fármacos, en los últimos 30 años se ha popularizado el empleo de técnicas experimentales de carácter masivo como la química combinatoria y el cribado de alto rendimiento (*high-throughput screening*). Sin embargo, el esfuerzo económico así invertido y el número de nuevos fármacos que han alcanzado el mercado no están en concordancia, y está claro que el número de estos está muy por debajo de las expectativas iniciales hechas en base a las técnicas masivas (Figura 16.1).

De alguna manera estos resultados tan inesperados han repercutido de forma positiva en el desarrollo paralelo de técnicas computacionales con el objetivo de servir como solución al cuello de botella existente en el actual modelo de búsqueda de nuevos fármacos. La idea subyacente es *intentar racionalizar y, en base al conocimiento, acelerar las etapas iniciales del diseño de fármacos*.

Los métodos teóricos, cuando están bien fundamentados e implementados, permiten seleccionar a partir de colecciones de millones de moléculas (*quimiotebas*) aquellos candidatos que tienen una mayor probabilidad de interaccionar con una *diana terapéutica* dada. Este reducido conjunto puede analizarse experimentalmente y aquellos compuesto que den señal de interacción con la diana, llamados *hits*, puede optimizarse hasta alcanzar los perfiles farmacocinéticos y farmacodinámicos adecuados y convertirse así en *leads*.



**Figura 16.1:** Evolución en el tiempo del número de nuevos fármacos que han alcanzado el mercado.

Desde una perspectiva teórica, y dependiendo de la información estructural que tengamos a nuestra disposición, se pueden presentar cuatro escenarios distintos que van desde el más favorable de todos (cuando tenemos información estructural de la diana y de alguno de sus ligandos, entonces empleamos *docking* y *cribado virtual*) pasando por dos casos intermedios (cuando conocemos o la estructura de algunos ligandos o de la diana, y entonces se usan los llamados *farmacóforos*) hasta el caso más desfavorable caracterizado por la ausencia de cualquier información estructural (donde es necesario llevar a cabo estudios experimentales que nos den idea sobre el tipo de estructuras involucradas en la interacción para poder aplicar algún método teórico).

En este capítulo revisaremos los fundamentos de la técnica denominada *docking* y su uso en el cribado virtual de quimiotechas, y repasaremos los tipos principales de interacciones que se producen entre un ligando y su diana y cómo cuantificarlas. De aquí en adelante usaremos indistintamente los términos *diana* y *receptor*, y en algunos casos proteína, si bien al final del capítulo hay dos apartados relacionados con el *docking* proteína-proteína y ligando-ácido nucleico, respectivamente. Terminaremos este estudio con un apartado donde se recogen las conclusiones principales así como las perspectivas futuras de estas técnicas.

## 16.2. Docking proteína-ligando

### 16.2.1. Definición del problema del *docking*

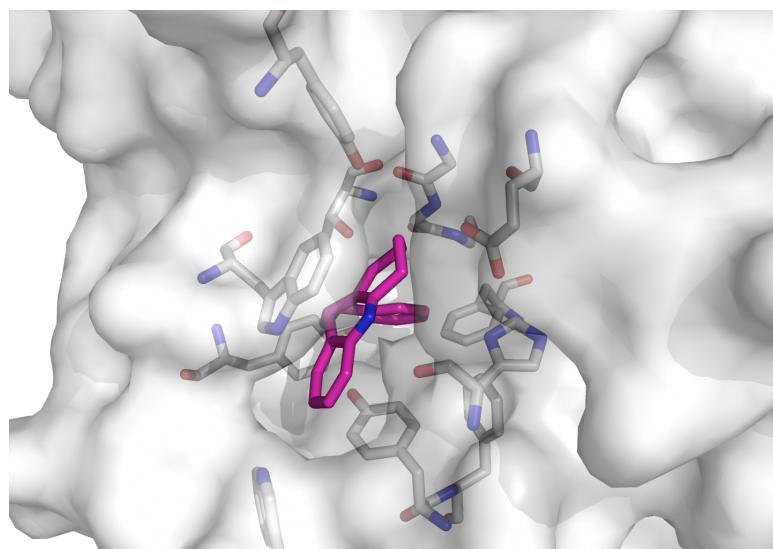
El problema del *docking* puede definirse de forma sencilla de la siguiente manera: dados a) la estructura 3D de una diana de interés, una proteína por ejemplo, y b) la estructura 3D de un ligando, encontrar cuál es la configuración 3D experimental que posee el complejo formado por la proteína y el ligando.

Dado que el *ligando* puede adoptar diversas posiciones dentro del sitio de unión de la proteína, la herramienta de *docking* consta, en primer lugar, de un *método de muestreo* (componente estructural) que enumera todas las configuraciones posibles ligando-receptor (*poses*) y, en segundo lugar, de una función matemática, llamada *función de puntuación o scoring*, que evalúa cómo de buenas son las interacciones existentes entre la proteína y el ligando en cada una de las poses (componente energética).

### 16.2.2. Componente estructural

El método de *docking* pertenece a una clase de técnicas más amplia generalmente conocida como *métodos basados en la estructura*, por lo que se asume un conocimiento previo de la estructura 3D tanto de la diana como del ligando. Por un lado, la estructura de la diana puede obtenerse por medio de diferentes métodos experimentales (principalmente cristalografía de rayos-X y espectroscopía de Resonancia Magnética Nuclear) o teóricos (modelado por homología) y se puede conseguir a través del Protein Data Bank (PDB)[4]. Por otro lado, la estructura experimental de muchos ligandos está accesible a través de la base de datos de Cambridge Structural Database (CSD) [1] o puede construirse de forma fácil con un programa de modelado molecular. Además, existen multitud de estructuras 3D de complejos receptor-ligando en el PDB, siendo ésta una fuente importante de estructuras para probar los algoritmos de *docking*, ya que conocemos a priori cuál es el resultado final.

Desde un punto de vista más técnico, para el *algoritmo de docking* la parte más importante de la estructura de la diana es la zona de unión del ligando, conocida como *bolsillo*, *sitio* o *centro activo*, o *cavidad de unión* (Figura 16.2). La unión del ligando al centro activo produce una modificación, activación o inhibición de la respuesta fisiológica de la diana. El centro activo puede estar localizado en la superficie de la diana o enterrado en su interior. La disposición 3D de las cadenas laterales y del esqueleto de los residuos de la diana en el centro activo determina la especificidad del ligando por esa diana en particular (Figura 16.2). Por último, algunos residuos del centro activo se han conservados a lo largo de la evolución, principalmente aquellos relacionados con la actividad de la diana. Si conocemos dónde está situado el centro activo (por ejemplo a través de la estructura 3D de complejos diana-ligando determinada experimentalmente) podremos guiar al algoritmo de *docking* a una región en particular de la diana, en lugar de buscar en toda su superficie (*docking ciego*).



**Figura 16.2:** Complejo proteína-ligando mostrando detalles del centro activo y las interacciones entre el ligando y las cadenas laterales de los residuos que lo forman.

### 16.2.3. Componente energética

La estabilidad de un complejo molecular, así como la de cualquier molécula individual, se puede cuantificar recurriendo a los principios básicos de la física usando *modelos clásicos*, *mecánica de Newton*, o

*cuánticos*. Dado que los cálculos moleculares basados en mecánica cuántica son muy costosos y sobre todo para sistemas con un elevado número de átomos (como los complejos que nos ocupan), la mecánica clásica es el método de uso común en los cálculos de energías de interacción entre la diana y el ligando.

Cuando se produce la unión entre el ligando y el centro activo de su diana se establecen una serie de interacciones específicas que son las responsables de la estabilidad total del complejo. El éxito de un método de *docking* radica principalmente en un conocimiento profundo y una implementación apropiada de las fuerzas directrices que rigen la unión entre la diana y el ligando. En concreto, a lo largo del protocolo de *docking* se hacen ciertas aproximaciones y se toman ciertas decisiones basadas en la cuantificación de la unión diana-ligando estableciendo un criterio de *bondad de ajuste* en virtud del cual se elegirá la mejor solución como resultado final del *docking*.

Los principales tipos de interacciones moleculares que se consideran fundamentales para entender y racionalizar la unión ligando-diana son las siguientes: *interacciones de van der Waals (vdW)*, *interacciones electrostáticas*, *interacciones por enlace de hidrógeno*, *interacciones con el disolvente*, *interacciones hidrofóbicas* y *contribuciones entrópicas*.

### **Interacciones de tipo *van der Waals* (vdW)**

Cuando dos moléculas se aproximan y van entrando en contacto, las interacciones de vdW dan cuenta de dos tipos de fuerzas diferentes: a) *repulsión*, la cual actúa a corta distancia debido al solapamiento o superposición de las nubes electrónicas de los átomos que se acercan, y b) *atracción*, que se da a larga distancia y es debida a la correlación entre los electrones de los diferentes átomos (*fuerzas de dispersión de London*), y se debe más a la forma (o volumen) que propiamente al contenido electrostático. Ambas fuerzas dependen de la distancia entre los átomos ( $r$ ) por lo que su representación es bastante directa. El modelo más usado es el del *potencial de Lennard-Jones*, donde el término repulsivo depende de  $r^{-12}$  y el atractivo, de  $r^{-6}$ .

### **Interacciones electrostáticas**

Las interacciones electrostáticas están presentes en la mayor parte de los procesos de unión (interacciones carga-carga, enlaces de hidrógeno, apilamiento de nubes  $\pi$  o  $\pi-\pi$  *stacking*, interacciones hidrofóbicas, desolvatación...). Las fuerzas que rigen estas interacciones también se denominan de ajuste de la selectividad, un aspecto clave en el desarrollo de nuevos y más específicos. Sin embargo, su cálculo exacto sigue siendo uno de los mayores retos de nuestros días. La aproximación más simple es el *modelo Culómbico* (el producto de las cargas dividido por la distancia y una función dieléctrica sencilla que simula el apantallamiento ejercido por el disolvente). Otros métodos más complejos, y más costosos computacionalmente, están basados en el *modelo generalizado de Born* (GB) [36] o en la resolución de la *ecuación de Poisson-Boltzmann* (PB) [14].

### **Interacciones por enlace de hidrógeno**

Se trata de una interacción muy selectiva y altamente dependiente de la orientación de sus constituyentes. Se establece entre un *átomo de hidrógeno* unido a un átomo electronegativo, llamado *donador* de enlace de hidrógeno, y otro átomo también electronegativo, llamado *aceptor* de enlace de hidrógeno. La fuerza del enlace depende de la posición relativa de los tres átomos implicados, es decir de las distancias y ángulos que haya entre ellos. Su papel en el reconocimiento molecular es importantísimo.

## **El efecto del disolvente**

Todas las interacciones a tener en cuenta en biología tienen lugar en un entorno acuoso. Cuando las moléculas están aisladas en disolución, están completamente rodeadas de moléculas de agua. Sin embargo, cuando se produce la unión ligando-diana muchas de estas moléculas de agua son desplazadas. Este desplazamiento conlleva un gasto energético que debe ser contrarrestado por las nuevas interacciones formadas. Además, se produce una ganancia de entropía en las moléculas de agua liberadas. Desde un punto de vista teórico, hay dos modelos extremos para tener en cuenta los efectos del disolvente: a) *modelos de disolvente explícito*, donde las moléculas de agua están representadas con detalle atómico, y b) *modelos de disolvente implícito*, donde se construye una función matemática que trata de simular el comportamiento global del disolvente en función de su constante dieléctrica. También es posible considerar *modelos mixtos* en los cuales se tienen en cuenta explícitamente determinadas moléculas y el resto se consideran de manera implícita. En *docking* se suelen emplear modelos de disolvente implícitos ya que son lo suficientemente rápidos como para permitir un gran número de cálculos en un corto espacio de tiempo. Sin embargo, es necesario llegar a un compromiso entre exactitud y velocidad, cualidades que suelen estar inversamente relacionadas. Los métodos más populares son los ya citados GB y PB.

## **Interacciones hidrofóbicas**

Algunos aminoácidos poseen cadenas laterales hidrófobas (leucina, valina, prolina...) al igual que muchos ligandos poseen partes hidrófobas en su estructura. Esto significa que no están bien, energéticamente, en un entorno acuoso. Si dos centros hidrófobos entran en contacto las moléculas de agua de su alrededor son liberadas y esta interacción (*efecto hidrofóbico*) produce una contribución positiva a la estabilización total de la unión.

## **Contribución entrópica**

El concepto de *entropía* está íntimamente relacionado con la idea de orden. Una molécula aislada es libre de desplazarse, rotar y vibrar. Cuando se forma un complejo intermolecular, algunos de estos movimientos se pierden. Como consecuencia de ello, se establece un mayor orden en el sistema lo que lleva asociado una disminución de la entropía. Aparte de la entropía del disolvente, la *entropía del soluto* (o *entropía configuracional*) se suele dividir en dos partes: *conformacional* y *vibracional*. La parte conformacional tiene que ver con la reducción del número de pozos de energía que tanto el ligando como la proteína pueden visitar una vez que ha ocurrido la unión, mientras que la parte vibracional se refiere a los movimientos dentro de un pozo de energía en particular. La entropía es considerada como una propiedad difícil de calcular, y aunque tiene un papel importante en la estimación de la energía libre se suele ignorar.

## **Otras interacciones**

En este apartado se incluyen las interacciones que dan lugar a la formación de *enlaces covalentes* entre el ligando y la diana produciendo inhibidores irreversibles, así como las *mediadas por moléculas de agua* específicas, *iones metálicos* o *átomos de halógenos*. Estas interacciones también son importantes y determinan en algunos casos la correcta predicción del modo de unión del ligando. Sin embargo, rara vez se tienen en cuenta o, si se hace, suele ser a un nivel teórico muy aproximado que está lejos de ser exacto.

#### 16.2.4. *Docking*: consideraciones teóricas

En esta sección se trata el aspecto teórico de las herramientas comúnmente empleadas en *docking*. Sin embargo, antes de profundizar en este tema, es necesario comentar brevemente el modelo físico en el que se basa la unión entre el ligando y su diana.

#### Modelo físico de unión

La magnitud principal que determina la unión entre un ligando y su diana es la *energía libre de unión*, que se define como la diferencia de energías libres entre la correspondiente al complejo ligando-diana y la de sus respectivas especies aisladas (Ecuación 16.1) con las que se encuentra en equilibrio:

$$\Delta G_{\text{unión}} = -RT \ln K \quad (16.1)$$

siendo  $R$  la constante de los gases ideales,  $T$  la temperatura y  $K$  la constante de equilibrio. Esta constante se puede medir experimentalmente y compararla con la estimada por la Ecuación 16.1.

Como en cualquier otro equilibrio químico, la propiedad clave del sistema es el *potencial químico* (de donde deriva la energía libre de unión), que puede estimarse a través de la termodinámica estadística por medio de la función de partición. Una expresión común obtenida en términos de la energía potencial y el efecto del disolvente se muestra en la Ecuación 16.2 [10]:

$$\Delta G_{\text{unión}} = \langle U_{PL} \rangle - \langle U_P \rangle - \langle U_L \rangle + \langle W_{PL} \rangle - \langle W_P \rangle - \langle W_L \rangle - T\Delta S_{\text{conf}} \quad (16.2)$$

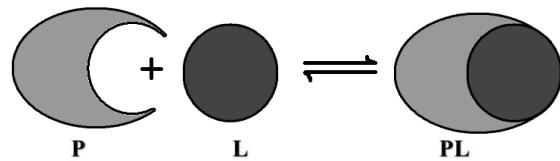
donde  $PL$  se refiere al complejo,  $P$  es la diana y  $L$ , el ligando.  $U$  es la energía potencial y  $W$  representa el efecto del disolvente. Todas estas magnitudes son energías promedio tipo Boltzmann, tal como indica el símbolo “ $\langle \rangle$ ”. El último término se refiere al cambio de entropía configuracional, donde normalmente se considera una sola configuración que representa al estado unido.

Actualmente hay tres modelos que representan la unión proteína-ligando. El primero de ellos data de 1894 y fue postulado por Emil Fischer [9]. Se trata del bien conocido modelo de *llave-cerradura* (*lock-and-key*): solo la llave correcta puede encajar en su cerradura. Se trata de una aproximación muy rígida, ya que no se consideran adaptaciones mutuas entre el ligando y la diana. Una aproximación más flexible, conocida como modelo de *acoplamiento inducido* (*induced fit*), fue posteriormente propuesta por Daniel Koshland en 1958 [24]. El acoplamiento inducido considera que la flexibilidad intrínseca de la diana se traduce en una reorganización de su centro activo para acomodar a los ligandos entrantes. Por último, el modelo de *selección conformacional* (*conformational selection*) (1999) [3], postula que es el ligando el que selecciona, de entre un conjunto de conformaciones accesibles de la diana, la más apropiada para su unión. En la Figura 16.3 puede verse una representación gráfica de los tres modelos.

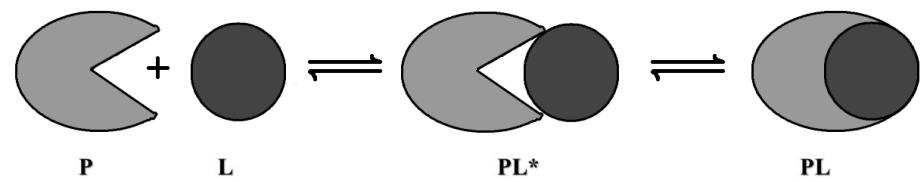
#### La función de puntuación o scoring

Una *función de puntuación*, o *scoring*, es una ecuación matemática que nos da un valor con el que cuantificar la fortaleza con la que un ligando se une a su diana. El paisaje energético de esta unión suele ser extremadamente complejo, con un gran número de *valles* (mínimos) y *montañas* (máximos) (Figura 16.3C). Se espera de la función de *scoring* que sea capaz de recorrer este paisaje y localizar los mínimos, ya que estos se corresponden con situaciones (configuraciones ligando-diana) plausibles en las que se tiene un complejo estable (Figura 16.3A y Figura 16.3B), siendo alguna de ellas similar a la

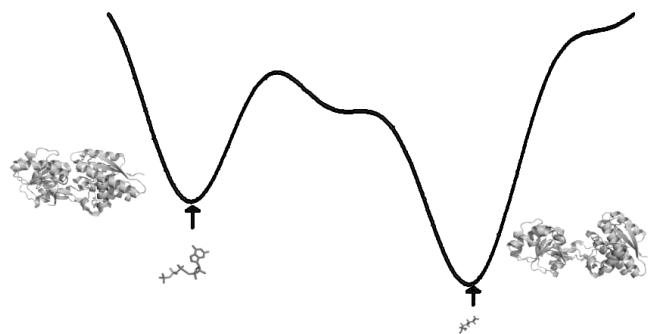
a) Modelo “Lock and Key”



b) Modelo “Induced Fit”



c) Modelo “Conformational selection”



**Figura 16.3:** Representación gráfica de los tres modelos de unión más comunes.

experimental. Además, debe de hacerlo lo suficientemente rápido como para evaluar el elevado número de posibles configuraciones que se generan en los estudios de interacción ligando-diana, y seleccionar, de todos los mínimos posibles, aquel que corresponde a la estructura experimental como la mejor de las soluciones. Por tanto, una función de *scoring* debería ser tanto computacionalmente eficiente como fiable.

Las funciones de *scoring* se utilizan en distintas etapas del proceso de descubrimiento de nuevos fármacos, desde la identificación de un *hit* en cribado virtual hasta su optimización a *lead*, para evaluar las distintas poses en *docking*, para la identificación de ligandos de alta afinidad y para predecir afinidades de unión.

Hay tres tipos principales de funciones de *scoring* que se diferencian entre sí por los datos usados en su derivación: empíricos, basados en el conocimiento y basados en campos de fuerzas.

Las *funciones de scoring basadas en datos empíricos* son aquellas obtenidas a través de un análisis de regresión multilineal entre medidas experimentales de actividad y una serie de propiedades consideradas como fundamentales para que se produzca la unión, como interacciones por enlace de hidrógeno, interacciones iónicas, contactos polares y no polares. La contribución de cada una de estas interacciones a la energía de unión total viene ponderada por un coeficiente que se obtiene usando un conjunto de prueba (*training set*) de complejos ligando-diana. Es por ello que estas funciones son de limitada aplicación más allá del conjunto de prueba (Ecuación 16.3):

$$\Delta G_{\text{unión}} = \Delta G_0 + \Delta G_{hb}f(hb) + \Delta G_{ionic}f(ionic) + \Delta G_{lipo}f(lipo) + \Delta G_{rot}f(rot) \quad (16.3)$$

donde cada  $f(int)$  es una función que representa a un tipo de interacción (*int*), y  $\Delta G_{int}$  es el coeficiente obtenido de la ecuación de regresión y que da peso a la contribución relativa de cada tipo de interacción, siendo  $\Delta G_0$  una constante.

Las *funciones de scoring basadas en el conocimiento* hacen uso de las bases de datos de estructuras 3D para buscar qué tipos de interacción suceden más comúnmente entre ligandos y dianas y con qué frecuencia. Las frecuencias se convierten en energía libre usando la *fórmula inversa de Boltzmann* (Ecuación 16.4) y definiendo un *estado de referencia* que corresponde a aquél en el que no existe tal interacción. Este último punto es precisamente la principal debilidad de estos métodos, al no haber una manera única, ni sencilla, de definir el estado de referencia:

$$\Delta G_{\text{unión}} = \sum_{ij} A_{ij}(r) = -kT \sum_{ij} \ln \left( \frac{g_{ij}(r)}{g(r)} \right) \quad (16.4)$$

donde  $A_{ij}(r)$  es una función que describe las interacciones entre los átomos tipo  $i$  y  $j$  (suma total de todos los átomos ligando-diana como una función de la distancia  $r$ ),  $g_{ij}(r)$  es la probabilidad de que los átomos  $i$  y  $j$  estén a una distancia  $r$ ,  $g(r)$  es la probabilidad del estado de referencia,  $k$  es la constante de Boltzmann y  $T$  la temperatura. Estas funciones también se conocen como *potenciales de fuerza media o potenciales estadísticos*.

Las *funciones de scoring basadas en campos de fuerzas* descomponen la energía de unión ligando-diana en la suma de una serie de términos de interacción individuales, tales como vdW, electrostático, enlace de hidrógeno, etc... que en su definición utilizan parámetros de mecánica molecular. Una función típica basada en un campo de fuerzas se muestra en la Ecuación 16.5, donde solo se han considerado las

interacciones de van der Waals y electrostáticas.

$$\Delta G_{\text{unión}} = \sum_{ij} \left[ \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6} + \frac{q_i q_j}{\varepsilon r_{ij}} \right] \quad (16.5)$$

$A_{ij}$  y  $B_{ij}$  son los parámetros de vdW que dependen del tipo de átomo asignado a  $i$  y  $j$ ,  $r_{ij}$  es la distancia entre el  $i$ -ésimo átomo de la diana y el  $j$ -ésimo átomo del ligando,  $q_i$  y  $q_j$  son las cargas parciales de los átomos  $i$  y  $j$  respectivamente, y  $\varepsilon$  es la constante dieléctrica del solvente.

### 16.2.5. El proceso de docking

Para tener en cuenta el número tan elevado de grados de libertad que se manejan en *docking*, se han desarrollado diversas metodologías, que de manera general se pueden englobar en tres aproximaciones: *docking rígido*, *docking con proteína rígida y ligando flexible* y *docking flexible*.

#### Docking rígido

En la aproximación de *docking rígido* se considera que tanto el ligando como la diana son componentes rígidos, sin grados de libertad internos. Esto reduce la complejidad del problema a rotaciones y traslaciones del ligando en relación a la diana. Sin embargo, esta aproximación es demasiado simplista, ya que ambas especies son entidades de naturaleza flexible. No obstante, es un método muy usado como primera alternativa en programas de *docking* nuevos.

#### Docking de diana rígida y ligando flexible

Asumiendo que la aproximación de llave-cerradura es aceptable como modelo de unión ligando-diana, la flexibilidad de la proteína se puede omitir. De esta manera la *búsqueda conformacional del ligando* se convierte en la parte más importante del problema a resolver. En muchos casos la exploración sistemática no es siempre posible, debido al efecto de la explosión combinatoria que supone la enumeración de todas las posibles rotaciones de cada ángulo de torsional en el ligando, y es por tanto necesario recurrir a otras aproximaciones. Hay multitud de métodos o algoritmos implementados en los programas de *docking* aunque se pueden clasificar, de forma general, en tres categorías principales: construcción incremental, pre-cálculo de las conformaciones del ligando o su generación *in situ*. La *construcción incremental* implica el análisis conformacional *on the fly* (al vuelo) dentro de las limitaciones del sitio de unión, por división del ligando en fragmentos y su posterior unión de manera secuencial (éste es el método usado en los programas FlexX [31] o Surflex [18]). En el método de *conformaciones del ligando pre-calculadas*, las conformaciones del ligando son generadas antes de la operación de *docking* y guardadas en una base de datos para su uso posterior (así funcionan los programas CRDOCK [6] y GLIDE [32]). Por último, cuando las *conformaciones* son *generadas in situ*, la totalidad del ligando se adapta de forma continua al sitio activo de la diana. En este último caso, las técnicas más empleadas son: a) *complementariedad de forma*, basada en la evaluación de la concordancia entre la conformación del ligando y el sitio activo de la diana en términos geométricos (es el caso del programa DOCK [8]); b) *algoritmos genéticos*, basados en la teoría de la evolución de Darwin o en la teoría de la herencia de Lamarck (como los empleados en los programas GOLD [41] o AutoDock [11]); c) *algoritmo de Monte Carlo*, generación aleatoria de grupos de rotación, traslación y orientación de los ligandos y evaluándolos después con una función de scoring (como en el programa LigandFit [40]); d) *búsqueda tabú*, basada

en la generación de poses de forma aleatoria llevando una lista de los sitios o conformaciones que ya han sido probadas (es el caso del programa PRO LEADS [28]); y e) *algoritmos bio-inspirados*, basados en la inteligencia de los enjambres o en las estrategias de las colonias de hormigas (como el programa PLANTS [23]).

### Docking flexible

Con respecto a la diana, y siempre que se incluya la flexibilidad del ligando, los métodos se clasifican en función del grado de flexibilidad que esta incorpore. Una primera aproximación es la conocida como *soft docking*, donde se realiza una relajación de los potenciales de interacción, que trae como consecuencia una expansión en las dimensiones del centro activo, lo que simulan el efecto del ajuste o acoplamiento inducido (esta aproximación se usa en los programas Glide y GOLD). El siguiente paso es el uso, a través de un *algoritmo de Monte Carlo*, de una colección de rotámeros para probar los cambios en las cadenas laterales de la diana, como se hace en los programas GOLD, Glide, AutoDock, FlexX e ICM [37]. Si se dispone de varias estructuras de la misma diana, puede usarse el *esquema del complejo relajado*, que consiste en realizar diferentes experimentos de *docking* de forma individual y hacer luego una promedio con los resultados. Este proceso también se puede emplear con estructuras generadas por simulaciones de dinámica molecular (Capítulo 17) o análisis de modos normales (Capítulo 18). Ambas técnicas se han empleado también para tener en cuenta la flexibilidad total de la diana, mientras que al mismo tiempo se hace el *docking* del ligando. Los programas AutoDock, ICM, Glide o GOLD permiten realizar este tipo de *docking* usando diferentes conformaciones para la diana.

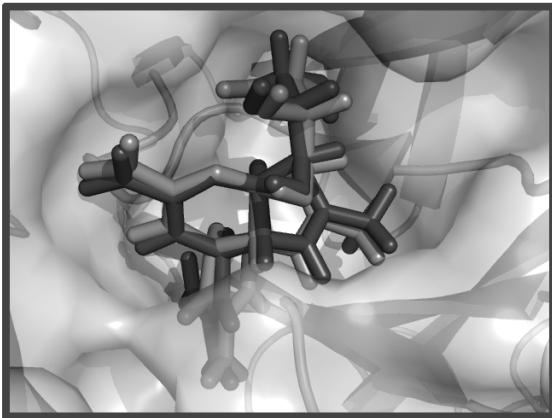
### La etapa de re-scoring

Es bien conocido que la clasificación de las poses de *docking* atendiendo a las funciones de *scoring* no garantiza siempre que la mejor solución de la lista sea la correcta. Es por ello aconsejable considerar una descripción más detallada del proceso de unión incluyendo funciones de *scoring* más precisas y complejas en etapas posteriores a las aproximaciones iniciales. Algunos métodos usan una aproximación basada en *factores de ponderación*, de manera que a la puntuación original es escalada por un factor que depende de aspectos geométricos o propiedades *drug-like*.

#### 16.2.6. El problema de docking: cómo evaluar la validez de los resultados

La evaluación y ordenación de las múltiples soluciones predichas por el algoritmo de búsqueda son los aspectos más críticos de los protocolos de *docking*. La función de *scoring* debería representar de la forma más adecuada posible la termodinámica de la unión ligando-diana para ser capaz de diferenciar el verdadero modo de unión entre todos los demás.

Al menos son necesarias dos cosas para probar la precisión de un nuevo programa de *docking* y su función (o funciones) de *scoring*: un conjunto de complejos con estructura 3D conocida y una medida para cuantificar cómo se parecen los resultados de *docking* a las estructuras experimentales. Aquí, el uso de la *desviación cuadrática media*, o *RMSD* de sus siglas en inglés *root-mean-square deviation*, es el más extendido (Figura 16.4). Con esto nos referimos a la parte estructural del *docking*. Además, si se tienen datos de afinidad/actividad experimental del ligando por la diana, se pueden evaluar cómo de bien las funciones de *scoring* reproducen estos valores. Esta es la parte energética del *docking*.



Negro = Estructura Obtenida Experimentalmente

Gris = Estructura Predicha por Docking



$$RMSD = \sqrt{\frac{\sum_{i=1}^n (x_{1,i} - x_{2,i})^2}{n}} = 0.66$$

$x_{1,i}$  = {número de átomo “i”, estructura experimental }

$x_{2,i}$  = {número de átomo “i”, estructura predicha }

$n$  = {número total de átomos}

**Figura 16.4:** Representación gráfica del RMSD.

## Evaluación estructural

En un estudio de *docking* típico, y con el fin de ver si nuestro algoritmo es adecuado, dado un conjunto de complejos ligando-diana, los ligandos se extraen de sus complejos y se hace *docking* de cada uno de ellos en su propia diana. Después, se calcula el RMSD para cada pose usando como referencia la estructura experimental. Si el valor del RMSD de la pose elegida como la mejor de las soluciones está por debajo de 1.0 Å se considera que el resultado es aceptable, aunque muchos autores aumentan este valor de corte hasta 1.5 Å o incluso hasta 2.0 Å (Figura 16.4). Por encima de este valor límite los resultados se consideran incorrectos. Otro parámetro para probar la eficacia del método de *docking* es el porcentaje de acierto o éxito, definido como el porcentaje de estructuras predichas con valores de RMSD por debajo de 2.0 Å. El valor medio para la mayoría de los programas de *docking* está en torno al 70-75 %, aunque pueden encontrarse porcentajes mayores dependiendo del programa y del conjunto de complejos ligando-diana utilizado en su evaluación.

## Evaluación energética

Esta es, con diferencia, la evaluación más complicada en los cálculos de *docking*. La función de *scoring* debe ser capaz de distinguir y elegir como mejor solución, de entre todas las posibles, aquella con el menor valor de RMSD con respecto a la estructura experimental. Generalmente se asume que el problema de muestreo (encontrar la pose correcta) está más o menos resuelto, pero la baja correlación encontrada entre los valores de energía calculados con la función de *scoring* y el RMSD indican lo contrario.

Tratar de predecir valores de afinidad/actividad es algo incluso más exigente. Un estudio bastante extensivo empleando diferentes dianas y combinaciones de programas de *docking* y funciones de *scoring* demuestra la falta de una correlación significativa entre valores de afinidad/actividad y las energías libres de unión calculadas. Esto significa que la función de *scoring*, aun siendo capaz de reproducir estructuras experimentales de forma precisa, no es suficientemente buena para la predicción de afinidad/actividad, principalmente debido al hecho de que sacrifica precisión en favor de la velocidad de cálculo. En otras palabras, los principios físicos subyacentes que gobiernan las interacciones ligando-diana no están debidamente implementados. De hecho, la entropía, los efectos del disolvente y la flexibilidad de la proteína raramente se tienen en cuenta o se usan a un nivel teórico muy bajo, a pesar del aumento en el poder de computación. Sin embargo, en un futuro cercano se esperan avances significativos en esta área.

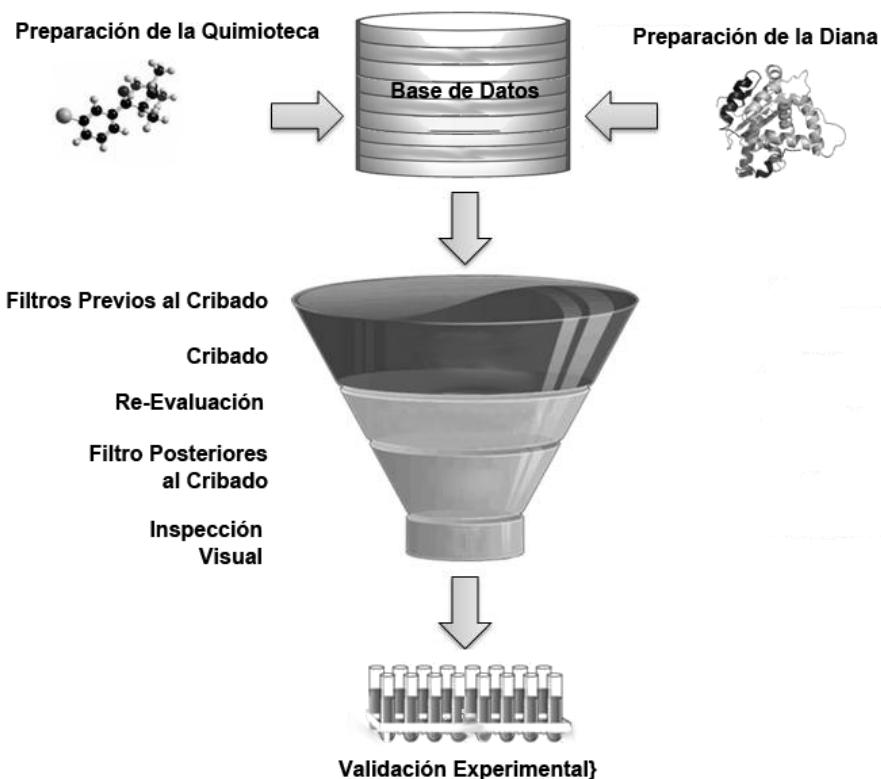
## 16.3. Cribado virtual

El objetivo principal del *cribado virtual* (VS, siglas que en inglés hacen referencia a *Virtual Screening*) es diferenciar, de entre un conjunto de pequeñas moléculas o ligandos (*quimioteca*), aquellas que teóricamente pueden encajar en una cavidad para bloquear/activar la función de una determinada diana (*ligandos verdaderos*) de las que no (*falsos ligandos*) [34]. Se trata de una alternativa teórica al método experimental conocido como *High-Throughput Screening* (HTS) o *cribado farmacológico de alto rendimiento*, pero con un coste y una necesidad de recursos mucho más reducidos.

Dado que los métodos teóricos más fiables son a su vez lo más costosos computacionalmente, y que el número de moléculas con el que estamos tratando es bastante elevado (del orden de millones), el protocolo de VS se configura como una serie de filtros sucesivos donde la complejidad del filtro aumenta a medida que el número de moléculas va disminuyendo. Normalmente se representa como un embudo

con la parte ancha hacia arriba y la estrecha hacia abajo, lo que da una idea de la reducción en el número de moléculas a medida que se avanza en el protocolo (Figura 16.5).

El VS está ganando aceptación dentro del campo del diseño de fármacos y cada vez contribuye con más moléculas como nuevos candidatos a fármaco. Sin embargo, esta técnica aún no está en un grado de desarrollo que podamos considerar maduro. Aun así, se están produciendo avances muy prometedores en la metodología que ya están produciendo buenos resultados.



**Figura 16.5:** Representación de un protocolo general de cribado virtual.

### 16.3.1. Posibles escenarios para el VS

Dependiendo de la información estructural de la que dispongamos, el VS se puede dividir en dos grandes grupos: el *VS basado en la estructura* (SBVS, siglas que en inglés corresponden a *Structure-Based Virtual Screening*), cuando la estructura 3D de la diana es conocida, y b) el *VS basado en el ligando* (LBVS, siglas que en inglés corresponden a *Ligand-Based Virtual Screening*), cuando lo que se conoce es la estructura de un grupo de ligandos activos (o no) frente a la diana de interés. En el primer caso la técnica más utilizada es *docking*, mientras que el segundo caso la estrategia dominante está basada en seleccionar motivos comunes entre las moléculas activas (o las inactivas) para definir lo que se denomina un *farmacóforo*. El farmacóforo se emplea entonces como un molde o plantilla sobre la que se realizan cálculos de semejanza molecular con los compuestos de la quimioteca con el fin de recuperar aquellos que más se parezcan a la plantilla.

En comparación, el número de aplicaciones publicadas donde se aplica el SBVS es aproximadamente el

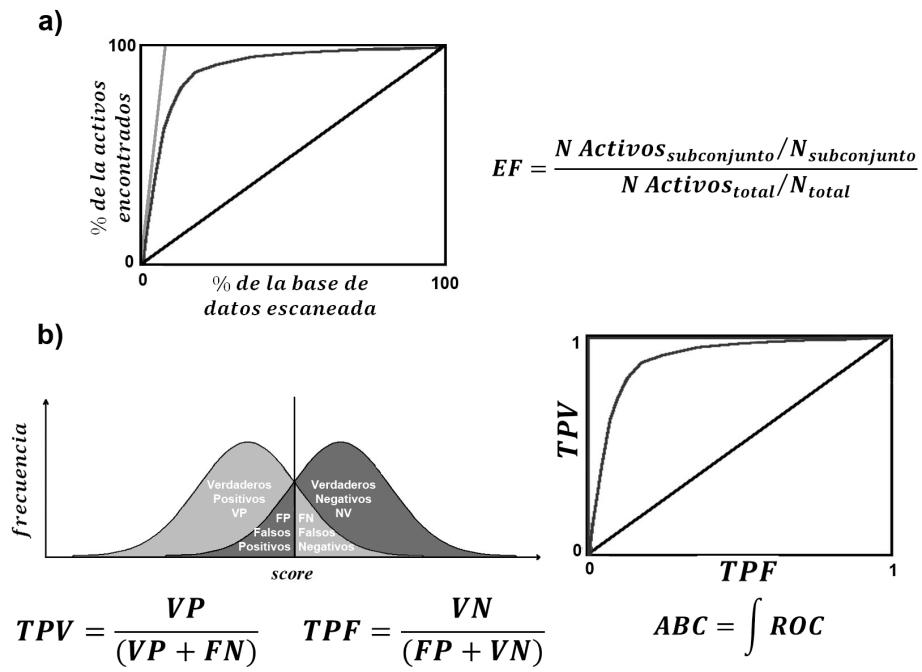
triple al correspondiente a LBVS, aunque este último ha demostrado ser más eficiente en la identificación de nuevas moléculas que el SBVS. Sin embargo, quizás el mejor enfoque sería combinar, en la medida de lo posible, las dos técnicas, de manera que se pueda aprovechar toda la información disponible relativa al problema que se está tratando. Esta estrategia debería sin duda aumentar nuestras probabilidades de éxito. De hecho, en muchos de los estudios de VS en los cuales se han combinado ambos métodos, los candidatos seleccionados han resultado ser los mejores en términos de actividad. En la Figura 16.5 se describe un protocolo general de VS.

### 16.3.2. Estudios de VS retrospectivos y prospectivos

La palabra *retrospectivo* hace aquí alusión a aquellos estudios de VS donde los resultados ya se conocen de antemano, y por lo tanto son muy útiles en la validación de nuevos protocolos. Por lo general, el conjunto de estudio consiste en unas cuantas moléculas cuya actividad frente a la diana de interés ya ha sido confirmada, y una serie de señuelos (*decoys*) que se suponen sin actividad frente a la misma diana. Como se ha mencionado antes, el experimento consiste en ver si el método elegido para realizar el VS es capaz de distinguir entre ambos grupos de moléculas. La elección de los señuelos es un punto muy delicado y en gran medida el factor determinante de que los resultados sean de alguna manera significativos y el método pueda aplicarse de manera prospectiva (ver más adelante) en estudios reales de búsqueda de nuevos fármacos. En particular, los señuelos deben de ser estructuralmente distintos a los compuestos activos pero con las mismas propiedades físico-químicas. De esta manera se evita un posible sesgo de que las moléculas sean seleccionadas simplemente porque se parezcan a las activas. Una fuente bien establecida de conjuntos de datos para estudios de VS es DUD [15] (acrónimo del inglés *Directory of Useful Decoys*) y su versión mejorada DUD-E [29] (la E corresponde a *Enhanced*).

Al igual que en docking, para hacer una valoración retrospectiva de un protocolo de VS además de un conjunto adecuado de datos (activos + señuelos) se necesita una medida para evaluar la eficiencia del método. En general, una medida fiable debería: a) ser independiente de variables extensivas (es decir, aquellas que dependan del número de moléculas activas, señuelos, o incluso de las dianas); b) ser suficientemente robusta; c) plantear un modo directo de evaluar el error; d) no contener ningún parámetro libre; y e) ser interpretable y fácilmente entendible. Las medidas más comunes son el *factor de enriquecimiento* (EF, acrónimo que en inglés corresponden a *Enrichment Factor*) y el *área bajo la curva* (AUC, acrónimo que en inglés corresponden a *Area Under the Curve*) de una representación ROC (acrónimo de *Receiver Operating Characteristic*). El EF mide, para un porcentaje determinado del conjunto de estudio, la relación entre la cantidad de compuestos activos recuperados y la cantidad que de estos se habrían seleccionado si se hubiera hecho al azar. El AUC de una curva ROC representa la fracción de veces que una molécula activa seleccionada al azar poseerá una mayor puntuación que una inactiva también seleccionada al azar. En la Figura 16.6 se muestra un ejemplo de cada una de las curvas así como las fórmulas necesarias para su cálculo.

Por otro lado, en los *estudios prospectivos* de VS no tenemos un conocimiento previo sobre el tipo de compuestos que podrían unirse o no a una diana en particular. Su objetivo es por tanto intentar encontrar candidatos mediante el cribado de quimiotecas. Estos estudios son la prueba final de la valía de cualquier protocolo de VS. La técnica claramente dominante es el *docking*, pero debido a su carga computacional, se usan filtros basados, por ejemplo, en puntos farmacofóricos obtenidos de la estructura de la diana, con el objetivo de reducir el tamaño de la quimioteca. Es necesario señalar de nuevo que en un experimento de VS basado en *docking* la función de *scoring* se evalúa no sólo por su habilidad en diferenciar entre ligandos verdaderos y falsos (compuestos activos e inactivos) a través de la asignación correcta de puntuaciones a cada una de las poses de docking, sino también por su habilidad para recuperar tantos quimiotipos (diferentes fragmentos o bloques químicos) como sea posible con el fin de



**Figura 16.6:** Curvas EF y ROC con sus ecuaciones.

lograr una mayor diversidad.

### 16.3.3. Realización de estudios de VS: herramientas e infraestructuras

La cantidad de datos que hay que manejar en los estudios de VS es enorme. Por lo tanto, el método tradicional de trabajar con una diana, unos pocos ligandos, un programa de *docking* y varios ficheros es inapropiado. El desafío del VS, al menos de una manera genérica, es cómo cribar quimiotebas que contienen millones de compuestos en un periodo de tiempo razonable con un cierto grado de confianza. Hoy en día es factible realizar LBVS en ordenadores personales, ya que es posible comparar varios millones de moléculas por CPU al día sin mayor problema. Sin embargo, para protocolos más elaborados es necesario recurrir a la *computación de alto rendimiento* (HPC, acrónimo del inglés *High Performance Computing*), a la *computación en grid* (*grid computing*), o a la *computación en la nube* (*cloud computing*).

En HPC la aproximación más directa es procesar los ligandos usando un esquema en paralelo donde la quimioteca se divide en  $N$  particiones iguales correspondientes al número de CPUs disponibles. *Grid computing* hace referencia a un conjunto heterogéneo de dispositivos de cálculo conectados entre sí a través de la red y que pertenecen y son administradas por distintas instituciones. La *computación voluntaria* (VC, acrónimo del inglés *Volunteer Computing*) es un caso extremo de computación grid. La diferencia entre computación grid y HPC estriba principalmente en que la primera tiende a ser más heterogénea y a estar geográficamente más dispersa que la segunda, aprovechando la gran cantidad de ordenadores conectados a través de Internet y proporcionando un poder de cálculo virtualmente infinito, mucho más allá de cualquier centro de supercomputación. Algunos de los proyectos de VC más

conocidos son SETI@home<sup>1</sup> y FOLDING@home<sup>2</sup>. También hay ejemplos donde se usa *docking* y VS como el proyecto Screensaver-Lifesaver (proyecto donde se emplearon más de 1.5 millones de PCs y se cribaron del orden de 3500 millones de compuestos frente a dianas contra el cáncer)<sup>3</sup>, Docking@Home (enfocado en la búsqueda de nuevos fármacos contra el virus del sida)<sup>4</sup>, WISDOM (para encontrar nuevos inhibidores contra la malaria) [20] o Ibercivis (para encontrar nuevos inhibidores contra el cáncer o la enfermedad de Alzheimer)<sup>5</sup>. Por último, cloud computing es una tecnología emergente que se refiere a la provisión de recursos computacionales bajo demanda por medio de una red de ordenadores, y la principal ventaja es que libera a los usuarios de la dependencia de cierto hardware y software.

Además de estas infraestructuras, hay una tendencia natural hacia la automatización de las tareas relacionadas con VS proporcionando una *Interfaz Gráfica de Usuario* (GUI, acrónimo del inglés *Graphic User Interface*) que facilite la definición de los diferentes pasos del protocolo: preparación de ligando y diana, *docking* y visualización de los resultados. La GUI da acceso a las principales capacidades implementadas en el programa de *docking* de una forma sencilla, y la mayoría de estos programas poseen ya una GUI (como BDT [39] y DOVIS [19] para AutoDock). Otra alternativa consiste en desarrollar *conectores* (*plugins* en inglés, un componente de software que añade cierta funcionalidad a una aplicación) que pueden implementarse en una interfaz tipo PyMOL<sup>6</sup> y parece ser la tendencia actual dada la cantidad de estos que han visto la luz últimamente (AutoDock/Vina [33], AMBER/AutoDock/SLIDE [26]). Un paso más allá son las llamadas plataformas de VS, es decir, sistemas más sofisticados que integran multitud de piezas que hacen posible de manera sencilla configurar protocolos de VS más complejos (Pipeline Pilot [13], DVSDMS [42], SOMA [25] y VSDMIP [5]). Una plataforma de este tipo debe hacer frente a la gran cantidad de datos de la forma más eficiente posible, lo que implica incluir un motor de base de datos por debajo de cada operación de VS. Aunque este tema parece ser de conocimiento e interés general, muy pocas plataformas la incluyen. Por último, otro ejemplo interesante es DOCK Blaster [17], donde se puede realizar un estudio completo de VS usando como entrada solamente el código PDB de la diana deseada, con todas las herramientas necesarias implementadas en una plataforma vía web.

## 16.4. Docking proteína-proteína

Al igual que en el estudio de las interacciones entre ligandos y sus dianas, entender la manera en la que se producen las interacciones entre proteínas aporta información sobre si esa unión está relacionada con su función y cómo manejarla con fines terapéuticos.

El punto de partida para estudiar el modo de unión entre dos proteínas mediante herramientas de *docking* es el conocimiento de las correspondientes estructuras tridimensionales de ambos sistemas, procedentes de la cristalograffía de rayos-X, la espectroscopía de RMN, la criomicroscopía electrónica o bien modeladas de forma teórica.

Los diferentes algoritmos que se emplean en los estudios de *docking* proteína-proteína tienen como objetivo obtener una lista ordenada de todas las posibles *soluciones* (*poses*), entre las cuales y al igual que en el *docking* ligando-diana, debe de haber una lo más semejante posible a la estructura nativa (experimental) del complejo. Obviamente, en la mayoría de los casos, la estructura del complejo formado entre las proteínas a estudiar es desconocida y es ahí donde juega un papel importante cómo evaluamos

<sup>1</sup>Proyecto SETI@home project. <http://setiathome.berkeley.edu>

<sup>2</sup>Proyecto FOLDING@home. <http://folding.stanford.edu>

<sup>3</sup>Proyecto Screensaver-Lifesaver. <http://www.chem.ox.ac.uk/curecancer.html>

<sup>4</sup>Proyecto Docking@Home. <http://docking.cis.udel.edu>

<sup>5</sup>Proyecto Ibercivis. <http://www.ibercivis.es>

<sup>6</sup>The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC. <http://www.pymol.org>

la calidad de las diferentes poses de *docking*. Esto se suele llevar a cabo a través de una función de *puntuación (scoring)*, la cual nos ayuda a producir una *priorización (ranking)* de las soluciones.

La fuerza que conduce a la unión entre proteínas se corresponde con el cambio de energía libre asociado a dicho proceso, lo que a su vez depende de las propiedades físico-químicas y estructurales de las proteínas implicadas tanto de una manera global (superficie polar/apolar) como de una manera más local a través de la interacción entre determinados residuos que pueden ser claves para la estabilidad del complejo que se forma.

La predicción de la unión entre dos proteínas representa un problema de mayor complejidad en comparación con el que hemos visto anteriormente sobre interacciones proteína-ligando, ya que el número de grados de libertad en este caso es mucho mayor. Tratar de reproducir de forma detallada las interacciones que se producen en cada una de las posibles poses usando métodos como dinámica molecular puede convertirse en algo totalmente inabordable con los medios computacionales actuales. En cualquier caso, si se conocen las zonas de unión en la superficie de las proteínas, el número de posibilidades del problema se reduce drásticamente, aumentando así las posibilidades de éxito. Para añadir un punto extra a la complejidad del problema, hay que tener en cuenta que las proteínas no son en absoluto estáticas (tal y como se pone de manifiesto en diferentes capítulos de este libro), y que las estructuras de una proteína antes y después de unirse pueden ser significativamente diferentes. Esto implica que para predecir de manera precisa dicha unión, en muchos casos debemos tener en cuenta la flexibilidad de las estructuras. Este problema se ha abordado desde diferentes puntos de vista y nos puede ayudar a entender las capacidades y limitaciones de los diferentes algoritmos de *docking* que vamos a tratar a continuación.

#### 16.4.1. El proceso de docking

Existen muchos métodos computacionales capaces de generar cientos de posibles poses entre las dos proteínas sometidas a estudio. El primer paso suele consistir en una búsqueda sistemática de las posibles geometrías de unión, seguido de una evaluación a través de una función de puntuación, para finalizar con una etapa de refinado de las estructuras más prometedoras, es decir, de aquellas que hayan obtenido una mejor puntuación. Este proceso se puede repetir varias veces, dependiendo del protocolo utilizado por los diferentes programas de *docking*.

##### Docking rígido

Está demostrado que un porcentaje importante de las proteínas estudiadas hasta nuestros días presentan movimientos relativamente pequeños una vez que se produce su unión a otras. Por tanto, en estos casos podemos considerar el *docking rígido* como una buena aproximación. Como veremos más adelante, es importante refinar las soluciones más prometedoras para tener en cuenta estos movimientos (por pequeños que sean) y así reproducir las interacciones nativas presentes en el complejo con un mayor acierto.

Dentro de las estrategias más comunes para realizar *docking rígido* entre proteínas están los métodos de *indexado geométrico*, donde la superficie de la proteína se reduce a una serie de descriptores y se buscan las partes de ambas proteínas que puedan encajar entre sí [30], o los métodos basados en *correlaciones entre las Transformadas Rápidas de Fourier* (TRF), que localizan de forma muy eficiente los solapamientos entre las superficies de las proteínas [21].

La aproximación basada en el método de *indexado geométrico* se usó originalmente como una técnica de visualización computacional, para ajustar uno o más conjuntos de datos. En este tipo de algorit-

mos se reduce cada proteína a un conjunto de triángulos que se almacenan en tablas indexadas, las cuales permiten la búsqueda de triángulos de manera rápida. Dado que estos triángulos representan puntos en la superficie con ciertas propiedades geométricas (concavidad/convexidad) y físico-químicas (hidrofobicidad/hidrofilicidad) podemos buscar triángulos coincidentes con propiedades geométricas o físico-químicas complementarias. Mediante este proceso, podemos evaluar las diferentes poses generadas a través de rotaciones y traslaciones de una de las proteínas (considerada como ligando) alrededor de la otra (considerada como receptor).

Desde otro punto de vista, en la *aproximación basada en TRF*, cada proteína se representa como una malla cúbica donde a cada punto de la malla se le asigna un identificador dependiendo de si pertenece al interior de la proteína, a su superficie o al exterior. Utilizando métodos geométricos se pueden superponer las superficies de las mallas del receptor y del ligando y calcular la bondad del ajuste. El problema de esta aproximación es de nuevo la capacidad de cálculo de los ordenadores disponibles hoy en día, por lo que se han desarrollado aproximaciones más eficientes basadas en el teorema de correlación de Fourier. La *transformación discreta de Fourier* de las mallas correspondientes al receptor sólo se calculan una vez, y una vez para cada orientación del ligando. El problema de tener que pre-calcular todas las rotaciones deseadas del ligando en el espacio de coordenadas cartesianas para después transformarlo a sus equivalentes en el espacio de Fourier, se puede evitar correlacionando bases de funciones polares esféricas que representen la forma de la superficie de la proteína.

## Docking flexible

Si las proteínas que forman el complejo experimentan cambios conformacionales apreciables una vez que este se ha formado, es difícil que obtengamos alguna solución parecida a la nativa usando *docking* rígido incluso con un refinado posterior de las mejores soluciones. Predecir los cambios de conformación de las proteínas es una tarea costosa en términos computacionales, así como compleja desde el punto de vista de la descripción física de la unión, y por ello, hasta el momento, no se ha conseguido desarrollar una estrategia directa que nos permita confiar en las soluciones obtenidas. En cualquier caso, ya que este es un problema de gran interés para entender cómo funciona la maquinaria celular, la comunidad científica está desarrollando múltiples estrategias que, en algunos casos, ya se ha conseguido aplicar con éxito. Alguna de ellas se revisa a continuación.

Si sabemos que el cambio conformacional es grande sólo en una de las proteínas implicadas en la formación del complejo, podemos intentar reproducir sus conformaciones mediante técnicas de dinámica molecular [12] (Capítulo 17), modos normales (Capítulo 18) o incluso RMN (Sección 7.7), generando así un conjunto de estructuras iniciales. En algunos casos encontraremos que la estructura nativa de esta proteína está dentro del conjunto de estructuras pre-generadas, por lo que es razonable pasar a la aproximación de *docking* rígido. Todas las poses que se obtengan serán evaluadas seleccionando aquellas que resulten con las mejores puntuaciones. Al usar esta aproximación debemos tener en cuenta que se puede producir un número significativo de falsos positivos ya que en algunos casos podemos encontrar poses con una buena puntuación debido a un buen ajuste de las superficies de interacción, pero que no sean similares a la estructura nativa.

En otros casos podemos tener ciertas evidencias sobre la localización del sitio de unión entre ambas proteínas o incluso sobre algunas de las interacciones que se establecen cuando se forma el complejo. De esta manera podemos reducir de forma importante el número de poses posibles. Aquí es más factible el uso de *docking* y dinámica molecular de forma simultánea. En este modelo es posible incluir la flexibilidad de toda la proteína o de una cierta región durante el *docking*, por lo que, en principio, es un método parecido al escenario real en el que se produce la formación de complejos entre proteínas.

#### 16.4.2. Clasificación y post-procesado de las soluciones

Una función ideal de evaluación de poses de *docking* debería ser capaz de reconocer contactos similares a los que presenta la estructura nativa, y a la vez, de diferenciarlos de aquellos que no lo son. Las *funciones de evaluación* pueden estar basadas en *campos de fuerza*, en las que diferentes contribuciones energéticas se encuentran ponderadas, o pueden desarrollar *potenciales estadísticos* basados en la información estructural que se encuentra en las bases de datos que contienen las estructuras experimentales de los complejos. Normalmente, un único *descriptor* (como la complementariedad de superficie) o un *término energético* (vdW o electrostático) no suele ser capaz de distinguir entre poses nativas y no nativas, por lo que normalmente se utiliza una combinación de varios términos.

Las *funciones de energía basadas en campos de fuerza* son semejantes a las descritas anteriormente para *docking* proteína-ligando (Sección 16.2), aunque algunos de sus parámetros pueden estar ajustados a tipos de átomo específicos pertenecientes a los aminoácidos que forman parte de la estructura de las proteínas [2]. Las *funciones empíricas*, al estar basadas en un modelo de regresión estadística entre la actividad experimental y ciertas propiedades de los modos de unión nativos, se entrena específicamente con un conjunto determinado de complejos proteína-proteína, por lo que su aplicabilidad es reducida [7]. Finalmente, las funciones de energía desarrolladas como *potenciales estadísticos* se basan en el análisis de la frecuencia con la que se observan ciertas interacciones entre determinados residuos en las estructuras experimentales de los complejos proteína-proteína. Este análisis estadístico se puede llevar a cabo tanto a nivel de contactos entre residuos, o de forma más detallada, contactos entre átomos. Basándonos en esta información, podemos construir una función de evaluación basada en el conocimiento (*knowledge-based*) donde se evalúa cada pose comparando sus contactos con la frecuencia con que estos se producen en la base de datos de estructuras [16].

Las poses obtenidas por un algoritmo de *docking* requieren siempre de un *refinado* posterior mediante el ajuste fino de las posiciones atómicas, lo que normalmente implica la reorientación de las cadenas laterales de los residuos, así como ciertos movimientos en las asas que conectan otros elementos de estructura secundaria. El éxito del refinado de las soluciones depende de que el conjunto de estructuras seleccionadas contenga alguna solución lo suficientemente parecida a la estructura nativa. Por lo tanto, en la clasificación de soluciones no sólo se necesita reconocer y preseleccionar modos de unión cercanos a la estructura nativa, sino que a la vez tiene que tolerar ciertas imprecisiones en la interfaz de contacto proteína-proteína para dar la oportunidad al refinado de reconstruir adecuadamente las interacciones nativas.

Con anterioridad al refinado de las soluciones se suelen aplicar *algoritmos de agrupamiento (clustering)* para reducir el número de candidatos. Estos algoritmos producen subconjuntos de soluciones de los que se escoge como representante de cada subconjunto aquella pose que tiene la mejor puntuación.

El *refinado de soluciones* puede llevarse a cabo mediante técnicas de *minimización de la energía*, donde la descripción de la estructura del complejo está basada en un campo de fuerzas de mecánica molecular (ver ). Estas minimizaciones suelen converger a mínimos de energía locales cercanos a la posición inicial donde se producen reorientaciones de las cadenas laterales de ciertos residuos, pequeños movimientos que eliminan choques entre átomos, o que permiten adoptar posiciones idóneas para establecer redes de enlaces de hidrógeno. Otra alternativa es el uso de la *dinámica molecular* mediante la cual es posible lograr mayores cambios conformacionales en comparación con las técnicas basadas en la minimización de la energía. De este modo podemos reorientar la solución de *docking* a mínimos de energía cercanos a la posición inicial pero que mejoren sustancialmente las interacciones entre las superficies de contacto. Para reproducir de forma realista la interacción, en la mayoría de los casos es necesaria la *incorporación de moléculas de agua e iones* durante la simulación, ya que es normal que se encuentren mediando ciertas interacciones entre las proteínas. Por supuesto el uso de la dinámica molecular para el refinado

incrementa notablemente el coste computacional, por lo que debemos tener en cuenta el número de soluciones que queremos refinar respecto al coste computacional que podemos permitirnos. Finalmente, la aplicación de *modos normales* en el refinado de soluciones de *docking* se aplica también con cierto éxito, si bien su uso está menos extendido [27] (ver Capítulo 18).

Por último, una vez que hemos obtenido el conjunto de soluciones refinadas, es habitual realizar una nueva evaluación, ya que tras el refinado muchas poses habrán mejorado sus energías de interacción y por tanto, el *ranking* habrá cambiado con respecto a los resultados iniciales del *docking*.

## 16.5. Docking proteína-ácido nucleico

Aunque en los pasados 25 años se han producido grandes avances en los métodos de *docking* proteína-ligando y proteína-proteína, hasta el momento, existen muy pocos diseñados específicamente para el modelado de las interacciones en las que están involucrados los ácidos nucleicos [22]. La falta de programas de este tipo se debe principalmente a la dificultad del problema. Por un lado, el número de estructuras tridimensionales de moléculas que contienen ácidos nucleicos es muy bajo en comparación con el número de estructuras que se pueden encontrar de otro tipo de moléculas biológicas, dada la dificultad que conlleva su cristalización debido a su gran flexibilidad y su naturaleza de polianiones. Otras dificultades asociadas tienen que ver con la identificación de las superficies de interacción con otras moléculas. Esto ha motivado que el estudio de estructuras proteína-DNA y proteína-RNA se suele hacer a través de los mismos métodos que se usan en *docking* proteína-proteína, haciendo ciertas adaptaciones que no van más allá de incluir los tipos de átomos específicos que nos encontrados en las moléculas de DNA y RNA. En este sentido, el número de tipos de átomos que nos podemos encontrar en el RNA es considerablemente mayor que en el DNA, ya que debido a las modificaciones postraduccionales el RNA puede estar formado por más de 100 tipos de nucleótidos diferentes. Es importante remarcar que las adaptaciones de estos métodos no sólo se refieren a la generación de poses de *docking* [35], sino que puede corresponder también a otras fases del modelado como el refinado de soluciones más prometedoras [27].

Finalmente, y debido a la ambigüedad de las soluciones obtenidas con métodos que no han sido diseñados para ese propósito, existen bases de datos con estructuras tridimensionales de complejos proteína-DNA obtenidas de forma experimental que permiten validar si un método es mejor que otro al generar las poses [38].

## 16.6. Conclusiones generales y perspectivas de futuro

A pesar de su notable éxito, la predicción rápida y eficaz de las interacciones ligando-diana sigue siendo el mayor desafío en *docking*. De la bibliografía de *docking* se deduce que tener en cuenta la flexibilidad de la diana y una buena función de *scoring* siguen siendo las principales preocupaciones, así como, en menor medida, la flexibilidad del ligando.

En lo que se refiere a la flexibilidad del ligando, los mejores resultados en términos de eficacia se obtienen normalmente cuando se usan como entrada múltiples conformaciones, en lugar de sólo una, siempre que éstas representen de forma adecuada el *espacio conformacional del ligando*. Por otro lado, como la flexibilidad de la diana depende en gran medida de la extensión de los cambios conformacionales que suceden durante el proceso de unión, el grado de movimiento permitido puede variar de una diana a otra. Cuando están involucrados grandes movimientos sigue siendo virtualmente imposible tenerlos en

cuenta, principalmente debido al coste computacional que requieren, aunque actualmente se han hecho algunos progresos en este sentido.

Con respecto al *scoring*, aunque se han desarrollado un gran número de funciones en las últimas décadas, el proceso de *docking* sigue siendo muy dependiente de las características específicas del sitio activo y del ligando. Por esta razón, la meta de obtener una función de *scoring* universal puede que sea demasiado ambiciosa, ya que ninguna de ellas podría funcionar bien en todos los casos. No obstante, las funciones de *scoring* hechas a medida para una determinada diana son una alternativa muy prometedora, sobre todo cuando se posee suficiente información acerca de ésta y algunos ligandos.

Como se ha demostrado en varios estudios comparativos, los métodos de *docking* actuales son, por lo general, más capaces de predecir modos de unión y no afinidades. Así, la mejora del rendimiento de la función de *scoring* en la *predicción de afinidad* parece ser una meta más urgente para desarrollos futuros. Esto requiere esfuerzos continuos en el diseño de algoritmos mejorados para las interacciones polares, las energías de solvatación/desolvatación, la entropía configuracional, etc. . . , sin comprometer la eficiencia en términos de tiempo de ejecución.

Si bien es cierto que los métodos de VS son muy populares, siempre se han caracterizado por una tasa muy alta de *falsos positivos* y un rendimiento muy desigual entre dianas consideradas asequibles, y aquellas imposibles de predecir sin tener información previa. Estos problemas, como se ha comentado antes, derivan claramente de la imposibilidad de predecir afinidades de una manera consistente y con un límite de error aceptable.

A día de hoy, a pesar de las continuas mejoras metodológicas en la interpretación de la unión entre moléculas pequeñas y dianas, como son la incorporación rutinaria de la desolvatación o la flexibilidad en los cálculos, el problema sigue vigente y está relacionado con la enorme complejidad del evento que se pretende simular y las aproximaciones que tienen que aplicarse para que el cálculo de la afinidad se pueda realizar en un tiempo adecuado con las tecnologías existentes.

Por otro lado, estos cálculos relacionados con efectos como la flexibilidad de las dianas, la desolvatación y la entropía, que intentan corregir este problema fundamental, conllevan un considerable aumento del tiempo de computación, por lo que seguramente serán también necesarias mejoras no solo metodológicas sino también de implementación incremento de la velocidad de computación, siendo este problema aún más acuciante al aumentar año tras año el número de moléculas disponibles en las *quimiotecas*.

Por lo que respecta al *docking proteína-proteína*, y aunque recientemente se han realizado grandes avances en la incorporación de la flexibilidad, el desarrollo de herramientas capaces de incorporar de forma precisa tanto los movimientos intrínsecos de la proteína como los movimientos asociados a la unión a otras moléculas sigue siendo una tarea pendiente, ya que existe un cuello de botella en el desarrollo de nuevos métodos por la falta de precisión en la estimación de estos movimientos.

También es de esperar que en los próximos años se produzcan mejoras en la evaluación de las *poses de docking*, ya que los métodos actuales presentan problemas asociados con la rigidez. Otras mejoras que aún están lejos de poder alcanzarse debido a su complejidad, pero que son de gran interés para la comunidad científica, tienen que ver con la predicción de posibles plegamientos relacionado con la unión.

Finalmente, en lo relativo al modelado de *interacciones proteína-ácido nucleico*, queda aún mucho trabajo por delante. Es de esperar que se produzcan mejoras notables en un futuro no muy lejano, sobre todo teniendo en cuenta el incremento que se está produciendo en el número de estructuras depositadas en las bases de datos. Por ejemplo, en el año 2000 sólo existían 551 estructuras en el PDB que contienen fragmentos de DNA, mientras que en el año 2012 esta cifra ascendía a 3956. Sin lugar a dudas, el hecho de que las técnicas aquí tratadas vayan por delante en el estudio de las interacciones

entre proteína-ligando y proteína-proteína allanan en buena medida el camino para cuando llegue el momento en el que se disponga de la suficiente información estructural como para abordar directamente el problema del *docking* proteína-ácido nucleico.

## 16.7. Bibliografía

- [1] F. H. Allen. The cambridge structural database: a quarter of a million crystal structures and rising. *Acta crystallographica. Section B, Structural science*, 58(Pt 3 Pt 1):380–8, 2002.
- [2] J. Audie. Development and validation of an empirical free energy function for calculating protein-protein binding free energy surfaces. *Biophysical chemistry*, 139(2-3):84–91, 2009.
- [3] C. Berger, S. Weber-Bornhauser, J. Eggenthaler, J. Hanes, A. Pluckthun, and H. R. Bosshard. Antigen recognition by conformational selection. *FEBS letters*, 450(1-2):149–53, 1999.
- [4] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–42, 2000.
- [5] A. C. Cabrera, R. Gil-Redondo, A. Perona, F. Gago, and A. Morreale. Vsdmip 1.5: an automated structure- and ligand-based virtual screening platform with a pymol graphical user interface. *Journal of computer-aided molecular design*, 25(9):813–24, 2011.
- [6] A. Cortes Cabrera, J. Klett, H. G. Dos Santos, A. Perona, R. Gil-Redondo, S. M. Francis, E. M. Priego, F. Gago, and A. Morreale. Crdock: an ultrafast multipurpose protein-ligand docking tool. *Journal of chemical information and modeling*, 52(8):2300–9, 2012.
- [7] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee. Empirical scoring functions: I. the development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *Journal of computer-aided molecular design*, 11(5):425–45, 1997.
- [8] T. J. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz. Dock 4.0: search strategies for automated molecular docking of flexible molecule databases. *Journal of computer-aided molecular design*, 15(5):411–28, 2001.
- [9] E. Fischer. Synthesen in der zuckergruppe. *Berichte der deutschen chemischen gesellschaft*, 23(2):2114–2141, 1890.
- [10] M. K. Gilson, J. A. Given, B. L. Bush, and J. A. McCammon. The statistical-thermodynamic basis for computation of binding affinities: a critical review. *Biophysical journal*, 72(3):1047–69, 1997.
- [11] D. S. Goodsell, G. M. Morris, and A. J. Olson. Automated docking of flexible ligands: applications of autodock. *Journal of molecular recognition : JMR*, 9(1):1–5, 1996.
- [12] R. Grunberg, J. Leckner, and M. Nilges. Complementarity of structure ensembles in protein-protein binding. *Structure*, 12(12):2125–36, 2004.
- [13] M. Hassan, R. D. Brown, S. Varma-O'brien, and D. Rogers. Cheminformatics analysis and learning in a data pipelining environment. *Molecular diversity*, 10(3):283–99, 2006.
- [14] B. Honig and A. Nicholls. Classical electrostatics in biology and chemistry. *Science*, 268(5214):1144–9, 1995.
- [15] N. Huang, B. K. Shoichet, and J. J. Irwin. Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, 49(23):6789–801, 2006.
- [16] S. Y. Huang and X. Zou. An iterative knowledge-based scoring function for protein-protein recognition. *Proteins*, 72(2):557–79, 2008.
- [17] J. J. Irwin, B. K. Shoichet, M. M. Mysinger, N. Huang, F. Colizzi, P. Wassam, and Y. Cao. Automated docking screens: a feasibility study. *Journal of medicinal chemistry*, 52(18):5712–20, 2009.
- [18] A. N. Jain. Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *Journal of medicinal chemistry*, 46(4):499–511, 2003.
- [19] X. Jiang, K. Kumar, X. Hu, A. Wallqvist, and J. Reifman. Dovis 2.0: an efficient and easy to use parallel virtual screening tool based on autodock 4.0. *Chemistry Central journal*, 2:18, 2008.
- [20] V. Kasam, J. Salzemann, M. Botha, A. Dacosta, G. Degliesposti, R. Isea, D. Kim, A. Maass, C. Kenyon, G. Rastelli, M. Hofmann-Apitius, and V. Breton. Wisdom-ii: screening against multiple targets implicated in malaria using computational grid infrastructures. *Malar J*, 8:88, 2009.
- [21] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. A. Friesem, C. Aflalo, and I. A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–9, 1992.

- [22] R. M. Knegtel, J. Antoon, C. Rullmann, R. Boelens, and R. Kaptein. Monty: a monte carlo approach to protein-dna recognition. *Journal of molecular biology*, 235(1):318–24, 1994.
- [23] O. Korb, T. Stutzle, and T. E. Exner. Empirical scoring functions for advanced protein-ligand docking with plants. *Journal of chemical information and modeling*, 49(1):84–96, 2009.
- [24] D. E. Koshland. Application of a theory of enzyme specificity to protein synthesis. *Proceedings of the National Academy of Sciences of the United States of America*, 44(2):98–104, 1958.
- [25] P. T. Lehtovuori and T. H. Nyronen. Soma—workflow for small molecule property calculations on a multiplatform computing grid. *Journal of chemical information and modeling*, 46(2):620–5, 2006.
- [26] M. A. Lill and M. L. Danielson. Computer-aided drug design platform using pymol. *Journal of computer-aided molecular design*, 25(1):13–9, 2011.
- [27] E. Lindahl and M. Delarue. Refinement of docked protein-ligand and protein-dna structures using low frequency normal mode amplitude optimization. *Nucleic acids research*, 33(14):4496–506, 2005.
- [28] C. W. Murray, C. A. Baxter, and A. D. Frenkel. The sensitivity of the results of molecular docking to induced fit effects: application to thrombin, thermolysin and neuraminidase. *Journal of computer-aided molecular design*, 13(6):547–62, 1999.
- [29] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–94, 2012.
- [30] R. Norel, D. Fischer, H. J. Wolfson, and R. Nussinov. Molecular surface recognition by a computer vision-based technique. *Protein engineering*, 7(1):39–46, 1994.
- [31] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of molecular biology*, 261(3):470–89, 1996.
- [32] M. P. Repasky, M. Shelley, and R. A. Friesner. Flexible ligand docking with glide. *Curr Protoc Bioinformatics*, Chapter 8:Unit 8 12, 2007.
- [33] D. Seeliger and B. L. de Groot. Ligand docking and binding site analysis with pymol and autodock/vina. *Journal of computer-aided molecular design*, 24(5):417–22, 2010.
- [34] B. K. Shoichet. Virtual screening of chemical libraries. *Nature*, 432(7019):862–5, 2004.
- [35] M. J. Sternberg, H. A. Gabb, and R. M. Jackson. Predictive docking of protein-protein and protein-dna complexes. *Current opinion in structural biology*, 8(2):250–6, 1998.
- [36] W. Still, A. Tempczyk, R. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J Am Chem Soc*, 112:6127–6129, 1990.
- [37] M. Totrov and R. Abagyan. Flexible protein-ligand docking by global energy optimization in internal coordinates. *Proteins*, Suppl 1:215–20, 1997.
- [38] M. van Dijk and A. M. Bonvin. A protein-dna docking benchmark. *Nucleic acids research*, 36(14):e88, 2008.
- [39] M. Vaque, A. Arola, C. Aliagas, and G. Pujadas. Bdt: an easy-to-use front-end application for automation of massive docking tasks and complex docking strategies with autodock. *Bioinformatics*, 22(14):1803–4, 2006.
- [40] C. M. Venkatachalam, X. Jiang, T. Oldfield, and M. Waldman. Ligandfit: a novel method for the shape-directed rapid docking of ligands to protein active sites. *Journal of molecular graphics & modelling*, 21(4):289–307, 2003.
- [41] M. L. Verdonk, J. C. Cole, M. J. Hartshorn, C. W. Murray, and R. D. Taylor. Improved protein-ligand docking using gold. *Proteins*, 52(4):609–23, 2003.
- [42] T. Zhou and A. Caflisch. Data management system for distributed virtual screening. *Journal of chemical information and modeling*, 49(1):145–52, 2009.

# Capítulo 17

## Dinámica molecular

*Juan A. Bueren-Calabuig*

### 17.1. Introducción

Para poder comprender con detalle la estructura, la función y los mecanismos de reacción de sistemas biológicos y químicos es necesario emplear métodos teóricos de modelado molecular y de simulación que complementen los resultados obtenidos mediante técnicas experimentales. A pesar de que la cristalografía de rayos X permite estudiar las propiedades físicas y estructurales de las moléculas, la mera visualización de una única estructura molecular no es capaz de resolver problemas más complejos como, por ejemplo, el movimiento de una proteína o su unión a un ligando. En estos casos es necesario acudir a la simulación de tales procesos. *El principal objetivo del modelado molecular aplicado en distintas áreas como la bioquímica o la farmacología, consiste en describir las interacciones biomoleculares en base a las leyes generales de la química y de la física* [15, 19].

Hoy en día, el modelado molecular está íntimamente ligado al uso de computadoras y gráficos interactivos. El espectacular desarrollo de los ordenadores y de las técnicas computacionales ha permitido el uso de modelos teóricos para el estudio de todos aquellos problemas relacionados con la geometría y la energía de las moléculas. De hecho, se podría llegar a modelar con exactitud cualquier sistema biológico con suficiente poder de cálculo y un alto nivel de teoría. Sin embargo, en la práctica, esto no es siempre posible debido a limitaciones tanto de tiempo como de recursos materiales. Por este motivo, es necesario introducir aproximaciones para mantener la viabilidad del experimento aplicando distintos niveles de teoría en función del sistema que va a ser analizado [29]. Como norma general, a medida que aumenta el nivel teórico aumenta la calidad de los resultados obtenidos, pero también se incrementa notablemente el coste computacional.

La descripción más rigurosa de un sistema viene definida por la *mecánica cuántica* (*Quantum Mechanics*, QM) que tiene en cuenta explícitamente los electrones en sus cálculos, haciendo posible el estudio de la estructura, las propiedades que dependen de la distribución electrónica y la reactividad química (formación y ruptura de enlaces). Sin embargo, su aplicabilidad está restringida a sistemas con centenares de átomos como máximo, siendo difícilmente viable para aquellos constituidos por miles de átomos en ausencia de recursos de supercomputación. Por otro lado, la *Mecánica Molecular o clásica* (MM) ignora los movimientos electrónicos y calcula la energía de una molécula o conjunto de moléculas únicamente en función de la disposición de los núcleos atómicos por lo que resulta el nivel de teoría generalmente escogido para estudiar macromoléculas biológicas como el DNA o las proteínas.

La Dinámica Molecular (DM) se basa en los principios de la mecánica clásica y constituye uno de los métodos más utilizados para la simulación de macromoléculas biológicas como el DNA o las proteínas. Ambas técnicas, no obstante, se pueden combinar en aproximaciones de Mecánica Cuántica / Mecánica Molecular (QM/MM). Con este método híbrido una pequeña parte del sistema implicada, por ejemplo, en una reacción química, se estudia mediante QM mientras que el resto de los átomos se considera mediante MM.

## 17.2. Mecánica molecular

Los cálculos de MM se basan en la *aproximación de Born-Oppenheimer* que permite separar los movimientos del núcleo y de los electrones. Se considera que, debido a que la masa del núcleo es muy superior a la de los electrones, éstos pueden adaptarse rápidamente a cualquier cambio en las posiciones de los núcleos. Por lo tanto, la energía de una molécula en su estado basal, puede considerarse como una función de las coordenadas de los núcleos atómicos. Esta función se la denomina *Campo de Fuerzas* o *Force Field*. Los cambios que se producen en la energía potencial de un sistema pueden representarse como una superficie, denominada *superficie de energía potencial* [11]. Uno de los objetivos en modelado molecular es encontrar los *puntos mínimos* en la superficie energética que corresponden a estructuras moleculares optimizadas. También es importante encontrar los “*puntos de silla*” (puntos de pendiente igual a cero en cualquier dirección). Estos puntos se consideran barreras de mínima energía en los caminos que conectan los distintos mínimos y que corresponden a los estados de transición.

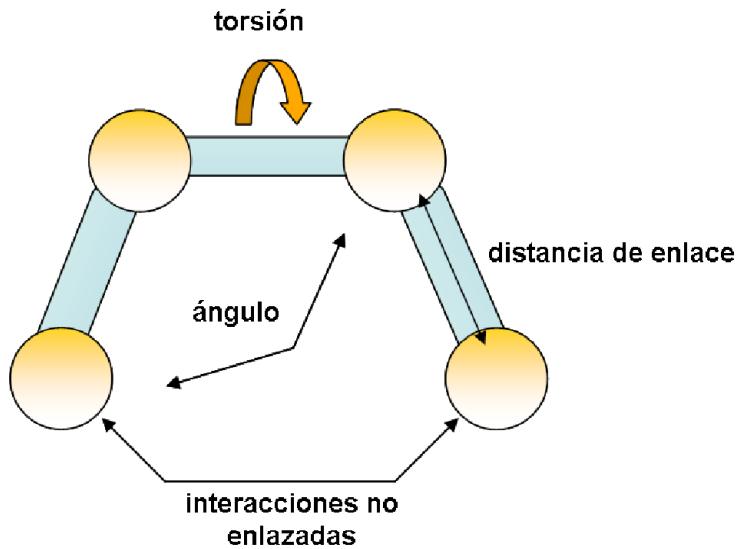
### 17.2.1. El campo de fuerzas

Los primeros campos de fuerzas para simulaciones biomoleculares fueron desarrollados por primera vez hacia 1970 [16, 41, 42, 44]. Desde entonces un gran número de campos de fuerza empíricos para MM han sido desarrollados para la simulación de proteínas, ácidos nucleicos, lípidos y otras moléculas biológicas [40]. Es importante diferenciar entre los programas empleados en simulación y los parámetros de MM desarrollados para ellos dado que en muchas ocasiones los nombres coinciden. Entre los programas más usados para las simulaciones de DM por ordenador de moléculas biológicas destacan AMBER [9], CHARMM [5], GROMOS [38], NAMD [32] y TINKER [34]. Un *campo de fuerzas* está formado por la función de energía potencial y por los parámetros empíricos usados por cada uno de los términos. Una característica importante de un campo de fuerzas es su capacidad de poder ser transferido, es decir, una misma serie de parámetros puede ser empleada para el estudio de distintas moléculas relacionadas entre sí [21]. La mayoría de los campos de fuerzas empleados para sistemas moleculares se pueden definir mediante una *ecuación con dos componentes principales* que describen las interacciones enlazantes y no enlazantes del sistema:

$$E_{total} = \underbrace{E_{enlace} + E_{ángulo} + E_{torsional}}_{\text{Enlazantes}} + \underbrace{E_{vdw} + E_{elec}}_{\text{No enlazantes}} \quad (17.1)$$

Los *términos enlazados* incluyen contribuciones debidas a los enlaces covalentes, ángulos de valencia y ángulos torsionales propios e impropios. Los *términos no enlazados* se definen por un término de atracción-repulsión de tipo Lennard-Jones para las fuerzas de van der Waals y un término Coulombico para las interacciones electrostáticas. En la Figura 17.1 se esquematizan ambos tipos de interacciones.

A continuación se describen los componentes individuales más importantes que forman parte del campo de fuerzas en MM.



**Figura 17.1:** Esquema de las interacciones enlazantes y no enlazantes que se tienen en cuenta en un campo de fuerzas de MM.

### 17.2.2. Términos enlazados

#### Término de enlace

El *término de enlace* (*bond stretching*) se encarga de mantener las longitudes de enlace cercanas a los valores de equilibrio medidos experimentalmente. En términos generales la energía potencial para un enlace covalente se define mediante una *función de Morse* representada en rojo en la Figura 17.2.

Sin embargo, el potencial de Morse no se emplea generalmente en los campos de fuerzas dado que, en los cálculos de MM, raras veces los enlaces se desvían significativamente de sus valores de equilibrio. Por ello se emplean expresiones más simples como la *ley de Hooke* por la cual la energía varía en función del desplazamiento desde la longitud de referencia del enlace  $r_{eq}$ :

$$E_{enlace} = \frac{K_r}{2} (r - r_{eq})^2 \quad (17.2)$$

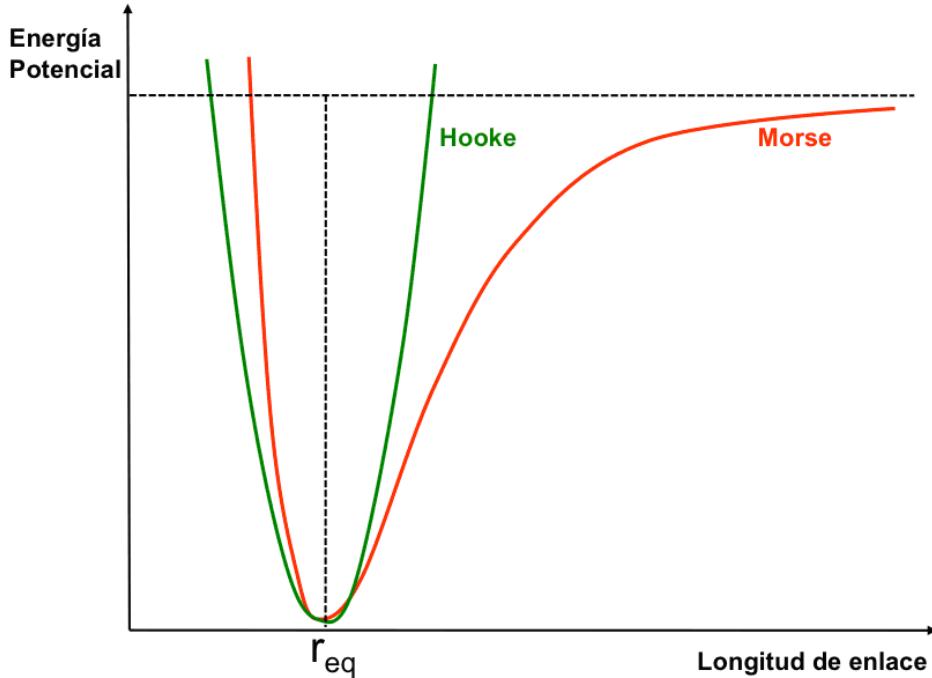
#### Término de ángulo

La *variación de los ángulos* (*angle bending*) con respecto a sus valores de referencia también se representa mediante un *potencial armónico de Hooke*:

$$E_{ángulo} = \frac{K_\theta}{2} (\theta - \theta_{eq})^2 \quad (17.3)$$

#### Término de torsión

El *término de torsión* (*torsional term*) describe la variación de la energía asociada a la rotación alrededor de un enlace B-C dentro de una serie de cuatro átomos A-B-C-D donde A-B, B-C, y C-D están



**Figura 17.2:** Forma del potencial de enlace según la representación de Morse y según una representación armónica.

unidos. Se caracteriza por presentar una periodicidad en el ángulo  $\phi$ : si el enlace rota  $360^{\circ}$  la energía debe volver al mismo valor. Su perfil energético se expresa como una *serie de Fourier*:

$$E_{torsional} = \frac{V_n}{2} [1 + \cos(n\phi - \gamma)] \quad (17.4)$$

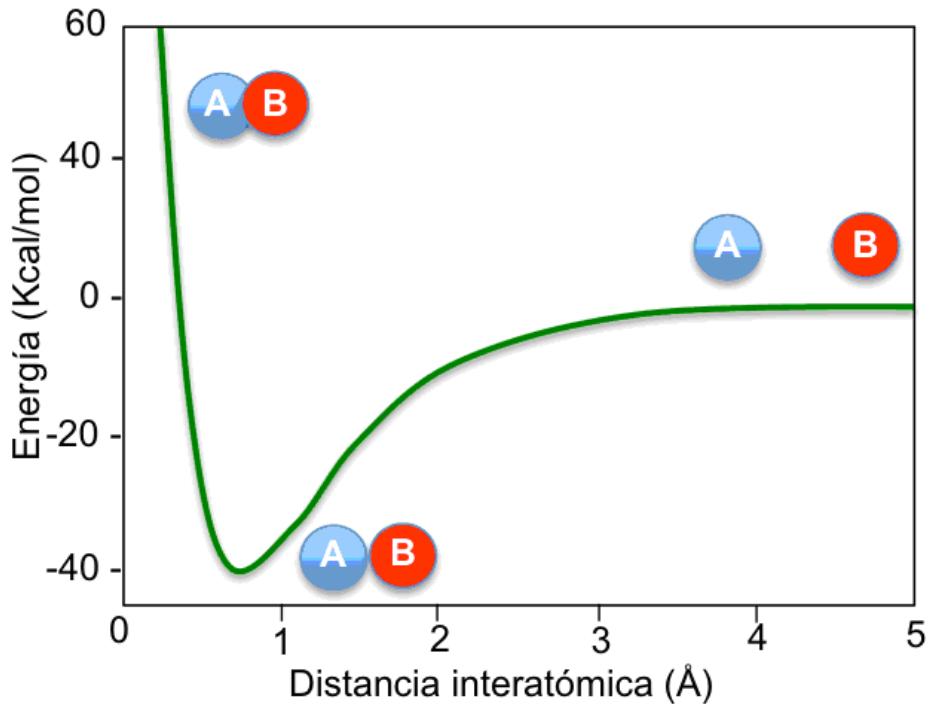
La constante  $V_n$  determina la altura de la barrera de torsión alrededor del enlace B-C,  $n$  describe la multiplicidad (el número de mínimos en la función cuando se rota  $360^{\circ}$ ),  $\phi$  el ángulo de torsión y  $\gamma$  el ángulo de fase (indica en qué punto pasa la torsión por mínimo energético).

### 17.2.3. Términos no enlazados

Las moléculas y los átomos independientes interactúan entre ellos a través de interacciones que no dependen de una relación específica de enlace entre átomos. Estos términos se agrupan en dos grupos: interacciones de van der Waals e interacciones electrostáticas.

#### Interacciones de van der Waals

La *interacción de van der Waals* entre dos átomos se origina a partir de un balance entre fuerzas atractivas y repulsivas. Esta energía de interacción varía en función de la distancia entre ambos átomos como muestra la Figura 17.3. La energía de interacción es cero a una distancia interatómica infinita (e incluso despreciable a distancias relativamente cortas). Al reducirse la distancia, la energía disminuye hasta llegar a un mínimo. Después, la energía crece rápidamente al continuar disminuyendo la distancia.



**Figura 17.3:** Potencial de Lennard-Jones.

Las interacciones de atracción y repulsión entre átomos y moléculas pueden ser calculadas usando *mecánica cuántica*. Sin embargo, ya que en muchos de los sistemas a estudiar existe un gran número de interacciones de van der Waals, se hace necesario emplear una función que las calcule de manera más eficiente. La función más conocida es la *función de Lennard-Jones*:

$$E_{vdw}(r_{AB}) = \frac{a_{AB}}{r_{AB}^{12}} - \frac{b_{AB}}{r_{AB}^6} \quad (17.5)$$

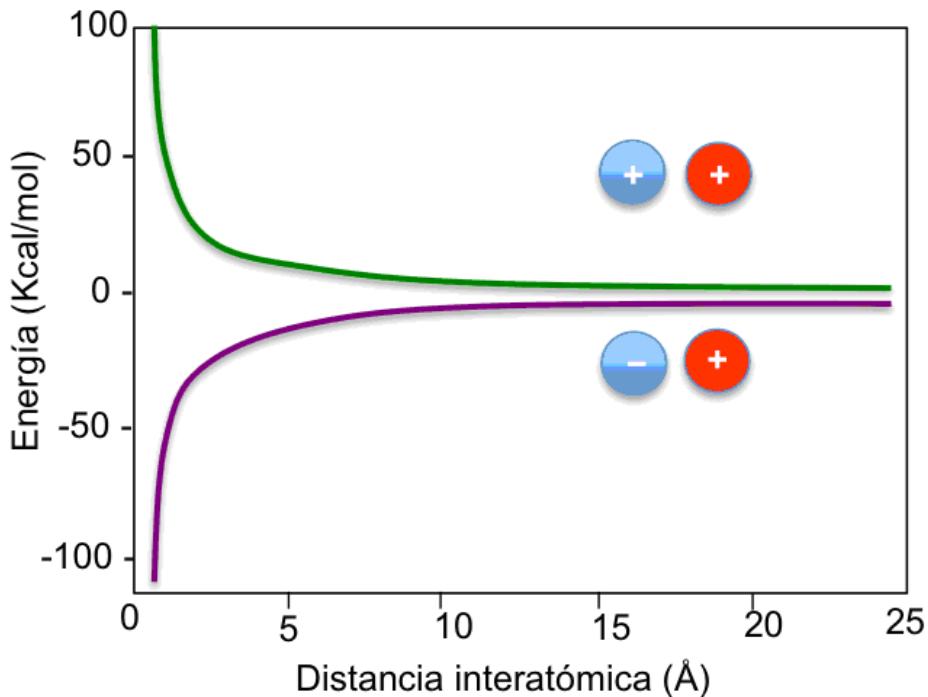
Donde  $a$  y  $b$  son dos constantes específicas del par de átomos A y B.

### Interacciones electrostáticas

La *distribución de la carga* en una molécula se puede representar como una ordenación de las cargas puntuales. Estas cargas reproducen las propiedades electrostáticas de la molécula. En el caso de que las cargas estén centradas en los núcleos, se las denomina *cargas atómicas parciales*. La interacción electrostática de una molécula se calcula, por tanto, como la suma de las interacciones entre pares de cargas puntuales según la *ley de Coulomb*:

$$E_{elec}(r_{AB}) = \frac{q_A q_B}{4\pi\epsilon_0 r_{AB}} \quad (17.6)$$

$q_A$  y  $q_B$  son las cargas puntuales de cada átomo,  $r_{AB}$  la distancia entre ellos y  $\epsilon_0$  la constante dieléctrica del medio que las separa. Si las cargas puntuales de dos átomos son contrarias éstos se atraerán entre sí, pero si las cargas son del mismo signo se repelerán (Figura 17.4).



**Figura 17.4:** Potencial Coulómbico.

#### 17.2.4. Parametrización del campo de fuerzas

Los *campos de fuerzas* contienen un gran número de parámetros, incluso si están diseñados para modelar sistemas pequeños. El objetivo a la hora de desarrollar estos parámetros es encontrar un modelo capaz de aproximarse lo mejor posible a los valores experimentales. Por ello, una fase importante en la *parametrización* consiste justamente obtener estos parámetros de datos experimentales. En MM, esta información proviene de datos estructurales, energéticos o electrónicos. Además, para complementar estos datos experimentales, se recurre a los *cálculos cuánticos ab initio* que son capaces de reproducir resultados experimentales en muchos sistemas. El gran inconveniente del uso de estos métodos para la obtención de parámetros es que se requieren muchos recursos computacionales para asegurarse de que los datos *ab initio* son suficientemente precisos.

#### 17.2.5. Minimización de energía

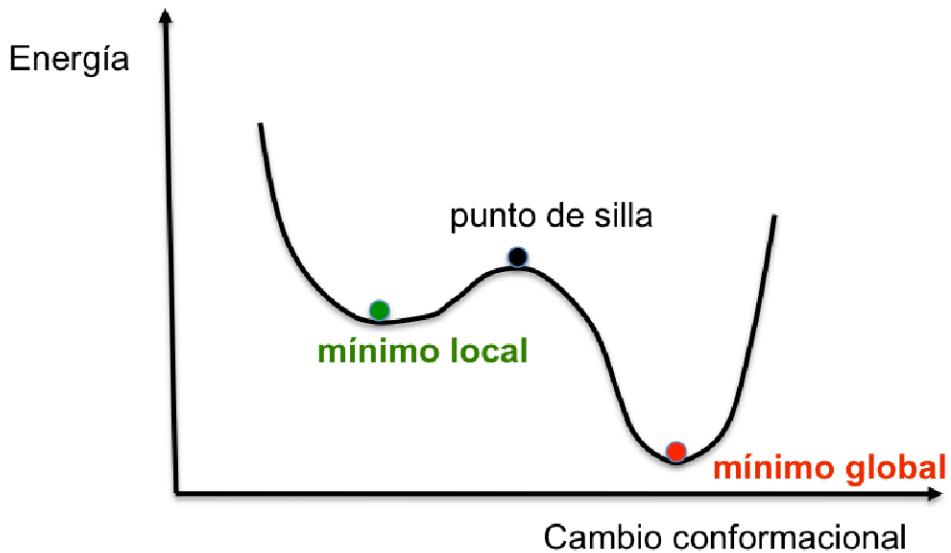
Como se dijo anteriormente, la *superficie de energía potencial* de un sistema viene definida por el modo en el que la energía de las moléculas varía en función de sus coordenadas. En modelado molecular, es necesario estudiar los *puntos mínimos* en la superficie de energía potencial que corresponden a los *estados estables* del sistema. Cualquier cambio en esta configuración produciría un incremento de la energía potencial [21]. En muchos casos puede haber un gran número de mínimos en la superficie energética. Aquel punto con el mínimo energético más bajo se le conoce como *mínimo global* mientras que los demás puntos se los denomina *mínimos locales* (Figura 17.5). En un punto mínimo, la primera derivada de la función de potencial  $f$  con respecto a las coordenadas cartesianas o internas ( $x_i$ ) es igual a cero mientras que las segundas derivadas son positivas (Ecuación 17.7).

El punto energético más elevado en el camino entre dos mínimos se le conoce como *punto de silla* y

corresponde al estado de transición. El proceso que permite identificar las geometrías del sistema que corresponden a los puntos de mínima energía potencial se denomina *algoritmo de minimización*.

Los métodos más empleados para la minimización de energía en modelado molecular son los basados en las derivadas de la función de potencial, especialmente el de “*descenso más pronunciado*” (*steepest descent*) y el de “*gradiente conjugado*” (*conjugate gradient*). Ambos modifican las coordenadas de los átomos mientras desplazan el sistema hacia el del punto de mínima energía. El primero realiza una búsqueda siguiendo la máxima pendiente y se suele emplear cuando la estructura a estudiar está muy alejada del mínimo. Al aproximarse a este punto mínimo, el método de *steepest descent* se vuelve demasiado oscilante y por ello puede sustituirse con el método de gradiente conjugado que evita estas oscilaciones.

$$\frac{\partial f}{\partial x_i} = 0 : \frac{\partial^2 f}{\partial x_i^2} > 0 \quad (17.7)$$



**Figura 17.5:** Esquema de la superficie energética.

Existe un gran número de algoritmos de minimización además de los expuestos anteriormente. Dado que este capítulo, al estar destinado a ofrecer una visión general de las técnicas de simulación, no ofrece los detalles de los distintos algoritmos, se recomienda la consulta de bibliografía más detallada [17, 21].

### 17.3. Simulaciones de dinámica molecular

Los *métodos de minimización* generan configuraciones individuales de mínima energía que, en muchos casos, pueden ser suficientes para predecir ciertas propiedades de un sistema. Sin embargo, estos métodos difícilmente pueden ser aplicados cuando queremos estudiar las propiedades estructurales y termodinámicas de sistemas macromoleculares que contienen numerosos puntos mínimos de energía. Los *métodos de simulación computacional* son capaces de obtener una muestra representativa de las configuraciones de estos sistemas macroscópicos a una determinada temperatura. Existen dos grandes técnicas de simulación computacional: el *método de Monte Carlo* y las *simulaciones de Dinámica Molecular* (DM). Mientras que el primero se basa en analizar la energía de distintas coordenadas generadas

de manera aleatoria, la DM permite generar una trayectoria de puntos que evolucionan con el tiempo siguiendo la segunda ecuación de Newton. Se trata por tanto de un método determinista, es decir, el estado de un punto de la trayectoria permite predecir el estado del siguiente. En este capítulo, se describirán los principios y aplicaciones de las simulaciones de dinámica molecular.

### 17.3.1. Cálculo de las fuerzas

Como se dijo al inicio del capítulo, La DM está basada en los *principios de la mecánica molecular*: los núcleos atómicos son suficientemente pesados como para ser considerados como partículas clásicas cuya dinámica puede ser estudiada mediante la *segunda ecuación de Newton*,  $F = ma$ , cuya forma diferencial se escribe:

$$\frac{d^2y}{dt^2} = \frac{F_{x_i}}{m_i} \quad (17.8)$$

siendo  $F_{x_i}$  la fuerza aplicada a la partícula  $i$  en la dimensión  $x$  y  $t$  el tiempo. A partir de unas coordenadas ( $x_0$ ) y velocidades iniciales ( $v_0$ ) se determina la energía potencial y la energía cinética del sistema, siendo la energía total la suma de ambas:

$$E_{total} = E_{potential(x_0)} + E_{cinética(v_0)} \quad (17.9)$$

La *evolución temporal del sistema* se puede seguir aplicando métodos de integración numérica. De este modo se obtienen pequeñas etapas sucesivas separadas en el tiempo por un intervalo fijo  $\partial t_i$  (tiempo de integración). La fuerza que actúa sobre cada partícula en un instante de tiempo  $t$  se determina mediante la derivada de la energía potencial con respecto a las coordenadas:

$$F = -\frac{\partial E_{potencial}}{\partial x} = 0 \quad (17.10)$$

El valor de la fuerza se asume constante durante cada paso de integración. Sumando sobre cada núcleo las fuerzas de interacción con las demás partículas del sistema se obtiene la fuerza total a partir de la cual podemos determinar las aceleraciones de las partículas ( $\frac{\partial^2y}{\partial t^2} = \frac{F_{x_i}}{m_i}$ ) que se combinan a continuación con las posiciones y velocidades a tiempo  $t$  para calcular las posiciones y velocidades a tiempo  $t + \partial t$ .

### 17.3.2. Integración de las ecuaciones de movimiento

El método más simple para integrar las ecuaciones de movimiento es mediante algoritmos que aplican las *series de Taylor*. Empleando esta aproximación, la posición de una partícula a tiempo  $t + \partial t$  se expresa en función de su posición, velocidad y aceleración:

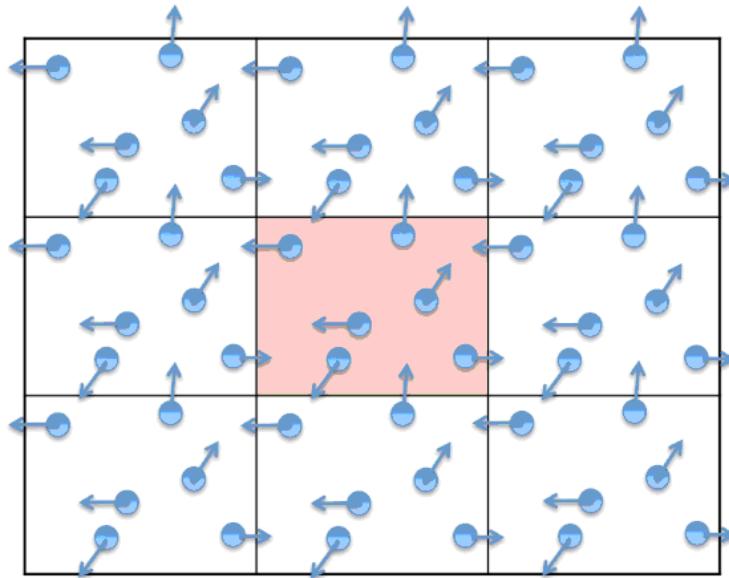
$$x_i(t = \partial t) = x_i(t) + \frac{\partial x_i}{\partial t} + \frac{1}{2} \frac{\partial^2 x_i}{\partial t^2} \quad (17.11)$$

Uno de los métodos más utilizados para integrar las ecuaciones de movimiento en simulaciones de DM es el *algoritmo de Verlet* [37]. Éste usa las posiciones y aceleraciones a tiempo  $t$  y las posiciones en la etapa previa,  $x_i(t - \partial t)$ , para calcular las nuevas posiciones a  $t + \partial t$ ,  $x_i(t + \partial t)$ . El *tiempo de integración*

se selecciona teniendo en cuenta que un tiempo de integración demasiado grande dará lugar a inestabilidades por solapamiento de altas energías pero un tiempo demasiado corto estudiará únicamente una región muy limitada del espacio conformacional. En el caso de biomoléculas como proteínas y DNA, los movimientos más rápidos provienen de las vibraciones de los enlaces entre átomos pesados e hidrógeno (X-H) que tienen lugar en la escala del femtosegundo. Sin embargo, estos movimientos rápidos contribuyen poco al comportamiento global de la molécula y supondrían emplear más recursos computacionales. Para evitarlo, se utiliza el *algoritmo SHAKE* [35] que mantiene constantes las longitudes de los enlaces X-H en sus valores de equilibrio y permite emplear un tiempo de integración mayor (2 fs) reduciendo el coste computacional.

### 17.3.3. Condiciones de límite periódico

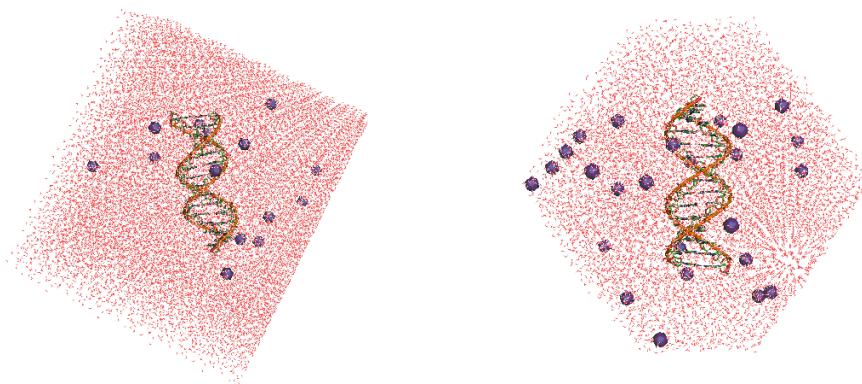
El agua juega un papel esencial en todos los organismos vivos. Por ejemplo, muchas enzimas necesitan moléculas de agua para poder desarrollar sus funciones biológicas [27]. Las simulaciones de DM, por tanto, deben intentar reproducir las *interacciones de un solvente acuoso* con el sistema biológico a estudiar, bien de manera explícita o implícita. Durante una simulación, el sistema biológico incluido en una caja de agua incluye átomos que se localizan en el borde del área de simulación. Para evitar anomalías y asegurar una inmersión completa del soluto en el solvente, se emplean *condiciones de límite periódico* de manera que el sistema se considera rodeado por réplicas iguales en todas las direcciones y el cálculo de la simulación se desarrolla como si el sistema fuera infinito en el espacio (Figura 17.6). La forma y tamaño de las cajas depende de la geometría del sistema, usándose cajas octaédricas y rectangulares para el estudio de ácidos nucleicos y proteínas (Figura 17.7).



**Figura 17.6:** Condiciones de límite periódico en dos dimensiones.

### 17.3.4. Cálculo de las interacciones no enlazantes

El cálculo de los *términos no enlazantes* en una DM es el proceso más costoso computacionalmente. Una aproximación muy utilizada para acelerar este proceso consiste en anular las interacciones entre



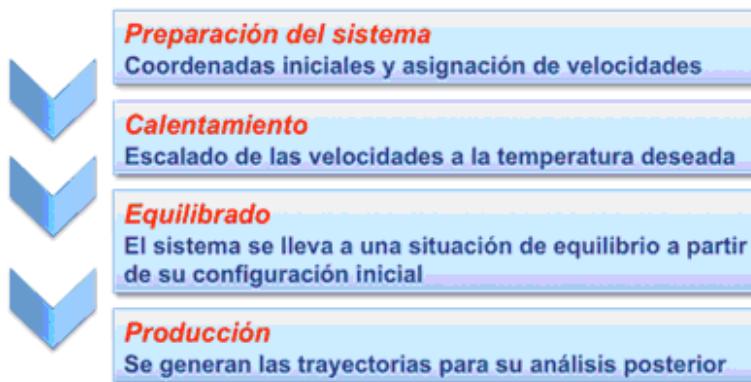
**Figura 17.7:** Caja rectangular y octaedro truncado.

pares de átomos que están separados por una distancia mayor a una distancia determinada o *distancia límite*. Es importante tener en cuenta que la distancia límite debe ser inferior a la distancia entre cualquier punto de la caja hasta cualquiera de sus copias vecinas.

El uso de esta distancia límite, sin embargo, afecta especialmente al término electrostático debido a que esta interacción es de largo alcance. Usando el *sumatorio de Ewald* [14] se consigue computar la interacción electrostática total sumando las interacciones con infinitos sistemas que son réplicas del original.

### 17.3.5. Preparación y ejecución de una DM

En esta sección discutiremos algunas *etapas del desarrollo de las simulaciones de DM*. Un ejemplo de un protocolo para ejecutar simulaciones de DM se observa en la Figura 17.8.



**Figura 17.8:** Protocolo general para la ejecución de DM.

En primer lugar, es necesario establecer una *configuración inicial del sistema* que puede provenir de datos experimentales (cristalografía de rayos X, NMR), de modelos teóricos o de una combinación de ambos. Una vez optimizada la estructura, es necesario *asignar velocidades* iniciales a cada uno de los átomos. Esto se realiza de manera aleatoria a partir de una distribución Maxwell-Boltzmann a una determinada temperatura. Finalmente, es preciso mantener fijas algunas de las condiciones de

simulación: número de partículas (N), volumen (V), temperatura (T), presión (P) o energía total del sistema (E). Al combinarlas se obtienen *simulaciones microcanónicas* (NVE), *isotérmico-isobáricas* (NPT) y *canónicas* (NVT) siendo NPT y NVT las más empleadas en DM de proteínas y ácidos nucleicos.

Una vez que el sistema ha sido preparado y optimizado y las velocidades han sido asignadas, puede comenzar la simulación propiamente dicha. Las simulaciones de DM se componen de dos etapas: una *fase de equilibrado* y una *fase de producción*. El objetivo del *equilibrado* es llevar al sistema a un estado de equilibrio a partir de la configuración inicial. Durante esta fase se monitorizan varios parámetros como la energía potencial, la temperatura y la densidad hasta que se estabilizan. Para conseguir un equilibrado óptimo, en ocasiones se aplican restricciones al sistema, liberándolas a continuación lentamente para permitir su adaptación a las condiciones deseadas. Este proceso suele durar entre 200 y 500 picosegundos aunque en ciertos casos conviene equilibrar el sistema durante varios nanosegundos.

Después de un correcto equilibrado comienza la *producción* en la cual permitimos la evolución del sistema obteniendo una trayectoria que será analizada a continuación. Cuanto mayor sea el tiempo de simulación, se explorará más espacio conformacional y por lo tanto los resultados serán más precisos. En general se estima que el tiempo de simulación debería ser al menos diez veces más largo que la escala temporal del proceso a estudiar. En la práctica esto significa que con los ordenadores actuales sólo se pueden simular procesos que tienen lugar en la escala de unos pocos nanosegundos aunque gracias a la mejora en los algoritmos de paralelización, la capacidad de almacenamiento en discos y el uso de supercomputadores se ha podido pasar de simulaciones de 10 ns a las de 1 microsegundo [29].

#### 17.3.6. Simulaciones de dinámica molecular de macromoléculas biológicas

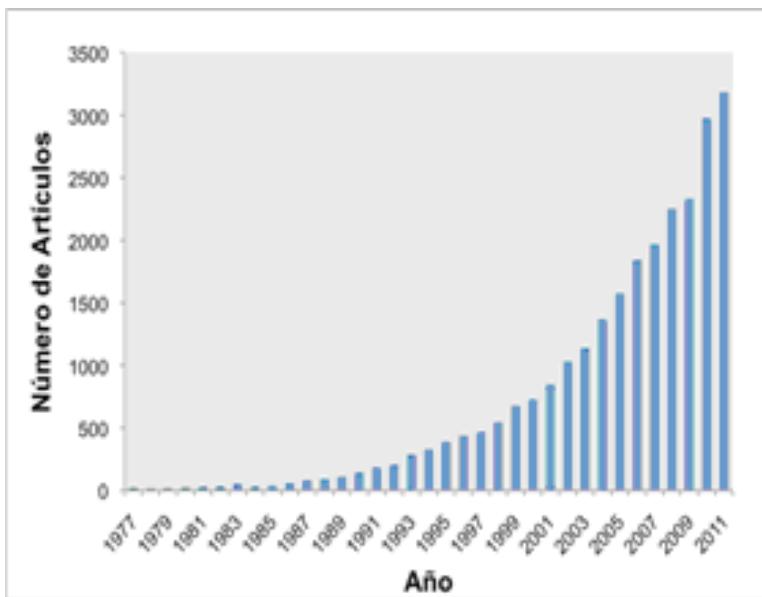
Los primeros métodos de simulación de DM fueron desarrollados por Alder y Wainwright en 1957 empleando un modelo de esferas sólidas en el cual los átomos interactuaban únicamente mediante interacciones perfectas ([2]. En 1964, Rahman realizó la primera simulación con potenciales continuos para reproducir las interacciones atómicas con mayor realismo (Rahman 1964). Ya durante los años 70, las simulaciones de DM se empezaron a aplicar a sistemas más complejos y en 1976 se realizó la primera simulación biomolecular investigando las isomerización del retinal [43] y la primera simulación de DM de una proteína, el inhibidor de la tripsina pancreática bovina (BPTI), por McCammon en 1977 [24]. Esta primera DM tenía una extensión de 9.2 ps e incluía 500 átomos. En la actualidad, gracias al desarrollo de las teorías de campos de fuerza y al incremento en el poder de cálculo, somos capaces de estudiar sistemas que contienen hasta un millón de átomos en la escala del micro y del milisegundo. De hecho, el uso de superordenadores especialmente diseñados para simulaciones de DM como el llamado “*Anton*” permiten realizar investigaciones que alcanzan una escala de tiempo en la cual tienen lugar muchos procesos biológicos [39].

El número de publicaciones relacionadas con la teoría de DM y su aplicación en el estudio sistemas biológicos está creciendo extraordinariamente [1]. Un ejemplo es la evolución desde 1970 hasta la actualidad del número de artículos que investigan las proteínas y la DM incluidos en PubMed (Figura 17.9).

Por supuesto, los experimentos tienen un papel fundamental a la hora de *validar los métodos de simulación* [19]. A la hora de desarrollar las investigaciones es imprescindible una acción interdisciplinar en la cual experimentos teóricos complementen a los experimentales y viceversa. Gracias a esta mutua colaboración se enriquecen las investigaciones y se generan resultados de mayor relevancia [7].

Las simulaciones de DM se emplean en la actualidad para estudiar prácticamente cualquier tipo de macromolécula con un interés biológico o médico (proteínas, ácidos nucleicos, carbohidratos) [4]. Entre las *aplicaciones* más importantes de las simulaciones de dinámica biomolecular destacan el estudio de

los cambios conformacionales (Elber 2005) o simulaciones de plegamiento y desplegamiento de proteínas [12]. La DM es también un método muy apropiado para estudiar los canales iónicos o la membrana de las proteínas cuyo estudio mediante métodos experimentales resulta complejo [3, 36].



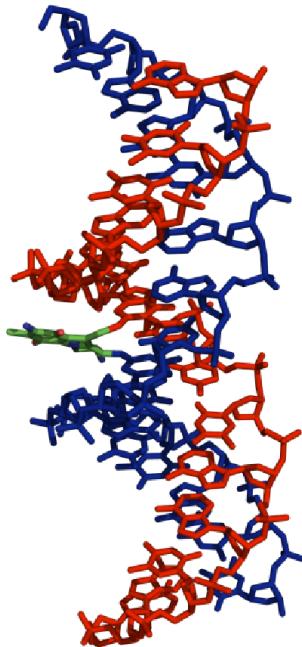
**Figura 17.9:** Número de artículos incluidos en PubMed que resultan de una búsqueda de los términos “Molecular Dynamics & Proteins”.

Otra ventaja de las simulaciones de DM es que permiten *refinar estructuras* obtenidas mediante cristalográfica de rayos X o por NMR [6, 10] (ver Sección 7.7). Además, la simulación de sistemas biomoleculares nos permiten calcular la energía libre de unión de pequeños ligandos a sus moléculas diana [20, 25] o la energía de activación necesaria para las reacciones catalizadas por enzimas [40].

La DM se ha convertido en una herramienta esencial en el estudio del DNA aportando gran información acerca de sus características estructurales y dinámicas. El DNA se caracteriza por ser una molécula muy flexible pudiendo presentar distintas conformaciones [30]. Los métodos experimentales son muchas veces insuficientes para analizar estos cambios conformacionales y por ello se recurre a la simulación para estudiar de la flexibilidad y estructura del DNA. Un ejemplo es el estudio de la desnaturación de una doble hélice de DNA simulado por DM demostrando que se trata de un proceso muy complejo siguiendo distintas rutas [31]. Investigaciones similares fueron también desarrolladas para estudiar el mecanismo de acción de fármacos antitumorales que interaccionan con el DNA. Se demostró mediante DM que los fármacos covalentemente unidos al DNA evitan la fusión completa de la doble hélice inducida por alta temperatura justificando su acción estabilizadora del DNA [7] (Figura 17.10).

Uno de los problemas de la DM como algoritmo de muestreo de la superficie de energía potencial de los sistemas biomoleculares es que la trayectoria puede quedarse estancada explorando únicamente un mínimo local [40]. Distintos métodos han sido desarrollados para superar esta dificultad y ampliar el espacio conformacional a estudiar, por ejemplo, la aplicación de un potencial tipo “umbrella” para forzar el muestreo siguiendo una determinada coordenada de reacción [33] o el acoplamiento de distintas trayectorias permitiendo que intercambien distintos parámetros entre ellas, como por ejemplo la temperatura [28].

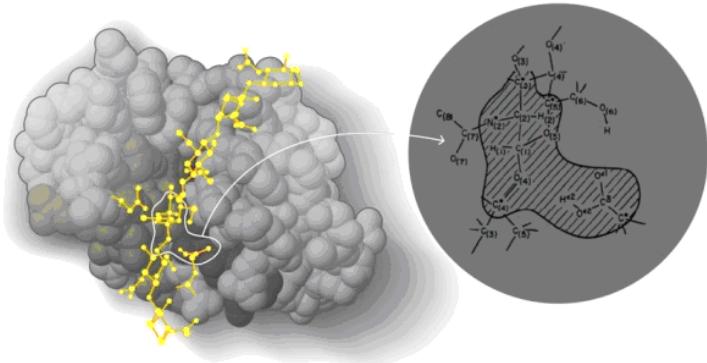
Gracias al uso de técnicas que amplifican el poder atomístico de las simulaciones de DM, se puede extender el margen de problemas biológicos a estudiar mediante modelado molecular. En las simulaciones



**Figura 17.10:** Modelo Molecular obtenido por simulación de DM del entrecruzamiento intercatenario formado entre el fármaco antitumoral Mitomicina C y las dos hebras de una molécula de DNA. Adaptado de Bueren-Calabuig et al. 2012 [8].

de modelos aproximados, llamados de “*grano grueso*” o *coarse grained*, ideados por Levitt y Warshel [22, 23], un grupo de átomos se modela como un único sitio de interacción de manera que, al bajar el detalle atomístico, se consiguen alcanzar escalas de tiempo biológicamente más relevantes para estudiar procesos más lentos y sistemas más grandes [4, 18].

En 2013, Martin Karplus (Universidad de Estrasburgo y Universidad de Harvard), Michael Levitt (Universidad de Standford) y Ariel Warshel (Universidad de Southern California) fueron galardonados con el Premio Nobel de Química “por el desarrollo de modelos multiescala para sistemas químicos complejos”. En los años 70, contribuyeron a desarrollar técnicas computacionales para modelar reacciones químicas o el plegamiento de proteínas. Al aplicar las técnicas de QM/MM, consiguieron estudiar reacciones enzimáticas con un gran nivel de detalle (ver Figura 17.11).



**Figura 17.11:** En 2013, El Premio Nobel de Química reconoció las técnicas de modelado molecular por ordenador que muestran, por ejemplo, cómo la lisozima hidroliza los enlaces glicosídicos (en amarillo). Los autores combinan métodos QM muy precisos estudiar la reacción (derecha), con métodos de MM que describen el resto de la proteína (izquierda) [43].

## 17.4. Métodos híbridos QM/MM

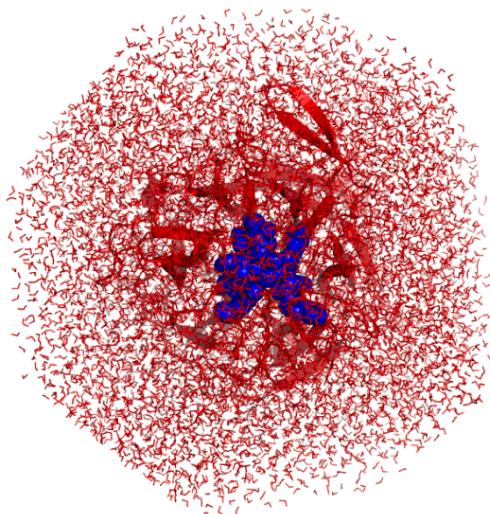
Esta técnica permite combinar potenciales cuánticos y mecánico-moleculares en un *potencial híbrido QM/MM* [13, 21, 26]. En este tipo de aproximación, una pequeña fracción del sistema, por ejemplo, aquella implicada en una reacción química, se analizan mediante una función QM mientras que el potencial de los demás átomos del sistema se examina por cálculos clásicos de MM. Este método combina la simplicidad y velocidad del tratamiento MM con el potencial de la química cuántica que permite el modelado de la formación y rotura de enlaces, así como la inclusión de la polarización electrónica. Al dividir el sistema, somos capaces de realizar cálculos significativamente más grandes que los que hubieran sido posibles únicamente con una aproximación QM pura y al mismo tiempo podemos estudiar reacciones químicas que no son susceptibles de un modelado riguroso mediante MM. En el estudio de un mecanismo de acción enzimático, la región QM correspondería normalmente al sitio activo incluyendo los grupos reactivos de la enzima mientras que la región MM incluiría la parte mayoritaria de la proteína, aquella que no incluye los residuos implicados en la reacción (Figura 17.12).

La energía total  $E_{TOT}$  para este tipo de sistemas se puede escribir de la siguiente forma:

$$E_{TOT} = E_{QM} + E_{MM} + E_{QM/MM} \quad (17.12)$$

donde  $E_{QM}$  y  $E_{MM}$  corresponden a la energía de aquellas partes del sistema tratadas exclusivamente con QM y MM, respectivamente.  $E_{QM/MM}$  es la energía de interacción entre las partes mecánico-cuánticas y mecánico-moleculares.

Se han implementado distintos métodos con el fin de estudiar un sistema mediante QM/MM, los cuales difieren entre sí por el nivel de teoría utilizado para la QM (semiempírico, *ab initio*, enlace de valencia o funcional de densidad), por el modelo de MM y por el modo de representar el disolvente (explícitamente o usando un modelo simplificado). Otra diferencia importante es el modo de tratar la unión entre las regiones QM/MM. En general, es preferible cortar enlaces no polares (como enlaces sencillos C–C) que cortar enlaces insaturados o polares. El método más utilizado incluye simplemente un “*link atom*” o átomo enlazante (normalmente hidrógeno) que asegura la conservación de la valencia.



**Figura 17.12:** Ejemplo de un esquema QM/MM utilizado en el estudio del mecanismo de acción de la trans-Sialidasa de *Trypanosoma cruzi* [33]. El centro activo (en azul) será estudiado por un método QM mientras que los demás aminoácidos de la enzima y el solvente (en rojo) serán tratados mediante MM clásica.

## 17.5. Programas y tutoriales de dinámica molecular

En la Tabla 17.1 se listan algunos de los programas más utilizados para realizar simulaciones de dinámica molecular junto con tutoriales y enlaces a sus páginas web.

Programa/Tutorial	URL
AMBER	<a href="http://ambermd.org">http://ambermd.org</a>
Simulación de un fragmento de DNA	<a href="http://ambermd.org/tutorials/basic/tutorial1">http://ambermd.org/tutorials/basic/tutorial1</a>
Uso de VMD con AMBER	<a href="http://ambermd.org/tutorials/basic/tutorial2">http://ambermd.org/tutorials/basic/tutorial2</a>
Simulación de un fármaco	<a href="http://ambermd.org/tutorials/basic/tutorial4b">http://ambermd.org/tutorials/basic/tutorial4b</a>
Análisis de trayectorias con Ptraj	<a href="http://ambermd.org/tutorials/basic/tutorial5">http://ambermd.org/tutorials/basic/tutorial5</a>
Dinámica Molecular dirigida	<a href="http://ambermd.org/tutorials/advanced/tutorial10">http://ambermd.org/tutorials/advanced/tutorial10</a>
QM/MM	<a href="http://ambermd.org/tutorials/advanced/tutorial2">http://ambermd.org/tutorials/advanced/tutorial2</a>
CHARMM	<a href="http://www.charmm.org">http://www.charmm.org</a>
GROMOS	<a href="http://www.gromos.net">http://www.gromos.net</a>
NAMD	<a href="http://www.ks.uiuc.edu/Research/namd">http://www.ks.uiuc.edu/Research/namd</a>
TINKER	<a href="http://dasher.wustl.edu/ffe">http://dasher.wustl.edu/ffe</a>

**Tabla 17.1:** Listado de enlaces a programas y tutoriales para realizar simulaciones de dinámica molecular.



## 17.6. Bibliografía

- [1] S. A. Adcock and J. A. McCammon. Molecular dynamics: survey of methods for simulating the activity of proteins. *Chemical reviews*, 106(5):1589–1615, 2006.
- [2] B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *The Journal of chemical physics*, 27(5):1208, 1957.
- [3] O. Beckstein and M. S. P. Sansom. A hydrophobic gate in an ion channel: the closed state of the nicotinic acetylcholine receptor. *Phys Biol*, 3(2):147–159, 2006.
- [4] D. W. Borhani and D. E. Shaw. The future of molecular dynamics simulations in drug discovery. *J. Comput. Aided Mol. Des.*, 26(1):15–26, 2012.
- [5] B. R. Brooks, R. E. Brucolieri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4(2):187–217, 1983.
- [6] A. T. Brunger and P. D. Adams. Molecular dynamics applied to x-ray structure refinement. *Acc. Chem. Res.*, 35(6):404–412, 2002.
- [7] J. A. Bueren-Calabuig. *Modelado Molecular de la Interacción de Fármacos Antitumorales y Nucleasas con el ADN*. Universidad Alcalá, 2011.
- [8] J. A. Bueren-Calabuig, A. Negri, A. Morreale, and F. Gago. Rationale for the opposite stereochemistry of the major monoadducts and interstrand crosslinks formed by mitomycin c and its decarbamoylated analogue at cpg steps in dna and the effect of cytosine modification on reactivity. *Org. Biomol. Chem.*, 10(8):1543, 2012.
- [9] D. A. Case, T. E. Cheatham III, and T. Darden. The amber biomolecular simulation programs. *Journal of . . .*, 2005.
- [10] J. Chen, H.-S. Won, W. Im, H. J. Dyson, and C. L. Brooks. Generation of native-like protein structures from limited nmr data, modern force fields and advanced conformational sampling. *J. Biomol. NMR*, 31(1):59–64, 2005.
- [11] C. J. Cramer. *Essentials of Computational Chemistry: Theories and Models*. Wiley, 2 edition, 2004.
- [12] V. Daggett and A. Fersht. The present view of the mechanism of protein folding. *Nat. Rev. Mol. Cell Biol.*, 4(6):497–502, 2003.
- [13] G. de M Seabra, R. C. Walker, M. Elstner, D. A. Case, and A. E. Roitberg. Implementation of the scc-dftb method for hybrid qm/mm simulations within the amber molecular dynamics package. *J. Phys. Chem. A*, 111(26):5655–5664, 2007.
- [14] P. P. Ewald. Die berechnung optischer und elektrostatischer gitterpotentiale. *Ann. Phys.*, 369(3):253–287, 1921.
- [15] F. Gago. *Métodos computacionales de modelado molecular y diseño de fármacos*. Monografías de la Real Academia Nacional de Farmacia, 2009.
- [16] A. T. Hagler, E. Huler, and S. Lifson. Energy functions for peptides and proteins. i. derivation of a consistent force field including the hydrogen bond from amide crystals. *Journal of the American Chemical*, 1974.
- [17] F. Jensen. *Introduction to Computational Chemistry*. Wiley, 2 edition, 2006.
- [18] S. C. L. Kamerlin, S. Vicatos, A. Dryga, and A. Warshel. Coarse-grained (multiscale) simulations in studies of biophysical and chemical systems. *Annu Rev Phys Chem*, 62:41–64, 2011.
- [19] M. Karplus and J. A. McCammon. Molecular dynamics simulations of biomolecules. *Nature structural biology*, 9(9):646–652, 2002.
- [20] P. A. Kollman, I. Massova, C. Reyes, B. Kuhn, S. Huo, L. Chong, M. Lee, T. Lee, Y. Duan, W. Wang, O. Donini, P. Cieplak, J. Srinivasan, D. A. Case, and T. E. Cheatham. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. *Acc. Chem. Res.*, 33(12):889–897, 2000.
- [21] A. Leach. *Molecular Modelling: Principles and Applications (2nd Edition)*. Prentice Hall, 2 edition, 2001.
- [22] M. Levitt. A simplified representation of protein conformations for rapid simulation of protein folding. *Journal of molecular biology*, 104(1):59–107, 1976.
- [23] M. Levitt and A. Warshel. Computer simulation of protein folding. *Nature*, 253(5494):694–698, 1975.
- [24] J. A. McCammon, B. R. Gelin, and M. Karplus. Dynamics of folded proteins. *Nature*, 267(5612):585–590, 1977.

- [25] J. Michel, N. Foloppe, and J. W. Essex. Rigorous free energy calculations in structure-based drug design. *Mol. Inf.*, 29(8-9):570–578, 2010.
- [26] A. J. Mulholland. Chemical accuracy in qm/mm calculations on enzyme-catalysed reactions. *Chem Cent J*, 2007.
- [27] D. R. Nutt and J. C. Smith. Molecular dynamics simulations of proteins: can the explicit water model be varied? *J. Chem. Theory Comput.*, 3(4):1550–1560, 2007.
- [28] A. Patriksson and D. van der Spoel. A temperature predictor for parallel tempering simulations. *Phys Chem Chem Phys*, 10(15):2073–2077, 2008.
- [29] A. Pérez. *Estudio de mecanismos de interacción macromolecular*. Universidad de Barcelona, 2008.
- [30] A. Pérez, F. J. Luque, and M. Orozco. Frontiers in molecular dynamics simulations of dna. *Acc. Chem. Res.*, 45(2):196–205, 2012.
- [31] A. Pérez and M. Orozco. Real-time atomistic description of dna unfolding. *Angewandte Chemie (International ed*, 49(28):4805–4808, 2010.
- [32] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with namd. *J. Comput. Chem.*, 26(16):1781–1802, 2005.
- [33] G. Pierdominici-Sottile, N. A. Horenstein, and A. E. Roitberg. Free energy study of the catalytic mechanism of trypanosoma cruzitrans-sialidase. from the michaelis complex to the covalent intermediate. *Biochemistry*, 50(46):10150–10158, 2011.
- [34] J. W. Ponder and F. M. Richards. An efficient newton-like method for molecular mechanics energy minimization of large molecules. *J. Comput. Chem.*, 8(7):1016–1024, 1987.
- [35] J. P. Ryckaert, G. Ciccotti, and H. Berendsen. Sciencedirect.com - journal of computational physics - numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 1977.
- [36] M. S. P. Sansom, P. J. Bond, S. S. Deol, A. Grottesi, S. Haider, and Z. A. Sands. Molecular simulations and lipid-protein interactions: potassium channels and other membrane proteins. *Biochem. Soc. Trans.*, 33(Pt 5):916–920, 2005.
- [37] D. Schiff and L. Verlet. Ground state of liquid helium-4 and helium-3. *Phys. Rev.*, 160(1):208–218, 1967.
- [38] W. R. P. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. van Gunsteren. The gromos biomolecular simulation program package. *J. Phys. Chem. A*, 103(19):3596–3607, 1999.
- [39] D. E. Shaw, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, R. O. Dror, S. Piana, Y. Shan, B. Towles, J. K. Salmon, J. P. Grossman, K. M. Mackenzie, J. A. Bank, C. Young, M. M. Deneroff, and B. Batson. Proceedings of the conference on high performance computing networking, storage and analysis, 2009.
- [40] M. W. van der Kamp and A. J. Mulholland. Computational enzymology: insight into biological catalysts from modelling. *Nat. Prod. Rep.*, 25(6):1001, 2008.
- [41] P. K. Warme, F. A. Momany, R. S V, R. W. Tuttle, and S. H. A. Computation of structures of homologous proteins. alpha-lactalbumin from lysozyme. *Biochemistry*, 13(4):768–782, 1974.
- [42] A. Warshel and M. Karplus. Calculation of ground and excited state potential surfaces of conjugated molecules. 1. formulation and parametrization. *Journal of the American Chemical Society*, 16:5612–5625, 1972.
- [43] A. Warshel and M. Levitt. Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *J Mol Biol*, 103(2):227–49, 1976.
- [44] A. Warshel, M. Levitt, and S. Lifson. Consistent force field for calculation of vibrational spectra and conformations of some amides and lactam rings. *Journal of Molecular Spectroscopy*, 33(1):84–99, 1970.

# Capítulo 18

## Análisis de modos normales

*Raúl Méndez Giráldez*

### 18.1. Introducción al análisis de modos normales

*El análisis de modos normales es en realidad, el estudio de una superficie de energía potencial (que se presupone armónica) de manera analítica.* En el contexto de las macromoléculas se usa para identificar y caracterizar los movimientos colectivos de frecuencias más bajas, es decir, aquéllos de mayor amplitud ya que son justamente los más relevantes desde el punto de vista de su función. Este análisis asume que, dentro del rango de temperaturas fisiológicas, la superficie de energía potencial multidimensional que describe el equilibrio conformacional puede ser aproximada como una parábola centrada en un único mínimo energético (esto puede no ser cierto incluso a temperaturas fisiológicas).

El análisis de modos normales no es más que la generalización en múltiples dimensiones del oscilador armónico simple cuyos fundamentos pasan a ser descritos a continuación.

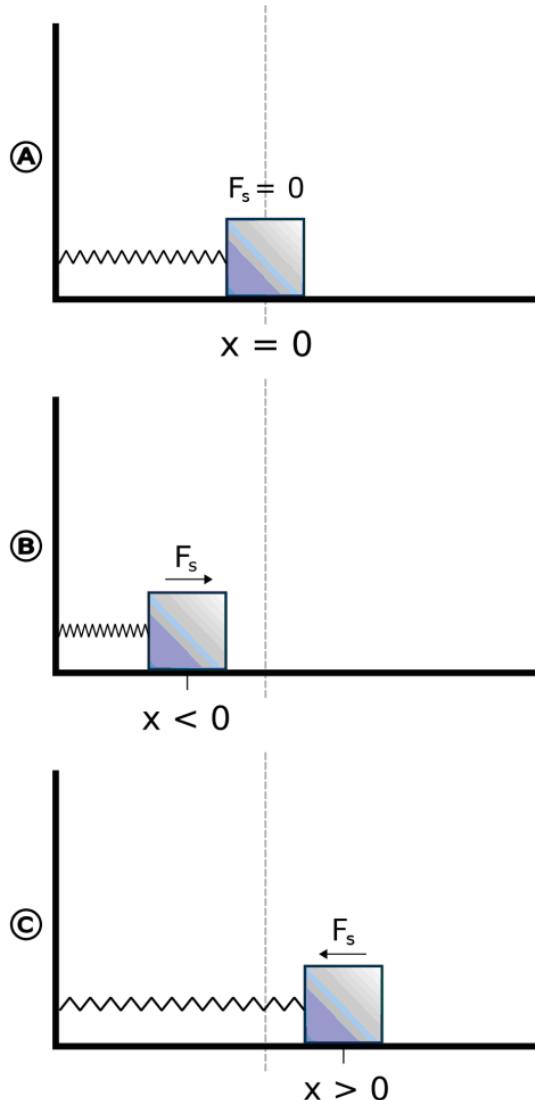
#### 18.1.1. El oscilador armónico simple

Supongamos que tenemos un objeto de masa  $m$  unido a un objeto estático a través de un muelle de masa despreciable, que se desliza sobre una superficie como la de la Figura 18.1. Como primera aproximación vamos a considerar también que el movimiento deslizante es sin rozamiento. De acuerdo con la *ley de Hooke*, toda masa  $m$  que se mueva una distancia  $x$  de su punto de equilibrio, experimentará inmediatamente una fuerza restauradora en sentido contrario al desplazamiento (hacia la posición de equilibrio) proporcional a dicho desplazamiento de acuerdo con la Ecuación 18.1, donde  $k > 0$  es la constante de fuerza de dicho muelle.

$$\vec{F} = -k\vec{x} \quad (18.1)$$

Es necesario decir que esta ley sólo se cumple para pequeñas perturbaciones del punto de equilibrio. En este contexto, si aplicáramos la *segunda ley de Newton* para describir la dinámica del sistema, tendríamos la siguiente ecuación diferencial de segundo orden lineal (Ecuación 18.2):

$$m\ddot{\vec{x}} = m \left( \frac{d^2\vec{x}}{dt^2} \right) = -k\vec{x} \quad (18.2)$$



**Figura 18.1:** Modelo de Oscilador armónico en 1 dimensión.

Se puede demostrar que la solución a la ecuación anterior es [12]:

$$x = A \cos(\omega t + \theta) \quad (18.3)$$

La Ecuación 18.3 proporciona la posición de la masa en cualquier instante. Como puede verse depende de varios parámetros: la frecuencia (angular)  $\omega$  que mide la velocidad con la que la masa  $m$  está pasando por el punto de equilibrio después de un ciclo completo; la amplitud  $A$  es la separación máxima del punto de equilibrio; y  $\theta$  es el ángulo de fase, de tal manera que en el instante  $t = 0$  la masa esté en el punto de equilibrio  $x = 0$ . Substituyendo la Ecuación 18.3 en la Ecuación 18.2 se llega a la expresión para el cálculo de la frecuencia:

$$\omega = \sqrt{\frac{k}{m}} \quad (18.4)$$

Los dos parámetros independientes  $A$  y  $\theta$  pueden calcularse de las condiciones iniciales, de la posición ( $x_0$ ) y velocidad ( $v_0$ ) iniciales:

$$\begin{aligned} x_0 &= A \cos(\theta) & v_0 &= \frac{d}{dt} [A \cos(\omega t + \theta)]_{t=0} = \omega A \sin(\theta) \\ A &= \left( x_0^2 + \frac{v_0^2}{\omega^2} \right)^{1/2} & \tan(\theta) &= -\frac{v_0}{x_0 \omega} \end{aligned} \quad (18.5)$$

Evidentemente el modelo del oscilador armónico descrito por la Ecuación 18.2 puede ser más realista y tener en cuenta una fuerza de fricción, que dependería linealmente con la velocidad (coeficiente  $b$ ) tal como se muestra en la Ecuación 18.6 (se utiliza la forma escalar por simplicidad). En este caso las oscilaciones armónicas están progresivamente amortiguadas en el tiempo, hasta llegar a una amplitud cero:

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = 0 \quad (18.6)$$

El oscilador armónico amortiguado puede ser aún generalizado al caso de que exista una fuerza externa aplicada a la misma masa. Hay muchos casos en Física en los que la fuerza que se considera es periódica en el tiempo. En este caso la ecuación final para el oscilador armónico forzado amortiguado se muestra en la Ecuación 18.7. La solución a dicha ecuación constaría de dos términos, uno temporal cuya amplitud decaería con el tiempo como en el caso del oscilador amortiguado, y un segundo término estacionario.

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = F(t) \quad (18.7)$$

Se pueden encontrar más detalles sobre las soluciones matemáticas para ambos modelos de oscilador armónico, el amortiguado y el forzado amortiguado, en el libro de texto “*Mechanics*” 3rd Edition, de Symon [12]. Para todo lo que vamos a discutir referido a macromoléculas vamos a requerir tan sólo el oscilador armónico simple. El caso más complejo del oscilador forzado amortiguado es la base de lo que se conoce con el nombre de Dinámica de Langevin [11], la cual queda fuera del alcance de esta sección y del libro.

### 18.1.2. Modos normales en espacio cartesiano

El *Principio de Superposición* establece que, para un sistema dinámico descrito por un modelo de oscilaciones armónicas, cualquier movimiento puede ser descompuesto como la suma de  $N$  oscilaciones armónicas independientes, donde  $N$  es el número de grados de libertad del sistema. Cada una de esas oscilaciones armónicas independientes recibe el nombre de “*Modo Normal*” y se caracteriza por una frecuencia y amplitud particulares.

Supongamos que tenemos la molécula de una proteína formada por  $N$  átomos, cuyas interacciones están caracterizadas por una función de energía potencial. Si desarrollamos dicha función de energía potencial genérica en serie de Taylor en el punto de mínima energía, pero no respecto a las coordenadas cartesianas tal cual, sino con respecto a las coordenadas cartesianas pesadas por las masas  $q_i = \sqrt{m_j} \Delta x_j$ ;  $q_{i+1} = \sqrt{m_j} \Delta y_j$ ;  $q_{i+2} = \sqrt{m_j} \Delta z_j$  quedaría:

$$V = \frac{1}{2} \sum_{i=1}^{3N} \frac{\partial^2 V}{\partial q_i \partial q_j} \Bigg|_{q_i q_j} \dots \quad (18.8)$$

O en notación matricial más compacta:

$$V \approx \frac{1}{2} q^t F q \quad (18.9)$$

Los términos lineales en la expansión de la Ecuación 18.8 son cero por definición, ya que las primeras derivadas son nulas en el mínimo de energía, así como también por convenio la energía en este punto. Estas condiciones son las que expresan la aproximación armónica, y por tanto los términos de orden superior a 2 se desprecian.  $F$  denota la matriz de derivadas segundas de la energía potencial con respecto las posiciones o también llamada *matriz Hessiana* (en analogía con el modelo de Hook, correspondería a la matriz de constantes de fuerza). En ausencia de fuerzas externas, la dinámica del sistema vendrá caracterizada por la ecuación:

$$\ddot{q} = F q + 0 \quad (18.10)$$

La Ecuación 18.10 en realidad es la forma matricial para un sistema de  $3N$  ecuaciones diferenciales de segundo orden lineales no independientes, análogos a la Ecuación 18.2 excepto que ahora los  $3N$  (donde  $N$  es el número de átomos) osciladores están acoplados. Dado que la matriz Hessiana es simétrica y definida positiva, se puede demostrar que existe la siguiente transformación llamada diagonalización, análoga a un cambio de base, tal que:

$$W^t F W = \Omega ; \text{ donde } \Omega_{ij} = 0 \Leftrightarrow i \neq j \quad (18.11)$$

Los elementos  $\Omega_{ii}$  de la matriz diagonal, son los valores propios, mientras que las columnas de la matriz  $W$  son los vectores propios correspondientes. La transformación de diagonalización definida por la Ecuación 18.11 nos permite definir un nuevo sistema de coordenadas:

$$Q = W^t q \equiv Q_i = \sum_{k=1}^{3N} W_{ki} q_k \quad (18.12)$$

Transformando el sistema de ecuaciones diferenciales lineales de segundo orden acopladas expresado en la Ecuación 18.10 según el sistema de coordenadas definido por la Ecuación 18.12, tenemos de nuevo un  $3N$  ecuaciones diferenciales de segundo orden, pero ahora desacopladas:

$$\ddot{Q} + \Omega Q = 0 \quad (18.13)$$

Cada solución independiente  $Q_i$  es una variable colectiva llamada coordenada de “*Modo Normal*” (o simplemente Modo Normal), que representa un oscilador armónico con una frecuencia  $\omega_i$ , amplitud y fase  $\theta_i$  características (tal y como habíamos visto para el oscilador armónico simple).

$$Q_i = A_i \cos(\omega_i t + \theta_i) \Rightarrow q_k = \sum_{i=1}^{3N-6} W_{ki} A_i \cos(\omega_i t + \theta_i) \quad (18.14)$$

La suma en la Ecuación 18.14 se extiende sobre todos los modos normales, excepto los primeros 6, de frecuencia nula, que corresponden a movimientos de cuerpo rígido. La amplitud y fase características de cada modo normal se podrían determinar a partir de las condiciones iniciales (velocidad y posición)

medidas experimentalmente. Sin embargo para la mayoría de aplicaciones, sólo nos fijaremos en los modos normales dados por las columnas de la matriz  $W$  de los vectores propios y sus frecuencias, dadas por la raíz cuadrada de los valores propios. Las amplitudes de cada modo normal se consideran en cierta manera factores multiplicativos, o dicho de otro modo, cuánto tenemos que elongar dicho modo normal para llegar a cierta conformación alejada de la de partida. Estas amplitudes o factores multiplicativos, no siempre se pueden calcular de manera directa.

En el equilibrio térmico, el *teorema de la equipartición* nos dice que la energía debe repartirse igualmente entre todos los grados de libertad. Por lo tanto, utilizando el formalismo de los modos normales se puede calcular las fluctuaciones cuadráticas media por átomo como:

$$\langle q_i \rangle = k_B T \sum_{k=1}^{3N-6} \left( \frac{W_{ik}}{\omega_k} \right)^2 \quad (18.15)$$

Donde  $k_B$  es la constante de Boltzmann y  $T$  la temperatura absoluta. Sumando la anterior expresión para todas las coordenadas de una macromolécula tendríamos:

$$\left\langle \sum_{i=1}^{3N-6} q_i^2 \right\rangle = \sum_{i=1}^{3N-6} \langle q_i \rangle = k_B T \sum_{k=1}^{3N-6} \frac{1}{\omega_k^2} \quad (18.16)$$

La Ecuación 18.16 muestra que *las fluctuaciones de menor frecuencia son aquellas que más contribuyen a las fluctuaciones totales de las macromoléculas*. Este es un resultado muy importante, que es inherente al análisis de modos normales independiente del modelo utilizado para calcular dichos modos.

Además, podríamos considerar las correlaciones cruzadas o las covarianzas entre las coordenadas Cartesianas tal y como se indica en la Ecuación 2.17.

$$\langle q_i q_j \rangle = k_B T \sum_{k=1}^{3N-6} W_{ik} W_{jk} \frac{1}{\omega_k^2} \equiv U = \langle qq' \rangle = k_B T W \Omega^{-1} W^t \quad (18.17)$$

Esta ecuación es importante porque indica que la *inversa de una matriz Hessiana* (lado derecho de la Ecuación 18.17, después del factor  $k_B T$ ) se corresponde con una *matriz de covarianzas*, o a la inversa. Dicho de otra manera, si somos capaces de calcular una matriz de covarianzas de las coordenadas Cartesianas, por ejemplo, tras una simulación de Dinámica Molecular (tal y como se ha explicado en la sección anterior) o como un conjunto de conformaciones de la misma macromolécula, podemos obtener también los modos normales después de diagonalizar dicha matriz. En el caso de una simulación de Dinámica Molecular, lo anteriormente formulado representa la equivalencia entre el *Análisis de Componentes Principales de una trayectoria de Dinámica Molecular* y el <sup>1</sup> hay que tener en cuenta que los modos normales resultantes son cuasi-armónicos, con frecuencias efectivas que tienen en cuenta los efectos del solvente así como otros efectos anarmónicos.: los vectores propios de la matriz de covarianza son a su vez los componentes principales y los modos normales, mientras que los valores propios asociados a cada componente principal de dicha representan la varianza del componente principal al cual está asociado o al inverso de la frecuencia al cuadrado de dicho modo normal.

1. *Minimización de los gradientes de energía potencial*, para que cumplamos la condición de mínimo de energía requerida para hacer válida la aproximación armónica expresada en la Ecuación 18.9.

---

<sup>1</sup>En el caso del análisis de componentes principales (*Principal Component Analysis* o PCA) de una trayectoria de Dinámica Molecular

2. *Cálculo de la matriz Hessiana* (en particular en las coordenadas Cartesianas pesadas por las masas), o de las segundas derivadas de la energía potencial con respecto de las posiciones.
3. *Diagonalización de la matriz Hessiana* para obtener los vectores propios (modos normales) y valores propios asociados (los cuadrados de las frecuencias).

El segundo y tercer paso son los computacionalmente más intensivos.

### 18.1.3. Modos normales en espacio diedro

Las macromoléculas se encuentran fluctuando en equilibrio térmico a temperaturas fisiológicas. Se sabe a partir de datos experimentales a dichas temperaturas, las longitudes y los ángulos de enlace varían muy poco, es decir que pueden considerarse como fijas en sus valores de equilibrio. La mayor parte de la flexibilidad proviene de la rotación de los enlaces rotables (alifáticos). En las proteínas estos enlaces son los ángulos  $\phi$ ,  $\varphi$  del esqueleto, y los ángulos  $\chi$  de las cadenas laterales, ya que el ángulo  $\omega$  del enlace peptídico permanece prácticamente rígido, normalmente en la conformación *trans* a  $180^\circ$  (aunque también se observan conformaciones *cis* a  $90^\circ$ ), como se muestra en la Figura 18.2.

Dado que, hay menos grados de rotación que cartesianos (en el caso de las proteínas en una relación de 1:8), realizar el *cálculo de modos normales en espacio diedro* resultaría en una reducción de los grados de libertad, es decir, una reducción de la dimensión de la matriz Hessiana que tiene que ser calculada y diagonalizada. Sin embargo, desde el punto de vista matemático, la formulación resulta más complicada. A continuación se resumirán las características más importantes del cálculo de modos normales en espacio diedro.

Lo que complica las expresiones matemáticas para el cálculo de modos normales en ángulos diedros es el hecho de que el término de energía cinética no puede expresarse como función simple de los ángulos diedros, tal y como ocurre en el caso de las coordenadas cartesianas pesadas por las masas. Además dicho cálculo de modos normales torsionales, deben de ser llevado a cabo en un sistema de referencia que garantice que un cambio en los ángulos diedros no resulte en ningún movimiento ni de rotación ni de translación de toda la molécula. Esto se consigue imponiendo las *condiciones de Eckart* [6] al *cálculo de la matriz Jacobiana* para la transformación del espacio cartesiano al de ángulos diedro.

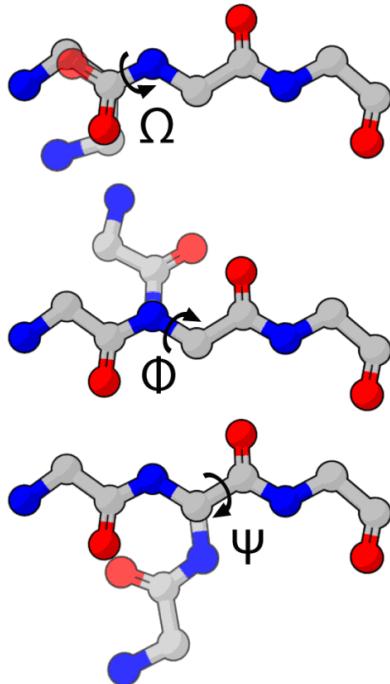
$$J_{ij} = \frac{\partial q_i}{\partial \phi_j} \quad (18.18)$$

A partir del cálculo de la matriz Jacobiana se puede calcular el *tensor de masas*  $H$ , y consecuentemente podemos calcular la *matriz de la energía cinética*  $T$ .

$$H = J^t J \quad (18.19)$$

$$T = \frac{1}{2} \phi^t H \phi \quad (18.20)$$

Donde  $\phi$  es el vector columna cuyos elementos son las derivadas con respecto del tiempo de los ángulos diedros. El cálculo de los modos normales en espacio diedro, sería análogo al de coordenadas cartesianas, excepto que en este caso se debe resolver un problema de valor propio generalizado, donde la matriz de vectores propios diagonaliza simultáneamente el tensor de masas  $H$  a la matriz identidad y la Hessiana a la matriz de valores propios.



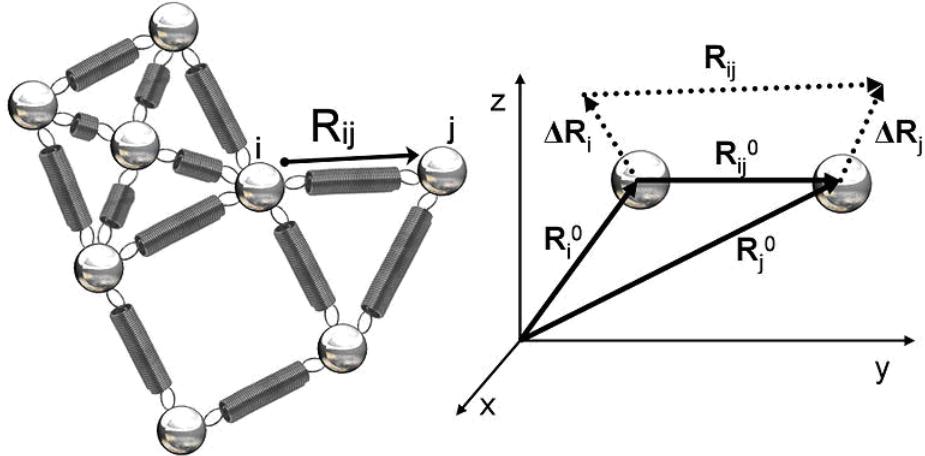
**Figura 18.2:** Ángulos diedros del esqueleto de las proteínas  $\phi$ ,  $\varphi$  y  $\omega$ .

## 18.2. Modelos de redes elásticas

El formalismo presentado hasta aquí es exacto, en el sentido que usa una descripción atomística completa de las macromoléculas, pero sufre las mismas limitaciones que los campos de fuerza que se utilizan para calcular la energía potencial del sistema. En términos de eficiencia computacional, el análisis de modos normales puede ser muy costoso para macromoléculas de gran tamaño, especialmente en cuanto a la cantidad de memoria necesaria para diagonalizar matrices Hessianas de gran tamaño  $3N \times 3N$  (donde  $N$  es el número de átomos). Además, la minimización de energía necesaria, previa al cálculo de la matriz Hessiana, puede distorsionar la estructura de la macromolécula significativamente, produciendo artefactos.

Alternativamente Tirion [13] demostró que un modelo simple, de un solo parámetro puede proporcionar resultados muy similares al de los cálculos atomísticos completos. El *modelo de Tirion* asume que la estructura de la macromolécula, en particular de las proteínas, tal como se determina a partir de los datos de cristalografía de Rayos X, se encuentra por definición en el mínimo absoluto de energía, por lo tanto no requiere ninguna minimización previa de la energía. Además el modelo utiliza un potencial armónico simple de pares de átomos con la misma constante de fuerza para todos los pares de átomos, si y sólo si, el par de átomos en consideración se encuentran a una distancia por debajo de un valor umbral determinad, o cero en caso contrario.

El modelo de Tirion inspiró varios otros que son incluso más simples en el sentido que consideran las proteínas como redes hechas de eslabones (los átomos  $C\alpha$  en las proteínas p. ej.), conectadas por muelles a otros átomos si y sólo si se encuentra a una distancia menor que la distancia umbral dada. Estos modelos se conocen genéricamente con el nombre de *Modelos de redes elásticas*.



**Figura 18.3:** El modelo de red elástica del GNM. Los nodos (átomos  $C\alpha$ ) se consideran conectados por muelles si se encuentran más cercanos que una distancia umbral (típicamente  $7\text{\AA}$ ). Las posiciones de equilibrio de los nodos se etiquetan con el superíndice 0, las fluctuaciones instantáneas se identifican con el símbolo  $\Delta$ , y las diferencias entre las posiciones con los subíndices  $ij$ .

### 18.2.1. El modelo de red Gaussiana o GNM

El *modelo de red Gaussiana* (ver Figura 2.3) es un caso particular del modelo de Tirion, y fue propuesto por Bahar y colaboradores [2]. Este modelo, pensado sobretodo para proteínas, considera éstas como si estuvieran hechas de eslabones (situadas cada átomo de  $C\alpha$ ). Cada par de átomos  $i, j$  se considera que interaccionan (con valor simbólico de interacción de  $-1$ ) si se encuentran más próximos que una distancia umbral determinada. La interacción de cada átomo consigo mismo es su conectividad. Una matriz de esta manera es en realidad un *matriz de Kirkchoff de contactos* (Ecuación 18.21)

$$\Gamma_{ij} = \begin{cases} -1 & \Leftrightarrow s_{ij} \leq r_C \\ 0 & \Leftrightarrow s_{ij} > r_C \end{cases} \quad (18.21)$$

$$\Gamma_{ii} = \sum_{k,k \neq i}^N \Gamma_{ik}$$

De esta manera, matriz de Kirkchoff hace el papel de la matriz Hessiana, describiendo la energía potencial (El modelo de red Gaussiana o GNM) así como la dinámica del sistema (El modelo de red Gaussiana o GNM).

$$V = (\gamma/2) \Delta R^t \Gamma \Delta R \quad (18.22)$$

(donde  $\gamma$  representa la constante de rigidez)

$$(\gamma \ddot{R}) - \Gamma \Delta R = 0 \quad (18.23)$$

Análogamente al análisis de modos normales utilizando una descripción atomística completa, la diagonalización de la matriz de Kirkchoff proporciona los modos normales (es decir, los vectores propios, o

lo que es lo mismo, los vectores columna de la matriz  $U$  en la El modelo de red Gaussiana o GNM) y sus frecuencias asociadas (es decir, los valores propios, o la matriz  $\Lambda$  diagonal):

$$U^t \Gamma U = \Lambda \quad (18.24)$$

Los modos normales del GNM no proporcionan información tridimensional sobre direcciones de fluctuación absolutas; en cambio, nos permiten calcular las amplitudes promedio de para residuos (de una proteína) individuales).

$$\langle (\Delta R_i)^2 \rangle = (3k_B T / \gamma) \sum_k [\Gamma^{-1}]_{ii} \quad (18.25)$$

Aquí  $[\Gamma^{-1}]_{ii}$  denota el elemento diagonal  $ii$  del inverso de la matriz de Kirkhoff, cuyo determinante es nulo, y por tanto no puede ser invertida, por lo que se calcula con los valores propios no nulos y sus vectores propios asociados. El GNM sólo es invariante respecto a las traslaciones de la molécula entera, por tanto sólo tiene 1 valor propio nulo, o lo que es lo mismo  $N-1$  grados de libertad (donde  $N$  es el número de residuos de la proteína considerada). La constante de rigidez  $\gamma$  puede ser estimada mediante regresión lineal de los valores experimentales de los factores B de las estructuras cristalinas de las proteínas (determinadas por difracción de Rayos X), vs. los valores predichos (que se calculan según la El modelo de red Gaussiana o GNM)

$$B_i = \frac{8\pi}{3} \langle (\Delta R_i)^2 \rangle \quad (18.26)$$

### 18.2.2. El modelo de red Anisotrópica o ANM

El *modelo de red anisotrópica* es una generalización del GNM, en el sentido que permite fluctuaciones diferentes en cada una de las direcciones cartesianas. Este modelo fue desarrollado por Bahar y colaboradores [1]. A continuación desarrollaremos las expresiones para calcular la matriz Hessiana de acuerdo con este modelo. Dado que el ANM tiene en cuenta las diferentes direcciones en las fluctuaciones, sus ecuaciones son un poco más complicadas que las del GNM.

La función de potencial que describe la interacción entre los residuos  $i, j$  en términos de sus coordenadas  $x, y, z$  y sus distancias de equilibrio ( $R_{ij}^0$ ) y sus distancias tras una perturbación ( $R_{ij}$ ):

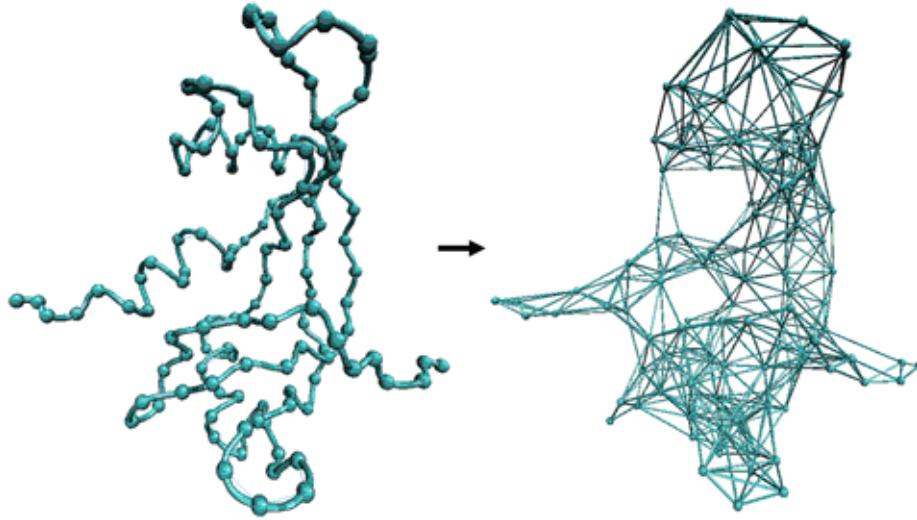
$$V_{ij} = \frac{1}{2} \gamma (R_{ij} - R_{ij}^0)^2 = \frac{1}{2} \gamma \left( [(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2]^{1/2} - R_{ij}^0 \right)^2 \quad (18.27)$$

Y las segundas derivadas con respecto a las coordenadas  $x, y, z$  quedan:

$$\frac{\partial V_{ij}}{\partial x_i} = -\gamma(x_j - x_i) \left( 1 - \frac{R_{ij}^0}{R_{ij}} \right) = -\frac{\partial V_{ij}}{\partial x_j} \quad (18.28)$$

$$\frac{\partial^2 V_{ij}}{\partial x_i^2} = \gamma \left( 1 + \frac{R_{ij}^0(x_j - x_i)^2}{R_{ij}^3} - \frac{R_{ij}^0}{R_{ij}} \right) = \frac{\partial^2 V_{ij}}{\partial x_j^2} \quad (18.29)$$

Las expresiones para las derivadas con respecto a  $y_i$  e  $z_i$  son análogas, substituyendo dichas coordenadas en  $x_i$  en la Ecuación 18.28 y Ecuación 18.29.



**Figura 18.4:** La red elástica del ANM. Los pares de átomos que interactúan aparecen conectados.

En el equilibrio  $R_{ij} = R_{ij}^0$ , entonces

$$\frac{\partial V_{ij}}{\partial x_i} = 0 \quad (18.30)$$

$$\frac{\partial^2 V_{ij}}{\partial x_i^2} = \gamma \frac{(x_j - x_i)^2}{R_{ij}^2} \quad (18.31)$$

Así mismo las segundas derivadas cruzadas serían:

$$\frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} = -\frac{\gamma(x_j - x_i)(y_j - y_i)}{R_{ij}^2} = \frac{\partial^2 V_{ij}}{\partial x_i \partial y_j} \quad (18.32)$$

Sin embargo las ecuaciones para las segundas derivadas sólo servirían para interacciones de pares simple, es decir un residuo interaccionando con otro a una distancia menor que el valor umbral. Si queremos tener en cuenta la conectividad de los residuos tenemos que sumar sobre todos los posibles vecinos para un residuo dado  $i$  (esa suma está dada por el valor  $\Gamma_{ii}$ ):

$$\frac{\partial^2 V}{\partial x_i^2} = \gamma \sum_j^{\Gamma_{ii}} \frac{(x_j - x_i)^2}{R_{ij}^2} \quad (18.33)$$

$$\frac{\partial^2 V}{\partial x_i \partial y_i} = \gamma \sum_j^{\Gamma_{ii}} \frac{(x_j - x_i)(y_j - y_i)}{R_{ij}^3} \quad (18.34)$$

La matriz Hessiana para un caso de una proteína de  $N$  residuos conectados con  $M$  interacciones es una matriz  $3N \times 3N$  que puede ser considerada como hecha de  $N \times N$  super-elementos de tamaño  $3 \times 3$ :

$$H = \begin{pmatrix} H_{1,1} & H_{1,2} & \cdots & H_{1,N} \\ H_{2,1} & H_{2,2} & \cdots & H_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ H_{N,1} & H_{N,2} & \cdots & H_{N,N} \end{pmatrix} \quad (18.35)$$

Donde los elementos  $H_{ij}$  para  $i \neq j$  son:

$$H_{ij} = \begin{pmatrix} \frac{\partial^2 V}{\partial x_i \partial x_j} & \frac{\partial^2 V}{\partial x_i \partial y_j} & \frac{\partial^2 V}{\partial x_i \partial z_j} \\ \frac{\partial^2 V}{\partial y_i \partial x_j} & \frac{\partial^2 V}{\partial y_i \partial y_j} & \frac{\partial^2 V}{\partial y_i \partial z_j} \\ \frac{\partial^2 V}{\partial z_i \partial x_j} & \frac{\partial^2 V}{\partial z_i \partial y_j} & \frac{\partial^2 V}{\partial z_i \partial z_j} \end{pmatrix} \quad (18.36)$$

Los elementos de  $H$  se calculan según la Ecuación 18.33 para los elementos de la diagonal ( $H_{ij}, i = j$ ), y Ecuación 18.34 para el resto ( $H_{ij}, i \neq j$ ).

De nuevo los modos normales se calculan después de diagonalizar la matriz  $H$ , al igual que se hacía con la matriz  $\Gamma$  en la Ecuación 18.24. Esto proporciona  $3N - 6$  modos normales de frecuencia no nulas (hay 3 movimientos de cuerpo rígido de rotación y 3 de rotación de la proteína entera que tienen frecuencia cero).

Al igual que el GNM, el ANM permite además la estimación de las fluctuaciones cuadráticas medias por residuos, y compararlas con los valores experimentales a partir de los factores B de las estructuras cristalinas, de tal manera que podamos estimar la constante de rigidez a partir de la regresión lineal de los primeros valores con respecto a los últimos. Respecto a las distancias umbral se estima que tienen que ser superiores para el ANM con respecto al GNM, para proporcionar fuerzas de muelles similares, es decir dichos umbrales tienen que estar en el rango de 12–15 Å.

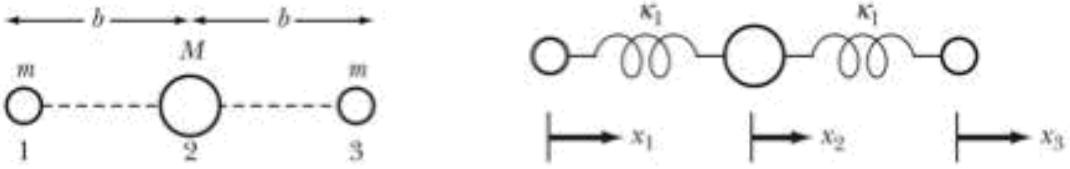
### 18.3. La molécula triatómica lineal

Cálculo analítico de modos normales para la molécula triatómica lineal

El siguiente ejemplo ha sido extraído del libro de texto “*Classical Mechanics. 3rd Edition*” escrito por Goldstein [7]. Imaginemos que tenemos una *molécula triatómica lineal* tal como la molécula de CO<sup>2</sup> (O=C=O). Vamos a llevar a cabo un cálculo de modos normales para descomponer las vibraciones de esta molécula en movimientos armónicos independientes con frecuencias características. En términos de la aproximación armónica podemos imaginar la molécula hecha de dos masas iguales (los átomos de Oxígeno) situadas en los extremos unidas a una diferente en el centro (el átomo de Carbono), por dos muelles de igual constante de fuerza cada uno  $k$ . Los tres átomos se encuentran alineados, con unas distancias de equilibrio iguales  $a$  y  $b$  (Figura 18.5).

Por razones de simplicidad, consideraremos solamente las vibraciones que tienen lugar en el eje que une los tres átomos, y que dichas vibraciones (de ambos enlaces) no están acopladas. En esas condiciones, la energía potencial del sistema de nuestro sistema vendrá descrita por:

$$V = \frac{k}{2}(x_2 - x_1 - b)^2 + \frac{k}{2}(x_3 - x_2 - b)^2 \quad (18.37)$$



**Figura 18.5:** Molécula triatómica lineal vista como un conjunto de osciladores armónicos.

Podemos simplificar las ecuaciones si introducimos un nuevo conjunto de coordenadas que correspondan a los desplazamientos internos:

$$\eta_i = x_i - x_{i0} \quad (18.38)$$

Con la restricción que las diferencias entre las posiciones de equilibrio sean ambas el parámetro  $b$ :

$$x_{02} - x_{01} = b = x_{03} - x_{02} \quad (18.39)$$

Substituyendo la Ecuación 18.38 y la Ecuación 18.39 dentro de la Ecuación 18.37 obtenemos una nueva expresión para la energía potencial (Ecuación 18.40):

$$V = \frac{k}{2}(\eta_1^2 + 2\eta_2^2 + \eta_3^2 - 2\eta_1\eta_2 - 2\eta_2\eta_3) \quad (18.40)$$

O en forma matricial:

$$V = \frac{1}{2}(\eta_1 \ \eta_2 \ \eta_3) \begin{pmatrix} k & -k & 0 \\ -k & 2k & -k \\ 0 & -k & k \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix} \equiv \frac{1}{2}\eta^t V \eta \quad (18.41)$$

Análogamente, la matriz de energía cinética nos quedaría como:

$$T = \frac{m}{2}(\dot{\eta}_1^2 + \dot{\eta}_3^2) + \frac{M}{2}(\dot{\eta}_2^2) = \frac{1}{2}(\dot{\eta}_1^2 \ \dot{\eta}_2^2 \ \dot{\eta}_3^2) \begin{pmatrix} m & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & m \end{pmatrix} \begin{pmatrix} \dot{\eta}_1^2 \\ \dot{\eta}_2^2 \\ \dot{\eta}_3^2 \end{pmatrix} \equiv \frac{1}{2}\dot{\eta}^t T \dot{\eta} \quad (18.42)$$

Dado que no estamos trabajando con coordenadas Cartesianas pesadas por las masas (lo podríamos hacer y el resultado obviamente no cambiaría), tenemos que resolver un problema de valor propio generalizado, donde la matriz  $W$  de los vectores propios diagonaliza simultáneamente la matriz de energía potencial a la matriz diagonal  $V$  de valores propios y la matriz de la energía cinética  $T$  a la matriz identidad (Ecuación 18.43):

$$(V - \omega^2 T)W = 0 \quad (18.43)$$

Si hubiéramos utilizado coordenadas Cartesianas pesadas por las masas, la matriz  $T$  de energía cinética se convertiría en la matriz identidad por lo que en realidad la Ecuación 18.43 se reduciría a una ecuación de valor propio ordinario. La matriz  $\omega^2$  representa la matriz diagonal de valores propios, cuyos elementos

son los cuadrados de las frecuencias de vibración. Para que la Ecuación 18.43 no tenga una solución trivial, debe cumplir que el determinante secular sea igual a cero:

$$|V - \omega^2 T| = 0 = \begin{vmatrix} k - \omega^2 m & -k & 0 \\ -k & 2k - \omega^2 M & -k \\ 0 & -k & k - \omega^2 m \end{vmatrix} = \omega^2(k - \omega^2 m)[k(M + 2m) - \omega^2 Mm] = 0 \quad (18.44)$$

El polinomio de la Ecuación 18.44 es cúbico en  $\omega^2$  y tiene las siguientes 3 raíces:

$$\omega_1 = 0; \omega_2 = \sqrt{\frac{k}{m}}; \omega_3 = \sqrt{\frac{k}{m} \left(1 + \frac{2m}{M}\right)} \quad (18.45)$$

A fin de obtener los componentes de los vectores propios (los modos normales en sí) hay substituir cada una de las frecuencias obtenidas en la Ecuación 18.45 dentro de la Ecuación 18.43, y como se trata de un sistema de ecuaciones lineales homogéneo, de tal manera que por cada modo normal tenemos un sistema de 3 ecuaciones lineales homogéneas de la siguiente forma:

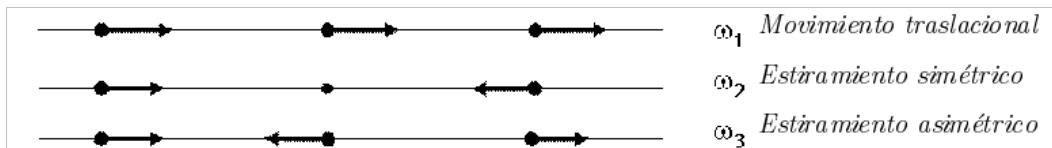
$$\begin{array}{ccc|c} (k - \omega_j^2 m)w_{1j} & -kw_{2j} & -kw_{3j} & = 0 \\ -kw_{1j} & (2k - \omega_j^2 M)w_{2j} & -kw_{3j} & = 0 \\ -kw_{2j} & -kw_{3j} & (k - \omega_j^2 m)w_{3j} & = 0 \end{array} \quad (18.46)$$

El subíndice  $j$  denota el índice para el modo normal (y va de 1 a 3). Como se puede ver, para resolver este sistema de ecuaciones (que son linealmente dependientes) hay que añadir una ecuación extra, la condición de normalización de cada uno de los modos normales (Ecuación 18.47). Estos modos normales obtenidos deben de ser ortogonales con respecto a la matriz de energía cinética.

$$m(w_{1j}^2 + w_{3j}^2) + Mw_{2j}^2 = 1 \quad (18.47)$$

Resolviendo el sistema formado por la Ecuación 18.47 y la Ecuación 18.48 para cada modo normal, de frecuencia asociada calculada en la Ecuación 2.44, obtenemos los 3 modos normales:

$$\begin{aligned} \omega_1 &= 0 & \text{Primer Modo Normal: } & \frac{1}{\sqrt{2m+M}}(1 \ 1 \ 1) \\ \omega_2 &= \sqrt{\frac{k}{m}} & \text{Segundo Modo Normal: } & \frac{1}{\sqrt{2m}}(1 \ 0 \ -1) \\ \omega_3 &= \sqrt{\frac{k}{m}(1 + \frac{2m}{M})} & \text{Tercer Modo Normal: } & \frac{1}{\sqrt{2m(1 + \frac{2m}{M})}}(1 \ -\frac{2m}{M} \ 1) \end{aligned} \quad (18.48)$$



**Figura 18.6:** Representación esquemática de los movimientos de vibración para cada uno de los modos normales.

Mirando la Figura 18.6 se puede apreciar que el modo normal de más baja frecuencia corresponde a un movimiento de traslación de toda la molécula. Este modo aparece porque hemos considerado 3 movimientos internos, pero en realidad hay sólo 2. El movimiento de cuerpo rígido puede ser evitado imponiendo la invariancia del centro de masas en las ecuaciones iniciales:

$$m(x_1 + x_3) + Mx_2 = 0 \quad (18.49)$$

En aplicaciones ordinarias de cálculo de modos normales para macromoléculas, tales como las proteínas, aún incluso en el contexto de las redes elásticas, donde las matrices Hessianas típicamente tienen miles de elementos, el problema de valores propios se resuelve directamente utilizando algoritmos optimizados implementados en programas informáticos, tal y como se verá en los ejemplos que siguen, sin necesidad de calcular el determinante secular que puede ser muy ineficiente.

## 18.4. Ejemplos prácticos del cálculo de modos normales

A continuación vamos a mostrar algunos ejemplos de cálculo de modos normales en el contexto del ANM, utilizando el paquete informático ProDy.

### 18.4.1. Introducción a ProDy

*ProDy* [4] es un conjunto de programas y librerías desarrollados en el lenguaje de programación Python que permiten el análisis de flexibilidad en biomoléculas: análisis de modos normales, análisis de componentes principales de una trayectoria de Dinámica Molecular o de un conjunto de estructuras de proteína, representación gráfica de resultados así como comparación de los diferentes métodos para calcular modos normales,... etc. Existen versiones para la mayoría de sistemas operativos que se distribuyen totalmente gratis<sup>2</sup>.

### 18.4.2. Cálculo de modos normales para el transportador de leucina

Una vez instalado ProDy vamos a mostrar cómo hacer un cálculo sencillo de modos normales, utilizando el modo interactivo, dentro de la *shell* que genera python propiamente. En Linux, simplemente ejecutar ‘python’<sup>3</sup> en una consola (Sección 4.3) escribiendo:

```
$ python
```

Que generará la siguiente respuesta por pantalla:

```
Python 2.7.3 (default, May 29 2012, 14:54:22)
[GCC 4.6.3 20120306 (Red Hat 4.6.3-2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Una vez dentro de la shell interactiva de Python, vamos a cargar todas las librerías y programas de Prody (el signo ‘\*’ hace referencia a todas las librerías):

```
>>> from prody import *
```

---

<sup>2</sup>ProDy Project. <http://www.csb.pitt.edu/prody>

<sup>3</sup>Para que ProDy funcione correctamente, tal y como se indica en la web oficial, hay que utilizar Python 2.6 o 2.8.

Si ProDy se ha instalado correctamente no debe de producirse ningún mensaje de error. A continuación leemos el fichero PDB que contiene los datos de la estructura del transportador de Leucina LeuT, en su forma monomérica y sin ningún ligando (fichero 1usg.pdb). Para ello utilizaremos la función *ParsePDB* que además es capaz de bajarse el fichero PDB correspondiente del repositorio Worldwide Protein Data Bank<sup>4</sup> (wwPDB).

```
>>> leuT = parsePDB('1usg')
```

El resultado de la función ParsePDB es un *objeto* que en este caso llamamos ‘leuT’. Siempre podemos comprobar si un objeto determinado se ha creado correctamente simplemente ejecutando su nombre como si de un comando se tratase:

```
>>> leuT
<AtomGroup: 1usg (2775 atoms)>
```

Una vez tenemos el objeto ‘leuT’ con los datos de dicha estructura de proteína, procedemos a calcular la matriz Hessiana según el modelo de red elástica. Pero antes, tal y como habíamos visto en la sección Subsección 18.2.2 vamos a hacer una *selección de sólo los carbonos alfa*. Existen dos formas para hacerlo:

```
>>> catoms = leuT.select('protein and name CA')
>>> catoms
<Selection: 'protein and name CA' from 1usg (345 atoms)>
>>> catoms2 = leuT.select('calpha')
>>> catoms2
<Selection: 'calpha' from 1usg (345 atoms)>
```

En la primera forma se hace uso del subconjunto pre-definido ‘protein’ que contiene todos los átomos, y su intersección con el subconjunto átomos llamados ‘ca’ o carbonos alfa. En el segundo caso se hace directamente referencia al subconjunto ‘calpha’. Para comprobar que ambas selecciones son iguales:

```
>>> catoms == catoms2
True
>>>
```

Antes de proceder al cálculo de la matriz Hessiana del ANM, utilizando sólo los carbonos alfa, debemos crear una *instancia* (un objeto) ‘ANM’ que contiene los métodos para realizar todos los cálculos pertinentes

```
>>> anm = ANM('LeuT ANM analysis')
anm.buildHessian(catoms)
>>>
```

La cadena de caracteres que le pasamos como argumento es simplemente un nombre. Podemos ver los tres primeros y los tres últimos elementos de dicha matriz (si queremos redondeando a 3 decimales los valores mostrados) con el siguiente comando:

```
>>> print(anm.getHessian().round(3))
[[ 10.259 -2.537  3.526 ...   0.      0.      0.    ]
 [-2.537  6.903 -0.418 ...   0.      0.      0.    ]
 [ 3.526 -0.418  8.839 ...   0.      0.      0.    ]
 ...
 [ 0.      0.      0.      ...  13.908  1.178 -0.78 ]
 [ 0.      0.      0.      ...  1.178  14.792  2.665]
 [ 0.      0.      0.      ... -0.79   2.655  14.3  ]]
>>
```

A la hora de calcular la *Matriz Hessiana* no se le ha pasado ningún parámetro, pero se le podían especificar por ejemplo el valor umbral para considerar que dos residuos interaccionan (por defecto es 14 Å) y/o el valor de la constante de “dureza” gamma (por defecto 1):

---

<sup>4</sup>Worldwide Protein Data Bank. <http://www.wwpdb.org>

```
>>> anm.buildHessian(catoms,cutoff=15.0,gamma=1.0)
```

Ahora ya se puede realizar la *diagonalización de la matriz Hessiana*, que es el cálculo de los modos normales propiamente dichos. En el ejemplo que mostramos a continuación por simplicidad se van a calcular sólo los 20 primeros modos de más baja frecuencia.

```
>>> anm.calcModes(n_modes=20,zeros=False,turbo=True)
```

En lo anterior se podría también haber llamado al método ‘calcModes’ sin pasarle ningún argumento, con los paréntesis vacíos ‘()’ como en el caso de ‘buildHessian’. Haciéndolo de esta manera se pueden cambiar varios de los parámetros más importantes, cuyos valores por defecto son los que se han indicado (cálculo de los 20 modos de más baja frecuencia, no mostrar los modos normales de frecuencia nula que corresponden a movimientos de cuerpo rígido, y utilizar aceleración de cálculo que utiliza mayor memoria pero es más rápida). Al igual que anteriormente podemos inspeccionar el contenido resumido de los valores (frecuencias de oscilación al cuadrado) y vectores propios (los modos normales propiamente dichos) mediante el comando:

```
>>> print( anm.getEigvals().round(3) )
[ 0.087  0.121  0.321  0.747  0.966  1.018  1.678  2.028  2.251  2.637
 2.694  2.819  2.859  2.961  3.141  3.452  3.59   3.805  3.964  4.033]
>>> print( anm.getEigvecs().round(3) )
[[ 0.002  0.061  0.007 ... -0.011  0.059  0.092]
 [-0.028  0.006  0.085 ... -0.096  0.054  0.103]
 [-0.009 -0.057  0.01   ... -0.011 -0.048 -0.099]
 ...
 [ 0.029 -0.039  0.006 ... -0.022  0.004  0.005]
 [ 0.027 -0.005 -0.041 ...  0.019  0.014 -0. ]
 [-0.046  0.036  0.011 ... -0.001  0.005  0.017]]
>>>
```

#### 18.4.3. Cálculo de los factores B

Cálculo de los factores B y otras propiedades derivadas de los modos normales

Como se mencionó en la Subsección 18.2.2 el ANM permite calcular los *factores B* a partir de las fluctuaciones cuadráticas medias. Utilizando el ejemplo anterior, vamos a mostrar el perfil de los factores B calculados a partir de los modos normales del ANM y compararlos con los correspondientes valores experimentales de la estructura determinada por difracción de Rayos X.

Primeramente calcularemos los *factores B de los carbonos alfa* del mismo transportador de Leucina a partir de la Ecuación 18.15 donde ahora los vectores propios corresponden a los modos normales según el ANM:

```
>>> bfact_calc=calcTempFactors(anm[0:19],catoms)
```

En este caso utilizamos solo los 20 primeros modos (del 0 al 19) ya que son los que hemos calculados anteriormente, pero se podrían utilizar todos sin ningún problema. Después se extraen los correspondientes factores B de los carbonos alfa (la selección de átomos que llamamos catoms) anotados en el fichero PDB (‘1usg.pdb’) que habíamos cargado en memoria.

```
>>> bfact_exp=catoms.getBetas()
```

Además utilizando las librerías numéricas de Python queremos calcular el *coeficiente de correlación entre los factores B calculados según el ANM y los experimentales*.

```
>>> import numpy as np
>>> r=np.corrcoef(bfactor_calc,bfactor_exp)[0,1]
>>> r
```

```
0.52508223958167421
```

Como se puede ver *el valor de correlación es estadísticamente significativo*, pero no es demasiado alto. ¿Por qué crees que pasa esto? Para una discusión al respecto se puede consultar la referencia [3]. Además vamos a representar gráficamente el perfil de los factores B calculados y experimentales. Para ello, debemos de generar una serie de valores que sean las “x” comunes para los factores B calculados y experimentales, que corresponden al número de residuo para cada carbono alfa.

```
>>> res_num=catoms.Resnums()
```

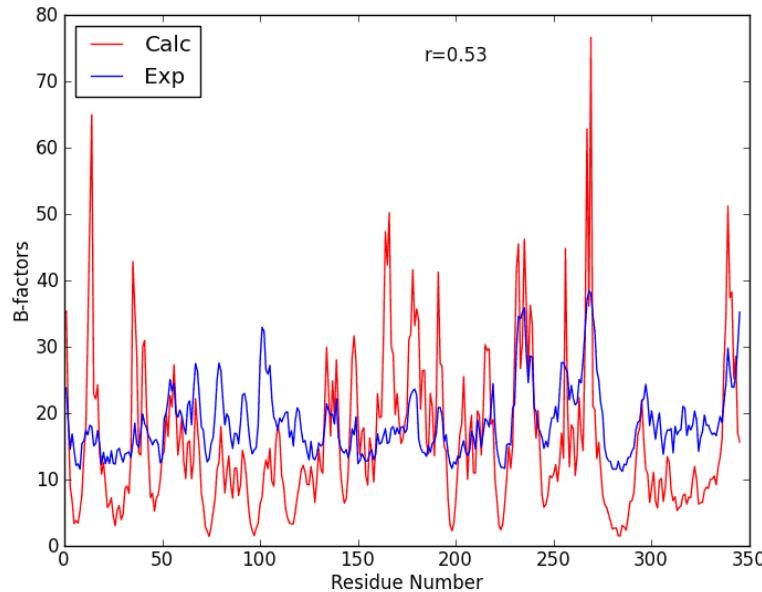
Para hacer la representación gráfica necesitamos importar las librerías gráficas para Python ‘matplotlib’ (en particular las ‘pyplot’) tal y como sigue

```
>>> import matplotlib.pyplot as plt
```

Representamos el perfil de los factores B calculados versus el número de residuo en rojo con línea lisa ('r-') y el perfil de los factores B experimentales en azul con línea lisa ('b-') en el mismo gráfico.

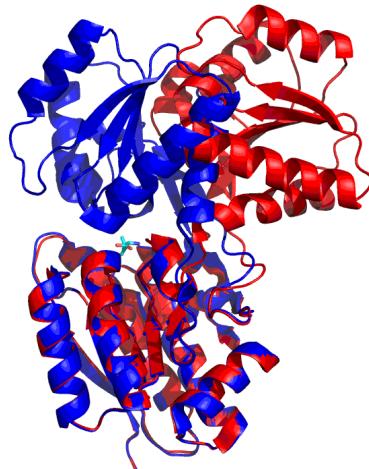
```
>>> plt.plot(res_num,bfact_calc,'r-')
[<matplotlib.lines.Line2D object at 0x41607d0>]
>>> plt.plot(res_num,bfact_exp,'b-')
[<matplotlib.lines.Line2D object at 0x3d9ef50>]
>>> plt.legend(('Calc','Exp'),loc='upper left')
<matplotlib.legend.Legend object at 0x4160b10>
>>> plt.xlabel('Residue Number')
<matplotlib.text.Text object at 0x413cbd0>
>>> plt.ylabel('B-factors')
<matplotlib.text.Text object at 0x415f350>
>>> plt.text(185,73,'r='+'%.2f' % c)
<matplotlib.text.Text object at 0x4160310>
>>> plt.show()
```

El comando ‘plt.show()’ muestra la figura en pantalla, y desde ese mismo terminal se puede guardar como fichero ‘.png’ quedando tal y como se muestra en la Figura 18.7.



**Figura 18.7:** Perfiles de factores B calculados y experimentales para los carbonos alfa del transportador de Leucina LeuT.

Una de las aplicaciones más corrientes de los modos normales es el estudio de los cambios de conformación que experimentan las biomoléculas, en particular las proteínas, tras unirse con otras biomoléculas y/o pequeños ligandos. En el caso que nos ocupa, el transportador de Leucina LeuT, sufre un cambio de conformación tras la unión a LeuT, que se puede interpretar bien como “inducido” por la unión del aminoácido Leucina, es lo que se conoce como la *hipótesis de ajuste inducido* o *induced fit* [8], o alternativamente como que la unión del aminoácido Leucina estabiliza una conformación más “cerrada” (en azul en la Figura 18.8), distinta de la conformación libre (en rojo), que se conoce como la hipótesis de *selección conformacional* o *conformational selection* [9, 10]<sup>5</sup>. En este ejemplo, los dominios N y C terminal experimentan un movimiento de tipo bisagra hacia la cavidad de unión a la leucina. Ambas conformaciones tal y como aparecen superpuestas en la Figura 18.8 (superponiendo como cuerpo rígido el dominio mayor, C-terminal y aplicando la rotación-traslación a toda la molécula en la conformación que une leucina) se encuentran a un RMSD de 14.84 Å. Podemos utilizar la proyección del cambio conformacional en el espacio de los modos normales para ver cuánto contribuye cada modo normal a dicho cambio.



**Figura 18.8:** Superposición de las estructuras tridimensionales para LeuT en su conformación libre (rojo) y en su conformación con Leucina unida (azul oscuro). El aminoácido Leucina libre se muestra en la representación de bastones. Esta figura se ha hecho con la representación ‘cartoons’ del programa Pymol[5].

En principio esperaríamos que para este tipo de *movimientos “bisagra”* de gran amplitud, los modos de más baja frecuencia sean los que más contribuyan (dado que como habíamos visto en la Subsección 18.1.2, éstos contribuyen más a las fluctuaciones térmicas). La contribución de cada modo al cambio de conformación se calcula a través del cuadrado de la proyección del vector que separa ambas conformaciones (tras superponer la conformación ligada a la libre como cuerpo rígido, sin restringir ningún dominio, al contrario que en la Figura 18.8) sobre el vector del modo normal correspondiente, o lo que es lo mismo al coseno al cuadrado entre ambos vectores (Ecuación 18.50).

$$c_{\alpha}^2 = \frac{(\Delta r^{obs} v^{\alpha})^2}{(\Delta r^{obs})^2} \quad (18.50)$$

---

<sup>5</sup>Lo que realmente determina que un cambio de conformación sea induced fit o conformational selection es si la barrera energética del cambio de conformación es más elevada que la de unión a ligando a la conformación libre. Esto se puede determinar mediante experimentos de cinética de NMR y citometría de flujo, ambas técnicas fuera del alcance de este libro.

Para llevar a cabo los cálculos correspondientes con las herramientas ProDy, necesitamos efectuar ciertas operaciones previas. Primeramente se lee el fichero PDB que contiene las coordenadas para la conformación unida a leucina, y se crea una selección que contenga todos los átomos para la cadena A (existen 4 monómeros en la subunidad asimétrica para ‘1usk’), a excepción del ligando Leucina (residuo 1347), para que ambas selecciones tengan el mismo número de átomos

```
>>> leuT_b=parsePDB('1usk')
>>> leuT_b=leuT_b.select('protein and chain A and not resnum 1347')
```

Ahora llevamos a acabo la superposición de cuerpo rígido de la cadena A para la conformación unida sobre la cadena A de la conformación libre (análoga a la mostrada en la Figura 18.8):

```
>>> leuT_b, t=superpose(leuT_b, leuT)
```

En este caso la función superpose devuelve el conjunto de átomos de la conformación ligada (‘leuT\_b’) una vez se le ha aplicado la matriz de rotación-traslación superponerla óptimamente a la conformación libre (‘leuT’), y dicha matriz de rotación traslación t. Dado que en el modelo ANM sólo tenemos componentes para los carbonos alfa, una vez transformada la estructura de la conformación ligada, creamos sendas selecciones de carbonos alfa para ambas conformaciones. A partir de estas selecciones calculamos el vector  $\Delta r^{obs}$ :

```
>>> catoms2=leuT_b.select('protein and name CA')
>>> catoms=leuT.select('protein and name CA')
>>> delta_r=calcDeformVector(catoms,catoms2)
```

Ahora utilizando las potentes librerías matemáticas de Python (‘numpy’ o ‘np’ en nuestro ejemplo) podemos proyectar el vector  $\Delta r^{obs}$  en los 20 primeros modos normales, que genera el vector de proyecciones ‘proj’:

```
>>> proj=(np.array(delta_r)*np.array(list(anm[:20])))
>>> proj/=float(abs(delta_r))
>>> proj=(proj*proj)
```

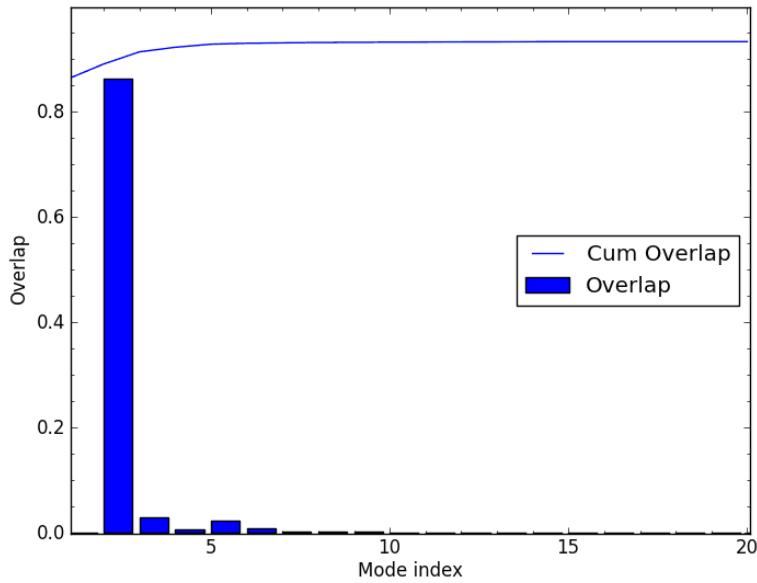
Además generamos: el vector de proyecciones ordenado en orden decreciente, otro vector que supone la suma cumulativa de los elementos del primero, y un tercer vector con los índices de los modos normales:

```
>>> proj_sorted=sorted(proj,reverse=True)
>>> proj_sorted_accum=np.add.accumulate(proj_sorted)
>>> index=range(1,21)
```

Finalmente representamos gráficamente la contribución cumulativa al cambio conformacional (de manera que se suma de mayor a menor las contribuciones a dicho cambio independientemente del índice del modo) de los 20 primeros modos normales de más baja frecuencia, en línea sólida de color azul, mientras que por cada modo se muestra en forma de barra azul su contribución a dicho cambio conformacional (Figura 18.9).

```
>>> plt.plot(index,proj_sorted_accum, 'b-')
>>> plt.bar(index,proj,color='b')
>>> plt.legend(('Cum Overlap', 'Overlap'), loc='center right')
>>> plt.xlim(1,20)
>>> plt.minorticks_on()
>>> plt.xlabel('Mode index')
>>> plt.ylabel('Overlap')
>>> plt.show()
```

Tal y como se aprecia, el segundo modo normal explica el 86 % del cambio conformacional. Aparte de este modo los otros 3 que más contribuyen son por orden el modo 3 (2.8 %), el modo 5 (2.3 %) y el modo 6 (0.9 %), contribuyendo en total los 4 a explicar un 92 % del cambio de conformación. Aunque en este caso el modo de más baja frecuencia no sea el que contribuya más y sea el segundo, podemos



**Figura 18.9:** Contribución al cambio conformacional de LeuT, conformación ligada a Leu con respecto a la conformación libre, de los 20 primeros modos normales según el modelo ANM.

concluir que el resultado está dentro de lo esperado para un movimiento de bisagra: pocos modos de baja frecuencia contribuyendo muy significativamente a dicho movimiento.

#### 18.4.4. Cálculo de una estructura deformada

Cálculo de una estructura deformada según uno o varios modos normales

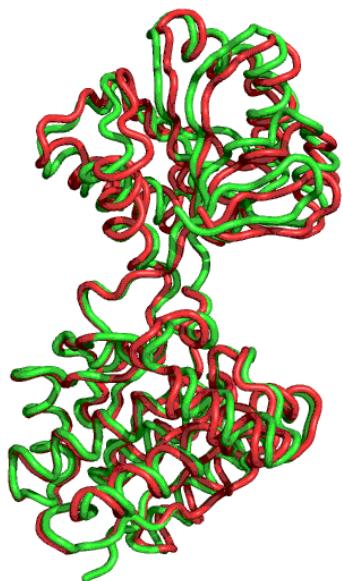
Dado que cada modo normal representa un movimiento oscilatorio según una dirección determinada, con una frecuencia característica, podemos generar una nueva estructura de proteína según un modo normal, dada una amplitud arbitraria. ProDy también permite hacer este cálculo para generar una estructura como resultado de deformar un modo (o combinación de modos) hasta una determinada separación de la estructura del equilibrio que se indica como RMSD. La manera como hacerlo sería:

```
>>> catoms_copy=catoms.copy()
>>> deformAtoms(catoms_copy,anm[0],rmsd = 2.0)
>>> writePDB('1leuT_deformed.pdb',catoms_copy)
```

En primer lugar se hace una copia del subconjunto de átomos carbono alfa creado anteriormente. Sobre este nuevo conjunto se guardan las coordenadas resultantes de deformar el primer modo normal (de índice 0) hasta producir una separación de 2.0 Ångstrom respecto a la estructura del transportador de Leucina en equilibrio. El resultado de dicha deformación se escribe en un fichero con formato PDB llamado ‘1leuT\_deformed.pdb’ que podemos visualizar con el programa Pymol [5] tal y como se muestra en la Figura 18.10.

De igual modo se podría haber generado una estructura de proteína utilizando una combinación de modos normales, p. ej. el primero y el segundo pesados por las fluctuaciones cuadráticas de cada modo:

```
>>> deformAtoms(catoms_copy,anm[0] * anm[0].getVariance()**0.5 +
               anm[1] * anm[1].getVariance()**0.5,rmsd = 2.0)
```



**Figura 18.10:** La estructura del transportador de Leucina en la posición de equilibrio se muestra en verde y la misma estructura perturbada según el primer modo normal hasta una separación de 2.0 Å (estructura en rojo). Ambas estructuras sólo contienen carbonos alfa, que están contenidas en el subconjunto de átomos 'calpha\_copy'.



## 18.5. Bibliografía

- [1] A. R. Atilgan, S. R. Durell, R. L. Jernigan, M. C. Demirel, O. Keskin, and I. Bahar. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophys J*, 80(1):505–15, 2001.
- [2] I. Bahar, A. R. Atilgan, and B. Erman. Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential. *Fold Des*, 2(3):173–81, 1997.
- [3] I. Bahar, T. R. Lezon, L. W. Yang, and E. Eyal. Global dynamics of proteins: bridging between structure and function. *Annu Rev Biophys*, 39:23–42, 2010.
- [4] A. Bakan, L. M. Meireles, and I. Bahar. Prody: protein dynamics inferred from theory and experiments. *Bioinformatics*, 27(11):1575–7, 2011.
- [5] W. DeLano. The pymol molecuar graphics system, 2002.
- [6] C. Eckart. Some studies concerning rotating axes and polyatomic molecules. *Physical Review*, 47:552–558, 1935.
- [7] H. Goldstein, C. P. Poole, and J. L. Safko. *Classical Mechanics*. Addison Wesley, 3rd edition, 2001.
- [8] D. E. Koshland. Application of a theory of enzyme specificity to protein synthesis. *Proc Natl Acad Sci U S A*, 44(2):98–104, 1958.
- [9] J. Monod, J. Wyman, and J. P. Changeux. On the nature of allosteric transitions: a plausible model. *J Mol Biol*, 12:88–118, 1965.
- [10] M. M. Rubin and J. P. Changeux. On the nature of allosteric transitions: implications of non-exclusive ligand binding. *J Mol Biol*, 21(2):265–74, 1966.
- [11] T. Schlick. *Molecular Simulations. An interdisciplinary guide*. Interdisciplinary Mathematics. Mathematical Biology. Springer-Verlag New York Inc., New York, 1 edition edition, 2002.
- [12] K. R. Symon. *Motion of a Particle in One dimension*, chapter 2, pages 21–71. Addison-Wesley series on Physics. Addison-Wesley, 3rd edition, 1971.
- [13] M. M. Tirion. Large amplitude elastic motions in proteins from a single-parameter, atomic analysis. *Physical Review Letters*, 77(9):1905–1908, 1996.



## Parte VI

# Biología de sistemas



# Capítulo 19

## Biología de sistemas

*Raúl Guantes, Jacobo Aguirre y Djordje Bajic*

### 19.1. Introducción

La biología molecular y celular tradicional se ha centrado en el estudio detallado de los componentes que constituyen los organismos vivos, por ejemplo identificando un gen afectado por un factor transcripcional o un tipo de neurona especializada en percepción de movimiento. *Las técnicas experimentales en la actualidad permiten obtener información de los seres vivos a escala global*, midiendo múltiples componentes simultáneamente en un organismo o identificando interacciones entre todos ellos. Estos componentes, ya sean proteínas o factores de transcripción en una célula, neuronas de un sistema sensorial o especies de un nicho ecológico, *se pueden representar por nodos conectados por diversas interacciones formando una red*. Incluso en los organismos más sencillos, estas redes constan de cientos de elementos conectados en una compleja estructura de interacciones. Algunas de las primeras *redes biológicas reconstruidas* de forma completa en organismos modelo son:

- La red neuronal de *Caenorhabditis elegans*, en la que se han identificado cada una de las sinapsis de las 393 neuronas que componen su sistema nervioso.<sup>1</sup>
- La red regulatoria de la cepa K12 de *Escherichia coli* [43], con información detallada de los factores de transcripción (alrededor de 200) y sus interacciones, así como de los promotores, regulones y operadores que intervienen en la regulación del genoma de *E.coli*.<sup>2</sup>
- La red metabólica de la misma cepa de *E.coli*, que comprende 1366 genes, 2251 reacciones metabólicas y 1136 metabolitos únicos [106].

Estos ejemplos dan idea de la dificultad de entender el funcionamiento de los organismos a escala global usando las herramientas moleculares tradicionales, y ponen de relieve la necesidad de nuevas aproximaciones teóricas y computacionales si queremos tener una visión de *sistemas* de los procesos biológicos. La *biología de sistemas* es precisamente la disciplina de la biología que estudia el comportamiento y función de los sistemas biológicos complejos a partir de su estructura y organización, teniendo en cuenta el papel que juegan las diferentes partes en el contexto global. En este sentido, es importante remarcar que en un sistema complejo, como son los organismos o redes biológicas, el resultado no es igual a la

<sup>1</sup>WormAtlas 1.0. <http://www.wormatlas.org/index.html>

<sup>2</sup>Regulon DB Database. <http://regulondb.ccg.unam.mx>

suma de las partes, sino que surgen propiedades nuevas (propiedades *emergentes*) como consecuencia del alto número de componentes y la no linealidad de sus interacciones.

Una aproximación posible al estudio de las redes biológicas consiste en explotar su abstracción matemática como *conjunto de nodos e interacciones* y utilizar técnicas de *teoría de grafos* (cuantificando diferentes propiedades de la conectividad de la red) para obtener información general de la estructura y su implicación en el funcionamiento global. Así, podemos estudiar propiedades estructurales como asortatividad, modularidad o división en comunidades, e investigar su papel en características importantes de la red como robustez o resistencia (frente a desaparición de componentes de la red o 'invasión' por nuevos elementos), plasticidad (respecto a mutaciones o cambios de conexiones entre elementos), etc. En la primera parte del capítulo proporcionaremos una introducción a la teoría de redes complejas, poniendo ejemplos de aplicaciones a redes biológicas concretas.

Los *grafos* son una herramienta útil para condensar diversas relaciones entre muchos componentes y a través de múltiples organismos, pero al no describir de forma detallada nodos ni conexiones no dan información sobre cambios en niveles de actividad de los nodos y por tanto su interpretación desde el punto de vista de la función de la red es limitada. Por tanto, debemos asignar 'números a las flechas', es decir, *modelizar matemáticamente los nodos y sus interacciones* teniendo en cuenta sus características biológicas. Obviamente esta descripción matemática depende del tipo de componentes de la red: si son neuronas y estamos interesados en la actividad eléctrica local, debemos describir cómo se transmite el potencial de membrana entre diferentes neuronas. Si son factores de transcripción que interaccionan con genes, debemos pensar en las reacciones bioquímicas que dan lugar a expresión de proteína. En este capítulo nos centraremos en las redes moleculares que controlan el funcionamiento celular, es decir en *redes de regulación/señalización* y en *redes metabólicas*, introduciendo las herramientas teóricas necesarias para su estudio cuantitativo.

Es importante señalar que la descripción matemática se puede dar a varios niveles de complejidad. Por ejemplo, para modelizar una red de regulación podemos representar las interacciones por puertas lógicas y los nodos por variables binarias que indican la actividad del gen (1 si está expresado y 0 si se encuentra inactivo). En este tipo de *modelos*, llamados *lógicos* o *booleanos*, no se requiere un conocimiento detallado de las interacciones bioquímicas, pero sólo describen patrones de expresión "digitales" (qué gen se puede activar en respuesta a una determinada señal) sin proporcionar información sobre los niveles de expresión o la dinámica de la red. Si deseamos conocer esto en más detalle, debemos recurrir a los *modelos cinéticos*, que se basan en una descripción físico-química (la ley de acción de masas) de las interacciones entre genes y proteínas. Estos modelos tienen en cuenta diferentes fuerzas y rangos de las interacciones y dan información sobre la dinámica de la red. Sin embargo, necesitan información bioquímica más detallada, que a veces es difícil conocer experimentalmente. En redes que constan de muchos componentes moleculares, como las redes metabólicas, en las que se conocen los nodos y sus interacciones pero se desconocen la mayor parte de los parámetros bioquímicos, se suele usar una aproximación intermedia (*modelos basados en restricciones*) en la que los parámetros desconocidos se sustituyen por una serie de restricciones realistas (por ejemplo, rangos de valores o imposición de balance de masas). Por último, las técnicas experimentales de célula y molécula individual han puesto de manifiesto el importante papel de las *fluctuaciones* o "ruido" en los niveles de expresión de una misma proteína en células de una población clonal. Si queremos entender el origen de estas fluctuaciones necesitamos una descripción física y probabilista de las interacciones bioquímicas, que también discutiremos en una sección del capítulo.

La modelización matemática de una red completa como los ejemplos descritos arriba es una tarea rara vez realizable, incluso en la situación ideal de conocer en detalle los parámetros que caracterizan las interacciones. Si estamos interesados en un proceso biológico determinado (por ejemplo, la división celular) una simplificación habitual consiste en tratar de identificar los componentes clave involucrados en

el proceso y aislarlos del resto de la red, suponiendo que son los responsables principales de esa función específica. Análisis topológicos de las redes biológicas, además, han puesto de manifiesto la existencia de determinadas *subredes* o estructuras que aparecen de forma recurrente y a las que se ha denominado *motivos de red*. Imitando el antiguo paradigma de que la “forma” determina la “función” en Biología, se ha propuesto que dichos motivos son especialmente abundantes porque desempeñan funciones clave dentro de la red, porque son elementos versátiles o porque constituyen entidades funcionales mínimas (del mismo modo que diferentes dispositivos electrónicos sencillos conectados apropiadamente conforman un ordenador capaz de funciones altamente complejas y especializadas). Abordaremos el análisis de algunos de estos motivos de red encontrados en redes de regulación, y discutiremos su posible relevancia en algunos procesos biológicos como la toma de decisiones celulares o la adaptación al nivel de estímulo.

Por último, estudiaremos algunas de las propiedades que surgen en los sistemas biológicos debidas a su estructura *global*. Entre ellas, una de las más interesantes es la *robustez fenotípica* frente a cambios ambientales, mutaciones o fluctuaciones en los componentes.

## 19.2. Introducción a las redes complejas.

### 19.2.1. Definición de red compleja y conceptos básicos

Un *grafo* o *red* ( $\mathcal{G}$ ) consiste en un grupo de *nodos* ( $N$ ) conectados mediante un conjunto de *enlaces* ( $L$ ). Una *red compleja* es una red con una estructura no trivial, cuyos patrones de conexión ni son completamente regulares ni totalmente aleatorios. Los nodos también son conocidos como *sistema* en física, *actor* en sociología o *vértice* en matemáticas, mientras que al enlace entre dos elementos se le conoce como *unión* o *link* en física, *relación* en sociología o *arista* en matemáticas.

La relación entre los nodos puede ser simétrica (dando lugar a las *redes no dirigidas*) o asimétrica (dando lugar a las *redes dirigidas*). Los enlaces no dirigidos se representan por líneas y los dirigidos por flechas. La dirección de los enlaces es crucial en los procesos dinámicos que ocurren en la red, como por ejemplo en la transmisión de información. Algunos ejemplos de redes no dirigidas son la red de routers, la red eléctrica, redes de colaboración entre personas, Facebook, etc. Por otra parte, Internet, Twitter, las redes tróficas, las redes de e-mails o de llamadas telefónicas son redes dirigidas.

La capacidad o intensidad de la interacción entre los nodos se puede plasmar en el grafo asociando un *peso* a cada enlace. Las *redes pesadas* son aquellas cuyos enlaces muestran diversidad de pesos y las *redes no pesadas* son aquellas cuyos enlaces son todos igualmente importantes. Otra vez, el peso de los enlaces es crucial en los procesos dinámicos que ocurren en la red. Ejemplos de redes pesadas son las redes tróficas, las redes de llamadas telefónicas, las redes de emails o las redes de relaciones sexuales, y ejemplos de redes no pesadas son Internet, Facebook, la red de citas científicas, etc...

Las *redes multipartitas* son redes con dos o más tipos de nodos, donde los enlaces sólo unen nodos de distinto tipo. Un ejemplo de red *bipartita* sería la red de alergias de una región dada [109], donde el conjunto A de nodos está formado por pacientes y el B por alérgenos susceptibles de provocar alergias. Cada paciente está conectado a los alérgenos que le producen reacción. Las redes bipartitas se pueden proyectar en redes no bipartitas, donde sólo existe un tipo de nodo, y que en general son más fáciles de analizar. En el caso de las redes de alergias antes citado, se podrían obtener dos redes proyectadas: (i) la red de pacientes, donde los nodos son pacientes, los enlaces unen pacientes que coinciden al menos en una alergia, y donde el peso de dichos enlaces es proporcional al número de alergias comunes sufridas por los pacientes conectados; y (ii) la red de alergias, donde los nodos representan alérgenos, los enlaces

unen alérgenos que provocan alergia compartida en al menos un paciente, y donde el peso de dichos enlaces es proporcional a la similitud entre el conjunto de pacientes sensibles a cada alérgeno.

En función de que las redes se mantengan constantes en el tiempo o evolucionen con él, se clasifican en *estáticas* y *dinámicas*. El estudio de las redes dinámicas y su evolución con el tiempo es una línea abierta y muy fructífera de la teoría de redes [62]. En particular, interesan dos cuestiones: (i) cuáles son las reglas que condicionan su evolución, y (ii) cuáles son las consecuencias de la estructura en los procesos que ocurren en la red (difusión de información, etc.)

Todas las redes anteriores se pueden describir de manera matricial, dado un conjunto de  $N$  nodos y  $L$  enlaces entre ellos. La *matriz de adyacencia*  $A$  asociada a una red  $\mathcal{A}$  viene definida por tener elementos  $A_{ij} = 1$  si existe un enlace entre los nodos  $i$  y  $j$ , y  $A_{ij} = 0$  en caso contrario. Esta matriz es simétrica si la red es no dirigida, y no simétrica si es dirigida.

La *matriz de pesos*  $W$  asociada a una red  $\mathcal{A}$  nos proporciona los pesos de las conexiones de la red. Sus elementos  $W_{ij}$  son iguales al peso del enlace entre los nodos  $i$  y  $j$  si éste existe, y  $W_{ij} = 0$  en caso contrario. Obviamente, esta matriz sólo es útil cuando la red es pesada.

### 19.2.2. Propiedades de las redes complejas

A pesar de que existen redes de muchos tipos, que a su vez pueden provenir de sistemas muy diferentes (neuronas, ordenadores, individuos, etc...), la mayoría comparte muchas propiedades.

#### Centralidad

La “*centralidad*”, como su nombre indica, es una medida que diferencia a los nodos de acuerdo a su influencia en una red. Existen varias cantidades que miden dicha influencia, siendo algunas de ellas locales (como el grado) y otras globales (como la *betweenness* o la centralidad de autovector).

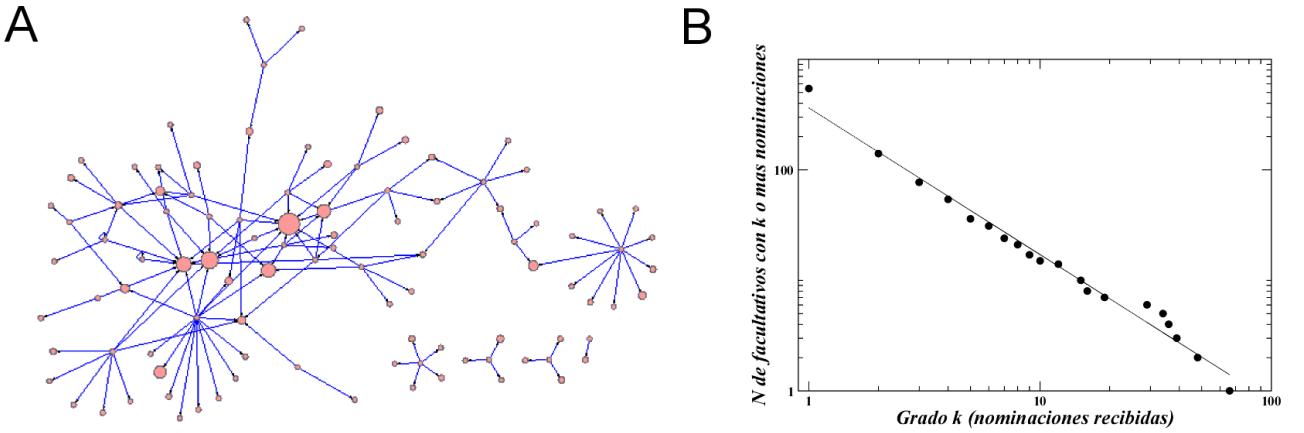
#### Grado y distribución de grado

El “*grado*”  $k_i$  de un nodo  $i$  es el número de conexiones (o vecinos) que tiene ese nodo. En el caso de redes dirigidas, el grado externo del nodo  $i$  es el número de conexiones que surgen de dicho nodo, mientras que el grado interno del nodo  $i$  es el número de conexiones que acaban en ese nodo. En dichas redes dirigidas, la suma del grado externo y el grado interno de un nodo es igual a su grado.

La *distribución de grado*  $p(k)$  de una red es la probabilidad de encontrar al azar un nodo de grado  $k$  en dicha red. A su vez, la *distribución acumulada*  $p_c(k)$  de una red nos indica la probabilidad de encontrar en ella y al azar un nodo de grado  $k$  o mayor. La red más simple que podemos encontrarnos es la red regular, donde todos los nodos tienen el mismo grado  $K$  y por lo tanto su distribución es  $p(k) = 1$  si  $k = K$  y  $p(k) = 0$  para el resto de  $k$ .

Sin embargo, hay dos tipos de distribuciones que aparecen con mayor frecuencia en las redes reales, y que son características de dos tipos de redes especialmente importantes, la aleatoria Erdős-Rényi y la libre de escala. Una *red aleatoria* se define como un grafo que está generado por algún tipo de proceso aleatorio. Las redes aleatorias más utilizadas son las de tipo *Erdős-Rényi* (ER), formadas por un número  $N$  de nodos que tienen todos la misma probabilidad  $p$  de estar conectados entre sí. En este tipo de redes, todos los nodos tienen un grado relativamente similar y por lo tanto no se suelen encontrar nodos especialmente conectados. La distribución exponencial  $p_c(k) \sim e^{-k}$  es típica en estos

casos. A su vez, la *distribución libre de escala* (o *scale-free* en su terminología inglesa) sigue una ley de potencia tal que  $p_c(k) \sim k^{-\alpha}$ , donde  $\alpha > 0$ . En este segundo caso, el decaimiento de la distribución de grado es muchísimo más lento que en las redes aleatorias ER, lo que significa que contienen algunos nodos extremadamente conectados, usualmente conocidos como *hubs*. Se conoce a estas redes con el término de libres de escala debido precisamente a que su enorme heterogeneidad de grado hace que no tengan una escala asociada (véase en la Figura 19.1 un ejemplo real de red social con distribución de grado libre de escala). El último tipo fundamental de red compleja, la red *small world* o de *mundo pequeño*, es aquella con bajo camino medio y alto coeficiente de clustering. Los conceptos de *clustering* y *distancia* en una red los definiremos más adelante en este capítulo.



**Figura 19.1:** Red de influencia entre médicos españoles que comparten especialidad. A: Representación gráfica de la red, las flechas apuntan hacia los colegas que cada uno de ellos tiene como referente profesional con respecto a una determinada patología. El número de flechas que llegan a cada nodo es su grado (en este caso el número de nominaciones), y el tamaño de cada nodo es proporcional a dicho grado. Los nodos amarillos representan médicos líderes en su campo. Al igual que otras muchas redes sociales, la red de influencia entre médicos españoles muestra una distribución libre de escala. B: Distribución de una red de varios miles de médicos donde unos pocos *hubs* reciben un altísimo número de nominaciones, mientras que la inmensa mayoría de la comunidad recibe muy pocas o ninguna. En este caso particular,  $p_c(k) \sim k^{-1.3}$ . Cortesía de Merck, Sharp & Dohme España.

## Otras medidas de la centralidad

El grado  $k$  de un nodo es una primera indicación de su centralidad, ya que es intuitivamente razonable suponer que los nodos con un alto grado serán atravesados por una cantidad proporcionalmente mayor de los caminos más cortos. Sin embargo, el grado es una medida excesivamente local, ya que, entre otros, no tiene en cuenta la influencia que sobre un nodo tienen sus vecinos. Para superar esta restricción, y como ya se ha comentado, la centralidad también se puede estimar a través de diferentes cantidades que no son locales, tales como la “*betweenness*” y la “*centralidad del autovector*” [23].

La definición de *betweenness*  $B(i)$  de un nodo  $i$  está dada por:

$$B(i) = \frac{1}{2} \sum_{j,k} \frac{g_{jik}}{g_{jk}} \quad (19.1)$$

donde  $g_{jk}$  es el número total de caminos más cortos entre los nodos  $j$  y  $k$ , y  $g_{jik}$  es el número de caminos más cortos entre los nodos  $j$  y  $k$  que pasan a través del nodo  $i$ . De forma cualitativa, se puede

decir que la *betweenness* de un nodo  $B(i)$  cuantifica la probabilidad de que el nodo  $i$  represente un paso intermedio en el camino más corto entre dos nodos elegidos aleatoriamente en la red.

La *centralidad de autovector*  $v_1(i)$  asociada al nodo  $i$  viene dada por el autovector por la derecha asociado al máximo autovalor  $\lambda_1$  de la matriz de adyacencia  $A$  [23]. Por lo tanto, es una medida especialmente interesante en redes sobre las que se va a estudiar la difusión de una población de forma aleatoria, porque según esta definición a cada nodo se le asigna un valor muy relacionado con la fracción de dicha población que se mantiene en dicho nodo transcurrido un tiempo suficientemente largo (Véase Subsección 14.5.3 para más información acerca de esta cuestión).

### **Coeficiente de agrupamiento o de clustering**

La densidad local de enlaces se mide por el *coeficiente de agrupamiento*  $C$ , más conocido por su término inglés “*clustering*”, que se define para cada nodo  $i$  como la probabilidad de que dos de sus vecinos estén conectados:

$$C_i = \frac{\text{número de pares conectados de vecinos de } i}{\text{número de pares de vecinos de } i} \quad (19.2)$$

donde el denominador es igual a  $\frac{1}{2}k_i(k_i - 1)$ , siendo  $k_i$  el grado del nodo  $i$ . El coeficiente de *clustering*  $C$  cuantifica la cantidad de enlaces existentes entre los vecinos de un determinado nodo. Es una medida de relación entre nodos que revela desviaciones con respecto a una relación aleatoria entre ellos [98]. Por lo general, los valores bajos de  $C$  corresponden a redes conectadas aleatoriamente, mientras que valores por encima de lo esperado para redes aleatorias suelen indicar la existencia de correlaciones locales.

El *clustering local* en función del grado  $C(k)$  se define como el promedio de  $C_i$  sobre todos los nodos con un determinado grado  $k$ :

$$C(k) = \langle C_i \rangle|_{k_i=k} \quad (19.3)$$

Por último, el *clustering* de una red se obtiene promediando sobre todos los nodos  $C = \langle C_i \rangle$ .

Veamos un ejemplo de cálculo del coeficiente de *clustering* en la red representada en la Figura 19.2. Aplicando la Ecuación 19.2 se obtiene  $C_{1,2,3,4,5,6,7} = \{0,2,1,0,0,0,1,1\}$ , y por lo tanto el *clustering* de la red es  $C = \langle C_i \rangle = 3,2/7 = 0,46$ .

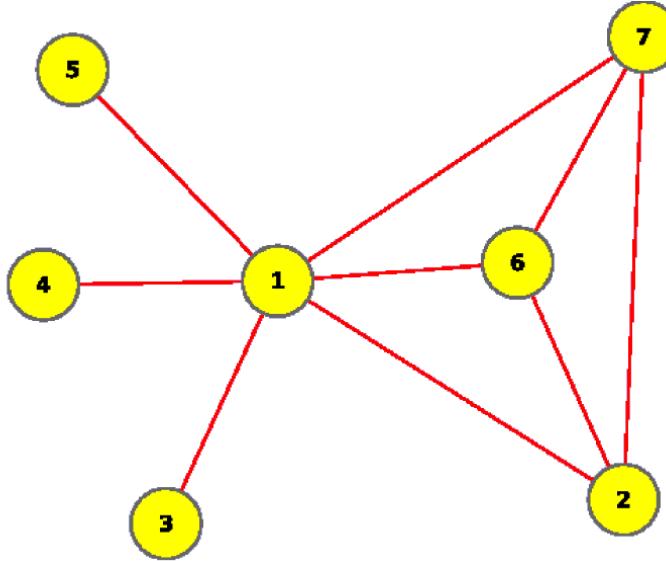
### **Distancias sobre una red**

El “*camino más corto*”  $d_{ij}$  entre los nodos  $i$  y  $j$  de una red es el número de pasos mínimo que hay que dar sobre la red para ir del nodo  $i$  al  $j$ . Si la red no es dirigida, entonces  $d_{ij} = d_{ji}$ . A su vez, el “*camino medio*”  $\langle d \rangle$  de una red se calcula como el promedio de los caminos más cortos  $d_{ij}$  entre todos los pares de nodos  $i, j$  que pertenecen a dicha red, de forma que

$$\langle d \rangle = \frac{\sum_{i,j} d_{ij}}{N(N-1)}. \quad (19.4)$$

Finalmente, el *diámetro de la red* es la máxima separación que existe entre dos nodos de la red,  $D = \max(d_{ij})$ .

Realicemos el cálculo del camino medio y el diámetro en la red de la Figura 19.2. Mediante la Ecuación 19.4 obtenemos el camino medio:  $\langle d \rangle = 66/42 = 1,57$ , y el diámetro de la red (o máxima separación entre dos nodos) es 2.



**Figura 19.2:** Ejemplo de red compleja.

### El grado medio de los vecinos y la asortatividad

El *grado medio de los vecinos*  $k_{nn,i}$  es una cantidad local que mide el grado medio de los vecinos de un nodo  $i$ . Por lo general se calcula en función del grado  $k$ ,

$$k_{nn}(k) = \sum_{k=0}^{\infty} k' p(k'|k) \quad (19.5)$$

donde  $p(k'|k)$  es la fracción de enlaces que están conectados a un nodo de grado  $k$  cuyos otros extremos están unidos a un nodo de grado  $k'$ . La variación de  $k_{nn}(k)$  con  $k$  está relacionada con la “*asortatividad*” de la red [98], lo que indica la tendencia de un nodo de grado  $k$  a conectarse con un nodo del mismo  $k$ . Cuando  $k_{nn}(k)$  es una función creciente, la red es *asortativa* y los nodos más conectados son propensos a estar vinculados a otros nodos altamente conectados. Si la función  $k_{nn}(k)$  es decreciente, la red es *disortativa* e indica que los *hubs* de la red están principalmente unidos a nodos escasamente conectados. La asortatividad se puede cuantificar por el *coeficiente de correlación grado-grado*  $r$ , que es el coeficiente de correlación de Pearson para los grados de los nodos en ambos extremos de un enlace:

$$r = \frac{\sum_i k_i^2 k_{nn,i} - (2L)^{-1} [\sum_i k_i^2]^2}{\sum_i k_i^3 - (2L)^{-1} [\sum_i k_i^2]^2}. \quad (19.6)$$

El parámetro  $r$  y la distribución  $k_{nn}(k)$  están estrechamente relacionados: una  $k_{nn}(k)$  monótona creciente corresponde a un valor positivo de  $r$ , y una  $k_{nn}(k)$  monótona decreciente corresponde a un valor negativo de  $r$ .

### Motivos

Los *motivos* de una red compleja (o *motifs*, en su terminología inglesa) son patrones de interacción entre un pequeño número de nodos tal que aparecen más frecuentemente de lo que se esperaría en una red aleatoria [92]. En cierta medida, se los puede entender como las piezas básicas sobre las que se

construyen las redes complejas. Aunque las propiedades que presentan los diferentes motivos no son únicas[65], pueden ser considerados como entidades funcionales, y su estudio ha sido particularmente fructífero en el ámbito de las redes biológicas.

## Modularidad y división en comunidades

En ocasiones, las redes pueden mostrar regiones internas donde los nodos estén más relacionados entre sí que con el resto de los nodos de la red. Esto da lugar al estudio de las estructuras de las redes a diferentes escalas, y un campo abierto de la teoría de grafos actual es el desarrollo de algoritmos para reconocer y diferenciar las subredes, generalmente conocidas como *comunidades*, dentro de una red dada.

Para ello, se ha introducido la *modularidad*, definida como [101]:

$$Q = \sum_{i=1}^m (e_{ii}a_i^2) \quad (19.7)$$

donde  $m$  es el número de comunidades dentro de la red,  $e_{ii}$  es la fracción de enlaces en la red que conectan los nodos de la misma comunidad  $i$ , y  $a_i$  es la fracción de enlaces que tienen uno o dos extremos en el interior de la comunidad  $i$ . Nótese que cuanto mayor sea la fracción de enlaces dentro de cada comunidad (conocidos como enlaces internos), mayor será el valor de  $Q$ . De esta manera, la modularidad  $Q$  generalmente se toma como parámetro de referencia para encontrar las divisiones óptimas en comunidades basadas en el análisis topológico de las redes [42].

Un ejemplo de red muy modular es la red de tráfico aéreo. Los aeropuertos de cada país están muy conectados entre sí, pero solamente desde los aeropuertos más grandes se puede viajar a otros países. Por eso, los aeropuertos nacionales y sus interconexiones forman comunidades distintas dentro de la red mundial de tráfico aéreo. Otro ejemplo de red modular es Facebook: nuestros colegas del trabajo, por ejemplo, formarán una comunidad porque estarán casi todos conectados entre sí, mientras que los colegas del trabajo de nuestra pareja formarán una comunidad distinta (¡si no trabajamos juntos, claro está!) Ambas comunidades, por cierto, estarán conectadas por el enlace existente entre nuestra pareja y nosotros mismos.

La *detección de las comunidades de una red* es relevante porque nos permite reconocer patrones comunes a los miembros de cada comunidad y que los distingan del resto, identificar las necesidades de dichas comunidades, analizar el papel de cada nodo dentro de su comunidad o incluso trabajar sólo con la parte de la red que más nos interese. Sin embargo, la división de una red en comunidades tiene asociada algunas dificultades importantes. La primera y principal es que el número de posibles divisiones de una red crece extraordinariamente rápido con el número de nodos y de comunidades, por lo que para redes incluso de tamaño pequeño es imposible calcular la modularidad de todas las posibles divisiones y así elegir la óptima. Esta dificultad ha dado lugar a una extensa línea de trabajo computacional en la que múltiples algoritmos de división en comunidades se han desarrollado con la finalidad de encontrar, si no siempre la división mejor, al menos una suficientemente buena para alcanzar nuestros fines. Un algoritmo muy utilizado en la actualidad es el de *Extremal optimization* [32] por su alto rendimiento incluso con redes de gran tamaño. El algoritmo *Fast* no es muy potente pero sí especialmente rápido [100], el *Louvain* [22] está indicado para redes grandes, el *Walktrap* [114] está basado en la teoría de caminantes aleatorios (aprovechando que un caminante aleatorio tiende a quedarse atrapado en la parte densa de las redes que corresponde a cada comunidad), y el *Rosvall-Bergstrom* [122] hace uso de teoría de la información en su diseño. Muchos de estos algoritmos y otros varios están implementados en la

herramienta *Radatools*<sup>3</sup> de la Universitat Rovira i Virgili. Otra dificultad a la que hay que enfrentarse es el hecho de que en muchas ocasiones las diferentes comunidades pueden compartir algunos nodos, es decir, pueden *solapar* [79]. Finalmente, puede ocurrir que la red ofrezca diferentes divisiones en comunidades en función de la escala que nos interese analizar. Estas escalas intermedias se conocen como *mesoescalas*.

Para una interesante revisión de la división de redes en comunidades, conviene acudir a [42].

### 19.2.3. Breve descripción de las redes biológicas

La aplicación de la teoría de redes complejas a sistemas biológicos ha dado resultados fructíferos acerca de cómo la topología de la red se relaciona con los procesos dinámicos que ocurren en dicha red [4, 13, 152]. En las *redes de interacción proteína-proteína*, por ejemplo, los nodos representan proteínas y están conectados a través de un enlace no dirigido si ambas proteínas se unen para formar un compuesto más complejo [61]. Este tipo de redes forma una componente gigante con configuración de mundo pequeño (alto clustering y corto camino medio entre nodos) [51, 154] y, en algunos casos, con conectividad de tipo libre de escala [66, 90, 154]. Las redes con esta estructura son muy robustas ante fallos aleatorios y, al mismo tiempo, son capaces de propagar una perturbación a través de la red en unos pocos pasos [99]. En el caso de las *redes metabólicas*, los nodos representan metabolitos, reacciones o enzimas, y los enlaces en dichas redes son dirigidos. Al igual que en las redes de proteínas, la distribución de grado de estas redes es libre de escala [67, 138] y poseen estructura de mundo pequeño [150]. En las *redes de regulación genética*, los genes son los nodos de la red y los factores de transcripción (activadores o represores) definen los enlaces entre los nodos, que son dirigidos [60]. Una vez más, a pesar de ser redes de naturaleza totalmente distinta a las anteriores, el número de enlaces que salen de un determinado nodo tiene una distribución libre de escala [75, 83].

Todas las redes biológicas mencionadas son el resultado de procesos constructivos que preservan la funcionalidad de la red en todas las etapas, modificando el tamaño de las redes a través de su evolución, y optimizando diferentes rasgos biológicos. Estos procesos son esenciales para determinar las propiedades topológicas de las redes resultantes. En este sentido, su naturaleza es diferente al de las *redes neutrales de estructuras secundarias de RNA*, cuya topología característica es consecuencia directa del proceso de plegado [3]. Como se analiza en detalle en la Subsección 14.5.2, las propiedades locales de las redes neutrales de RNA están fijadas por la existencia de cuatro nucleótidos diferentes que forman la secuencia de RNA y por los motivos estructurales principales de la estructura secundaria (*stems* y *loops*).

Como ejemplo, la Tabla 19.1, obtenida de [3], muestra una *comparativa entre las propiedades topológicas de multitud de redes biológicas* clasificadas en 5 tipos distintos.

Tipo de red biológica	$p(k)$	$C(k)$	$k_{nn}(k)$	$r$
Redes neutrales de RNA	pico único [3]	$\sim k^{-1}$ [3]	$\sim k^\delta$ [3]	$r > 0$ [3]
Redes metabólicas	LE [90, 121, 150, 154]	$\sim k^{-1}$ [121]	CN	$r < 0$ [99]
Redes de proteínas	LE [51, 66, 81, 148]	$\sim k^{-2}$ [154]	CN [90] CP [8]	$r < 0$ [66] $r > 0$ [8]
Redes funcionales cerebrales	LE [34]	CP [27]	CP [27]	$r > 0$ [34]
Ecosistemas (redes tróficas)	LE [33, 93, 94] LET [33, 93] E [33]	—	—	$r < 0$ [63, 89]

<sup>3</sup>Radatools. [http://deim.urv.cat/\\$\sim\\$sgomez/radatools.php](http://deim.urv.cat/$\sim$sgomez/radatools.php)

**Tabla 19.1:** Propiedades topológicas de diversas redes biológicas. Las cantidades reflejadas son la distribución de grado  $p(k)$ , el clustering en función del grado  $C(k)$ , el grado medio de los vecinos  $k_{nn}(k)$ , y la assortatividad  $r$ . Las abreviaturas corresponden a distribución libre de escala  $\sim k^\gamma$  (LE), distribución libre de escala truncada  $\sim k^\gamma e^{-k/\xi}$  (LET), distribución exponencial  $\sim e^{-k/\xi}$  (E), correlación positiva (CP) y negativa (CN). (Véase la Tabla 14.2 para una comparativa entre la topología de las redes neutrales de RNA de longitud de secuencia  $l = 12$  y las redes aleatorias Erdős-Rényi y libres de escala Barabási-Albert).

### 19.3. Introducción al análisis de redes de regulación.

Dentro de las redes biológicas, una clase fundamental son las redes de interacción entre los distintos genes de una célula mediante las proteínas que codifican (que actúan como factores de transcripción de otros genes). Las *redes genéticas* o *redes de regulación* controlan, mediante la expresión de determinadas proteínas, todas las funciones celulares: desde la respuesta a determinados estímulos, hasta la diferenciación celular o la apoptosis. Un aspecto fundamental de estas redes es su *componente dinámico*: no todos los genes se activan en todo momento, sino que es importante la coordinación temporal de determinados grupos de genes[155], o la evolución temporal de la respuesta de un gen[6]. Por ello, es fundamental disponer de herramientas matemáticas que nos permitan describir cómo cambia la expresión de un gen o grupo de genes en función del tiempo y en función de la activación de otros genes que interactúen con éstos.

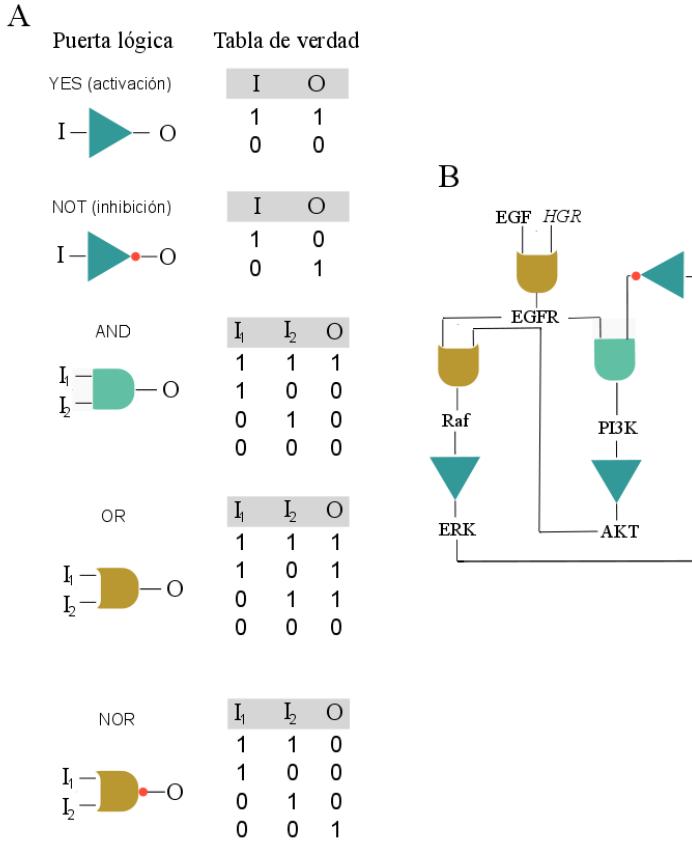
En esta sección describiremos diferentes aproximaciones matemáticas a la evolución temporal de redes de regulación, desde las más simplificadas (utilizando lógica booleana o discreta) hasta las más aproximadas a la dinámica real (modelos estocásticos).

#### 19.3.1. Modelos lógicos o booleanos

La primera aplicación de *métodos lógicos* o *booleanos* para modelar redes de interacción bioquímicas se atribuye a Kauffman [71], que usó lógica discreta para estudiar regulación genética. Para modelizar una red regulatoria necesitamos partir de su representación como un grafo, en el que los nodos representan genes o su producto (proteínas) y las conexiones interacciones entre ellas, bien físicas o correlaciones en perfiles de expresión genética. Las conexiones deben ser dirigidas (indicando qué nodo actúa como ‘*input*’ y cuál como ‘*output*’) y se debe especificar su signo. En *conexiones activadoras*, si el nodo *input* está activo su *output* también se activará. En *conexiones inhibidoras*, un *input* activo implica la desactivación del *output* (Figura 19.3A).

Cuando dos o más conexiones convergen en un mismo nodo, para calcular su actividad necesitamos saber cómo le afectan las actividades simultáneas de los diferentes *inputs*. Estas dependencias se especifican por ‘*puertas lógicas*’ o, en lenguaje de lógica Booleana, por ‘*tablas de verdad*’ (*truth tables*) que asignan los posibles estados del *output* para todas las posibles combinaciones de estados del *input* (Figura 19.3). Por ejemplo, si un nodo afectado por dos *inputs* sólo se activa cuando ambos *inputs* están activos, el nodo es equivalente a una puerta lógica ‘AND’. Con esta información, es posible calcular la respuesta de la red a un *input* dado y también las respuestas debidas a la inactivación de un nodo (por ejemplo, por una droga). El estado de cada nodo  $i$  a un tiempo  $t$ ,  $x_i(t)$ , viene especificado por una variable binaria ( $x_i(t) = 1$  si el nodo está activo,  $x_i(t) = 0$  si está inactivo).

La forma habitual de hacer evolucionar la red booleana consiste en *actualizar de forma sincrónica*



**Figura 19.3:** A. Puertas lógicas y tablas de verdad para un solo *input*  $I$  (YES/NOT, activación/inhibición) y para combinaciones de dos *inputs*  $I_1, I_2$  (OR/AND/NOR). B. Esquema de lógica booleana de las rutas de activación de los receptores del factor de crecimiento epidérmico (EGFR) en respuesta a dos estímulos diferentes: EGF y HGR (adaptado de la referencia [95]).

todos los nodos de la red en instantes de tiempo discretos, siguiendo las ‘*tablas de verdad*’. De forma alternativa, podemos sustituir las tablas de verdad de la lógica Booleana por funciones que produzcan una variable binaria dependiendo de si todos los *inputs* que activan un cierto nodo  $i$  superan un umbral  $\theta_i$  (lo que podría relacionarse con una fuerza determinada del promotor del gen  $i$ ) [80]. Formalmente:

$$x_i(t+1) = \begin{cases} 0, & \sum_j a_{ij}x_j(t) > \theta_i \\ 1, & \sum_j a_{ij}x_j(t) < \theta_i \\ x_i(t), & \sum_j a_{ij}x_j(t) = \theta_i \end{cases} \quad (19.8)$$

donde  $a_{ij}(t) = 1$  para una conexión activadora del nodo  $j$  al  $i$ ,  $a_{ij}(t) = -1$  para una conexión inhibidora, y  $a_{ij} = 0$  si no hay interacción entre ambos nodos. Se suele tomar el valor de umbral  $\theta_i = 0$ , pero otros convenios para umbral y las fuerzas de interacción  $a_{ij}$  son posibles.

A medida que se actualizan en el tiempo todos los nodos según las tablas de verdad o las reglas anteriores, la red convergerá eventualmente a un único estado estacionario en el tiempo, o a un ciclo periódico de estados, denominado *atractor*. Los atractores representan el patrón de expresión genética de la red regulatoria.

El método descrito es el más habitual para modelizar redes de muchos componentes, y se ha utilizado con éxito para identificar modos de regulación a partir de datos de *microarrays* [110] o para predecir

propiedades del ciclo celular en levadura [30, 80] (en este caso, estudiando la convergencia a atractores periódicos). La simplificación fundamental de este método, como ya apuntábamos arriba, consiste en no diferenciar valores absolutos de fuerzas de interacción, y no distinguir entre diferentes escalas temporales de las interacciones bioquímicas. Existen extensiones del método booleano que remedian en parte estos inconvenientes. Una de ellas es la *actualización asíncrona* de los nodos en el tiempo, actualizando antes unos nodos que otros, de forma que podemos establecer diferentes jerarquías de escalas temporales en la red [72]. Otra mejora del método, que es capaz de reproducir estados intermedios de actividad, consiste en utilizar técnicas de *lógica difusa* para describir interacciones entre nodos, que transforman reglas lógicas en relaciones continuas entre *inputs* y *outputs* [95].

### 19.3.2. Modelos cinéticos

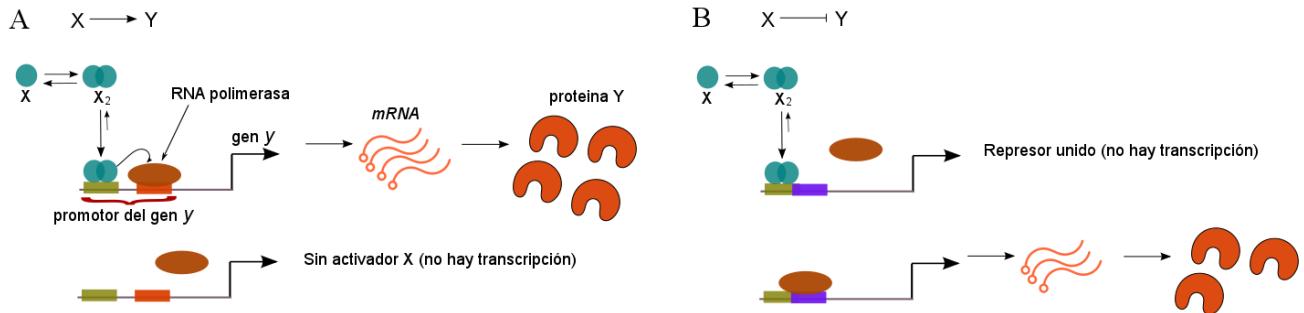
Los modelos booleanos suelen ser útiles cuando la red regulatoria tiene un tamaño apreciable, cuando no hay un conocimiento detallado de los parámetros bioquímicos (constantes de reacción, fuerzas de interacción o degradaciones) que determinan las interacciones entre los nodos, y cuando se está interesado en patrones de expresión globales (qué gen se activa en respuesta a una serie de *inputs*) y no en niveles absolutos de expresión o en la respuesta detallada en función del nivel de *input*. A menudo, sin embargo, es necesario describir la respuesta de la red de forma más cuantitativa. Los métodos de biología molecular y genómica actuales permiten conocer los niveles absolutos de expresión de mRNA y proteína de muchos genes, así como sus interacciones. Cuando se diseñan redes artificiales que realizan una función predeterminada usando técnicas de “*biología sintética*” [132], es necesario caracterizar en detalle la dependencia funcional de la red de los parámetros bioquímicos, ya que la respuesta de una misma red de nodos e interacciones puede depender críticamente de éstos.

La modelización más cuantitativa de la respuesta y la dinámica de una red regulatoria se suele hacer recurriendo a los llamados ‘modelos cinéticos’, que se basan en una descripción físico-química de las interacciones que tienen lugar entre los nodos de la red. El resultado final es un sistema de ecuaciones diferenciales ordinarias (ODEs, de sus siglas en inglés) para las concentraciones de cada una de las especies moleculares que representan los nodos, y que dan su evolución temporal hasta alcanzar eventualmente un estado de equilibrio o estacionario. En la siguiente sección ilustraremos cómo se construye un modelo cinético basándonos en unas pocas premisas elementales sobre las interacciones entre los nodos a escala molecular.

#### Los números sobre las flechas: la función de regulación

Consideremos una subred elemental en que una especie  $X$  actúa sobre otra  $Y$  (por ejemplo, dos proteínas, siendo  $X$  un factor transcripcional de  $Y$ ). Esta interacción puede ser de dos tipos: la producción de  $X$  puede a su vez favorecer la producción de  $Y$  (conexión activadora, representada como  $X \rightarrow Y$ ), o puede inhibir la producción de  $Y$  (conexión inhibidora,  $X \dashv Y$ ). Estas interacciones representan en realidad una serie de reacciones bioquímicas esquematizadas en la Figura 19.4, y que reflejan de forma simplificada el ‘dogma central’ en Biología: una proteína  $Y$  se produce a partir de un gen  $y$  en cuya zona promotora se une la polimerasa para iniciar su transcripción en mRNA, el cual se traduce a proteína en los ribosomas (Figura 19.4A). Si  $X$  es un factor transcripcional, generalmente ha de activarse antes de poder regular la transcripción de  $Y$  (usualmente por un inductor, un cambio de conformación debido a fosforilación, o por la unión de varios de sus monómeros). La proteína  $X$  activa, representada por su forma dimérica  $X_2$  en la Figura 19.4, puede unirse a la zona promotora del gen  $y$ . Dependiendo de que  $X$  sea un activador o inhibidor de  $Y$  su efecto en este momento será diferente: si  $X$  es un activador, Figura 19.4A, una vez unido favorecerá la unión de la polimerasa en la zona promotora, aumentando la

transcripción. Si es un inhibidor, Figura 19.4B, impedirá la unión de la polimerasa a la zona promotora, evitando la transcripción.



**Figura 19.4:** Reacciones elementales de regulación genética para un factor de transcripción que activa una proteína  $Y$  (A) o inhibe su expresión (B). A: El factor  $X$  se activa formando dímeros  $X_2$ . Éstos se unen a la zona promotora del gen  $y$  que tiene sitios específicos tanto para su unión (marcado en verde) como para la RNA polimerasa (marcado en naranja), que inicia la transcripción del mRNA. En este caso la función del activador es reclutar la RNA polimerasa y favorecer su unión al promotor. En ausencia de activador la polimerasa no tiene afinidad suficiente como para unirse al promotor del gen  $y$  y por tanto no hay transcripción (esquema inferior). B: Si  $X_2$  actúa como represor, su unión al promotor evita la interacción con la RNA polimerasa y por tanto se inhibe la transcripción. En este caso los dos sitios de unión (en morado y verde) solapan y la mayor afinidad de  $X_2$  por el promotor dificulta la unión de la polimerasa. En ausencia del represor la polimerasa no tiene impedimento estérico para unirse al promotor y se produce transcripción (esquema inferior).

Cada uno de estos procesos se puede representar por reacciones bioquímicas elementales.

- Supongamos que la forma activa del factor de transcripción  $X$  corresponde a un oligómero de  $n$  moléculas de  $X$  (muchos factores de transcripción se sabe que forman dímeros, tetrameros e incluso hexámeros/octómeros). Las *reacciones de asociación/disociación* del complejo activo  $X_n$  se pueden escribir como



donde  $k_a$  y  $k_d$  son las constantes de reacción para la asociación y disociación respectivamente.

- El factor activo  $X_n$  se une al promotor del gen  $y$ , denotado por la variable  $P_y$ . Las *reacciones de unión/desunión* son:



- Transcripción* del mRNA del gen  $y$ ,  $m_y$ , tanto desde el promotor libre como desde el promotor ocupado,  $P_y X_n$ :



donde  $\alpha$  y  $\beta$  representan las tasas o velocidades de reacción desde el promotor libre y el promotor ocupado respectivamente.

- *Traducción* del mRNA en proteína  $Y$  (condensando en un solo proceso todas las reacciones que tienen lugar en el ribosoma y maduración de la proteína):



- *Degradación* de mRNA y proteína (por RNAsas, proteasas o por dilución debida a crecimiento y división celular):



La evolución en el tiempo de cada una de las especies moleculares viene dada por una ecuación 'cinética' de la forma:

$$\frac{dx_i}{dt} = \text{velocidad de producción} - \text{velocidad de degradación} \quad (19.14)$$

donde  $x_i$  representa la *concentración* de la especie  $i$ . Tanto las velocidades de producción como degradación son proporcionales a los productos de las concentraciones de los reactivos, ya que toda reacción química es proporcional a la velocidad de colisión de las especies reactivas, que depende del número de moléculas en un volumen determinado. Esto es lo que se conoce como *ley de acción de masas*, y permite expresar los términos de producción y degradación en la Ecuación 19.14 en función de las variables de interés y las constantes de reacción. Por ejemplo, para la concentración de factor de transcripción inactivo  $X$  podemos escribir, fijándonos en las reacciones de asociación/disociación, Ecuación 19.9:

$$\frac{dX}{dt} = nk_d X_n - nk_a X^n \quad (19.15)$$

El primer término indica que tenemos una velocidad de producción de  $X$  proporcional a la cantidad de factor activo  $X_n$  que se disocia en  $n$  moléculas de  $X$ , y la constante de proporcionalidad es la constante de reacción  $k_d$ . El segundo término indica que reaccionan  $n$  moléculas idénticas de  $X$  (de ahí el exponente  $n$ ) para formar el complejo  $X_n$ . Es importante recalcar que a nivel cinético, las unidades de nuestras variables son concentraciones y por tanto las constantes de reacción son constantes macroscópicas. En la sección siguiente describiremos una formulación microscópica rigurosa del mismo sistema.

Podemos proceder de forma similar para obtener la *evolución temporal* del resto de las variables, fijándonos en las reacciones que producen una determinada especie y en aquellas reacciones que la eliminan. Para el conjunto de reacciones especificadas más arriba obtenemos:

$$\frac{dX_n}{dt} = k_a X^n + k_{off} P_y X_n - k_d X_n - k_{on} P_y \cdot X_n \quad (19.16)$$

$$\frac{dP_y}{dt} = k_{off} P_y X_n - k_{on} P_y \cdot X_n \quad (19.17)$$

$$\frac{dP_y X_n}{dt} = k_{on} P_y \cdot X_n - k_{off} P_y X_n \quad (19.18)$$

$$\frac{dm_y}{dt} = \alpha P_y + \beta P_y X_n - \delta_{my} m_y \quad (19.19)$$

$$\frac{dY}{dt} = s_y m_y - \delta_y Y \quad (19.20)$$

Nótese la diferencia en las ecuaciones 19.16-19.18 entre la variable  $P_y X_n$ , que representa la fracción de promotor ocupado, y el producto  $P_y \cdot X_n$  (fracción de promotor libre por cantidad de factor de transcripción activo). A pesar de que no hemos considerado los procesos de producción y degradación de la proteína inactiva  $X$  (suponiendo que siempre hay una cantidad determinada de este factor de transcripción, que consideramos como nuestro *input*), este modelo cinético de una red genética de dos componentes con una sola interacción, la red más sencilla posible, consta de 6 variables dinámicas (el factor de transcripción  $X$ , su forma activa  $X_n$ , los promotores libres y ocupados del gen  $y$ , el mRNA del gen  $y$  y la propia proteína  $Y$ ), y 10 parámetros: las 9 constantes de reacción de las ecuaciones 19.9-19.13 y  $n$ , el número de monómeros que constituyen la proteína activa  $X_n$ . Estos modelos se suelen *simplificar* recurriendo a una serie de aproximaciones biológicamente plausibles, de forma que al final cada uno de los nodos de la red genética quede representado por una sola variable dinámica y por un número reducido de parámetros. La primera simplificación consiste en tener en cuenta las *ligaduras* o cantidades conservadas en el tiempo. Por ejemplo, sumando las ecuaciones 19.17 y 19.18 vemos que:

$$\frac{d(P_y + P_y X_n)}{dt} = 0$$

lo que indica que

$$P_y + P_y X_n = P_y^T \quad (19.21)$$

donde  $P_y^T$  es una constante que denota el número de copias del promotor del gen  $y$ . Generalmente consideramos  $P_y^T = 1$ , es decir que el gen  $y$  está presente en una sola copia, aunque en plásmidos bacterianos las copias pueden llegar a varios cientos.

A parte de las ligaduras, se suele hacer uso extensivo de la separación de escalas temporales de los diferentes procesos biológicos implicados en regulación genética. En los descritos por las reacciones 19.9-19.13, hay algunos mucho más rápidos que otros. La Tabla 19.2 da una idea de los tiempos en que transcurren cada una de las reacciones bioquímicas en la bacteria *E. coli* (adaptada de la referencia [5]):

Proceso celular	Escala temporal
Activación/desactivación de un factor transcripcional	1 -100 $\mu$ s
Unión de un factor transcripcional activo a su promotor en el DNA	$\sim$ 1 s
Transcripción de un gen típico	$\sim$ 1 min (80 bp/s en fase exponencial)
Traducción de mRNA a proteína	$\sim$ 2 min (40 aa/s en fase exponencial)
Vida media de mRNA	$\sim$ 2-5 min
Vida media de proteína (por dilución debida a división celular)	$\sim$ 20-30 min en medio rico en nutrientes

**Tabla 19.2:** Escalas temporales de diferentes procesos de regulación en *E.coli*. aa/s: Aminoácidos por segundo. bp/s: Pares de base por segundo.

Como puede observarse, hay procesos muy rápidos (en la escala de los microsegundos al segundo) como la activación del factor transcripcional y su unión al promotor del gen, y otros más lentos, de al menos dos órdenes de magnitud mayores, como la transcripción y la traducción. Una buena aproximación consiste en considerar las reacciones rápidas en equilibrio respecto a las lentas (*aproximación* que se conoce con el nombre de *adiabática* en Física y *cuasi-estacionaria* en Biología), ya que en la escala de los minutos las reacciones rápidas han tenido tiempo de llegar al estado estacionario. Suponemos entonces que las variables rápidas, en este caso  $X_n$ ,  $P_y$  y  $P_yX_n$ , son constantes en el tiempo y por tanto sus derivadas temporales se anulan, lo que nos permite reducir el número de variables y ecuaciones únicamente a aquellas que describen la dinámica ‘lenta’. Por ejemplo, asumiendo que las reacciones de activación/desactivación del factor de transcripción (Ecuación 19.9) están en equilibrio podemos igualar la ecuación diferencial 19.16 a cero y obtenemos:

$$X_n = \frac{X^n}{K_{act}} \quad (19.22)$$

donde  $K_{act} \equiv k_d/k_a$  es la constante de equilibrio de las reacciones de activación/desactivación del factor transcripcional. De igual modo, suponiendo que la unión al promotor del gen  $y$  es un proceso en equilibrio e igualando la Ecuación 19.17 a cero expresamos la fracción de promotor ocupado  $P_yX_n$  como:

$$P_yX_n = P_y \cdot \frac{X^n}{K_{act}K_u} \quad (19.23)$$

donde  $K_u \equiv k_{off}/k_{on}$  es la constante de equilibrio de la unión/desunión del promotor dado por las reacciones (Ecuación 19.10), y donde hemos usado también la Ecuación 19.22.

Teniendo en cuenta la ligadura dada por la Ecuación 19.21 con  $P_y^T = 1$  y usando la ecuación anterior, podemos expresar la fracción de promotor libre  $P_y$  como:

$$P_y = \frac{1}{1 + X^n/K_x} \quad (19.24)$$

donde  $K_x \equiv K_{act}K_u$  engloba las dos constantes de equilibrio rápido anteriores. Por tanto, sustituyendo las expresiones 19.23 y 19.24 en la Ecuación 19.19 para la dinámica del mRNA, obtenemos un sistema de dos ecuaciones para las variables ‘lentas’ (mRNA y proteína) en función de únicamente de la especie  $X$  (nuestro *input*) de la forma:

$$\begin{aligned} \frac{dm_y}{dt} &= \frac{\alpha + \beta \cdot X^n/K_x}{1 + X^n/K_x} - \delta_{my}m_y \\ \frac{dY}{dt} &= s_y m_y - \delta_y Y. \end{aligned} \quad (19.25)$$

Una aproximación adicional que se emplea con frecuencia consiste en eliminar la dinámica del mRNA ( $\frac{dm_y}{dt} \simeq 0$ ), suponiendo que es bastante más rápida que la de proteína. Esta aproximación es válida si la vida media del mRNA es mucho menor que la de proteína, es decir  $\delta_{my} \gg \delta_y$  (las constantes de degradación se obtienen como  $\delta_i = \ln 2/\tau_i$ , donde  $\tau_i$  es la vida media de la especie  $i$ ). Para proteínas estables esta relación se cumple (ver Tabla 19.2), de forma que las interacciones inhibidoras  $X \dashv Y$  o activadoras  $X \rightarrow Y$  se pueden formalizar matemáticamente como una única ecuación de evolución temporal para la especie  $Y$  en función del *input*  $X$ :

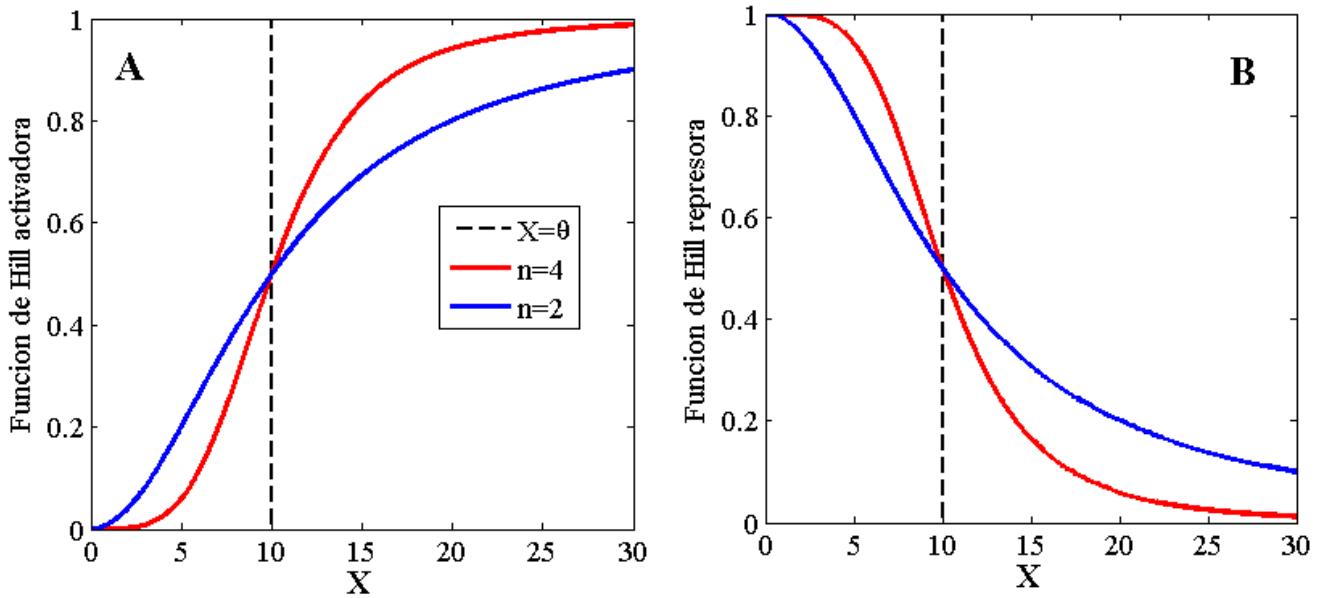
$$\frac{dY}{dt} = \gamma \frac{\alpha + \beta \cdot X^n/K_x}{1 + X^n/K_x} - \delta_y Y \quad (19.26)$$

donde  $\gamma \equiv s_y/\delta_{my}$  da el número de proteínas sintetizadas por mRNA. Este parámetro, también llamado ‘burst parameter’ (parámetro de ráfaga) jugará un papel importante cuando discutamos el ruido en expresión genética.

Cuando el factor de transcripción  $X$  es un activador,  $\beta > \alpha$ . Supongamos  $\alpha = 0$  (es decir, no hay expresión basal del gen  $y$  si su promotor no tiene unido el activador, como es el caso representado en la Figura 19.4A). El término de producción de  $Y$  es proporcional a:

$$Hill_{act}(X) \equiv \frac{(X/\theta)^n}{1 + (X/\theta)^n} \quad (19.27)$$

lo que se conoce como *función de Hill*. También se suele denominar *relación input/output* o *función respuesta*, ya que determina la cantidad en equilibrio de la proteína  $Y$  en función de su regulador  $X$ . Esta función de Hill de activación se caracteriza únicamente por dos parámetros: la constante  $\theta \equiv K_x^{1/n}$  o *umbral*, que determina el valor de *input*  $X$  en el que  $Y$  alcanza la mitad de su valor máximo, y el exponente  $n$  o *coeficiente de Hill* que indica la pendiente de la respuesta. La *función de Hill* (19.27) viene representada en la Figura 19.5A para  $\theta = 10$  y dos valores diferentes del coeficiente de Hill (recuérdese que este exponente representa biológicamente el grado de cooperatividad del factor transcripcional activo). Si el factor transcripcional  $X$  es un represor,  $\beta < \alpha$ . Supongamos  $\beta = 0$  (no hay



**Figura 19.5:** Funciones respuesta del gen  $y$  a un factor transcripcional. A:  $X$  es un activador que actúa como dímero ( $n = 2$ , curva azul), o tetrámero ( $n = 4$ , curva roja). La línea negra discontinua representa el valor umbral  $X = \theta$  para  $\theta = 10$ . B: Función de Hill cuando  $X$  actúa como represor con los mismos parámetros del panel A.

transcripción si el promotor del gen  $y$  está ocupado por el represor activo). El término de producción es en este caso proporcional a la función de Hill represora:

$$Hill_{rep}(X) \equiv \frac{1}{1 + (X/\theta)^n} \quad (19.28)$$

reproducida en la Figura 19.5B para dos valores diferentes de cooperatividad o coeficiente de Hill.

Estas funciones son ampliamente usadas para formalizar matemáticamente las interacciones regulatorias en redes genéticas. Aquí las hemos deducido detalladamente, aplicando la ley de acción de masas

a cada una de las reacciones involucradas y usando separación de escalas temporales, pero en modelos cinéticos se suelen utilizar directamente para representar y parametrizar la respuesta de un gen a diferentes factores transcripcionales, ajustando los parámetros  $\theta$  y  $n$  de cada una de las interacciones a resultados experimentales o usando valores biológicamente plausibles.

### 19.3.3. Modelos termodinámicos

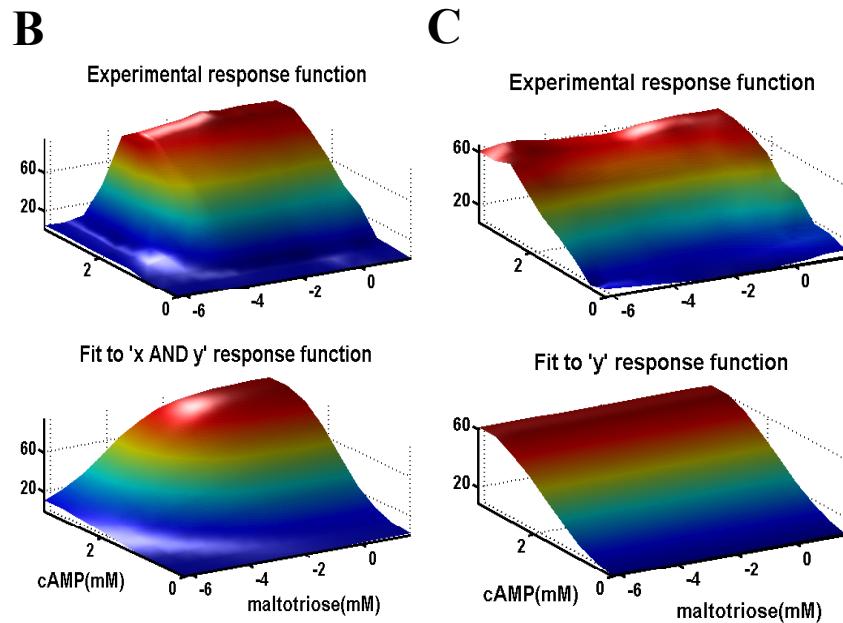
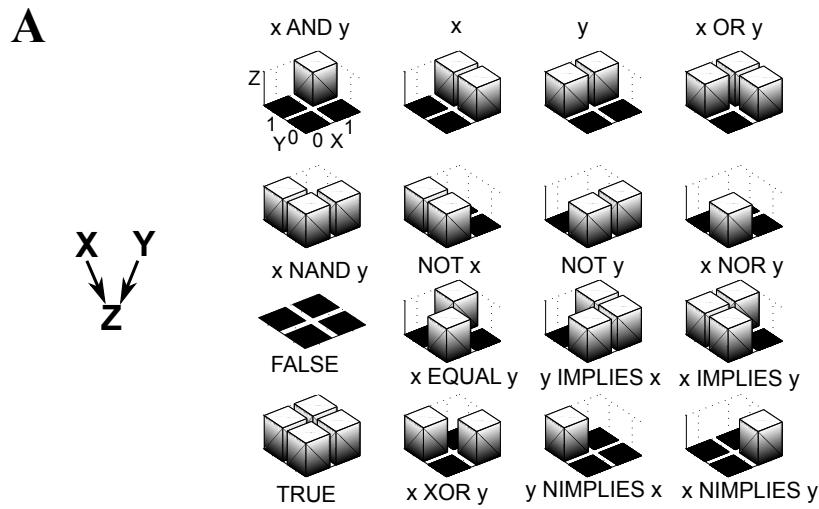
Funciones de regulación similares a las dadas por funciones de Hill se pueden deducir de primeros principios partiendo de leyes microscópicas fundamentales y usando herramientas de física estadística. Los *modelos termodinámicos* fueron introducidos por Shea y Ackers [1, 128] para estudiar cuantitativamente la regulación del represor del fago lambda, y han sido sistematizados y revisados más recientemente en las referencias [19, 20, 130]. Estos modelos se basan en considerar la respuesta de un gen proporcional a la cantidad de RNA polimerasa unida al promotor de dicho gen, y en asumir que el sistema regulatorio del gen se encuentra en *equilibrio termodinámico*, es decir, en un estado de mínima energía libre. Por tanto la cantidad de proteína producida por un gen se puede calcular a partir de la *probabilidad* de ocupación de su promotor por la polimerasa, teniendo en cuenta los diferentes factores que pueden regular dicho promotor. Esta probabilidad se calcula mediante técnicas de física estadística, teniendo en cuenta todas las configuraciones posibles en que  $P$  moléculas de polimerasa se pueden distribuir entre el promotor de interés y  $N$  sitios no específicos de la cadena de DNA [19]. Cada una de estas configuraciones lleva asociada una probabilidad dada por la diferencia de energía libre,  $\Delta G_i$ , de las reacciones de unión/desunión de la RNA polimerasa al DNA, donde  $i = \{\text{promotor, sitio no específico}\}$  indica una de las dos posibilidades de sitios de unión de la polimerasa. Las probabilidades tienen la forma funcional habitual de un *factor de Boltzmann*:

$$p_i = \frac{e^{-\Delta G_i/k_B T}}{Z} \quad (19.29)$$

donde  $Z$  es la función de partición termodinámica. Como las probabilidades de unión de la polimerasa al promotor son también proporcionales a los diferentes factores de transcripción que lo regulan, así como a su configuración espacial (tal como *DNA looping*), podemos expresar la respuesta del gen en función de las concentraciones de los factores transcripcionales y su configuración espacial, con los factores de Boltzmann agrupados en constantes (constantes de equilibrio termodinámicas) que multiplican a las concentraciones de los reguladores. La forma funcional de estas probabilidades, que da la función respuesta del gen, es similar a la de las funciones de Hill discutidas en la sección anterior [19].

### 19.3.4. Regulación combinatoria

Es habitual en regulación genética, especialmente en eucariotas, que dos o más factores transcripcionales regulen la expresión de un mismo gen. Diferentes factores transcripcionales se pueden activar en respuesta a diferentes señales que se procesan de forma combinatoria, dando lugar a programas genéticos específicos. La forma en que diferentes señales o *inputs* se combinan para determinar la respuesta transcripcional de un gen se puede representar por una función respuesta generalizada. Esta *función respuesta* proporciona la actividad del gen en función de  $N$  *inputs*, y en muchas ocasiones se puede obtener como combinación de funciones de Hill simples, dando lugar a diferentes puertas lógicas como las que vimos en la sección de ‘Modelos booleanos’. Por ejemplo, en la activación de un gen  $Z$  por dos factores transcripcionales diferentes  $X$  e  $Y$ , Figura 19.6A, estos factores pueden actuar de forma cooperativa y sólo favorecer la transcripción de  $Z$  cuando ambos estén unidos a su promotor, dando lugar a una puerta lógica de tipo ‘AND’ (primer panel de la fila superior en la Figura 19.6A). Una



**Figura 19.6:** A: Esquema de un gen  $Z$  en el que convergen dos *inputs* activadores  $X$  e  $Y$ . A la derecha se representan las 16 posibles puertas lógicas que pueden describir la respuesta de  $Z$  en función de los niveles de expresión de  $X$  e  $Y$ . Por ejemplo, para el primer panel de la fila de arriba, ' $x \text{ AND } y$ ', el gen  $Z$  está activo únicamente si tanto  $X$  como  $Y$  tienen valores altos de expresión. B: Función respuesta experimental [69] del gen  $malK$  que controla el metabolismo del azúcar maltotriosa. Este gen se activa en respuesta a dos inductores:  $cAMP$ , que activa el factor transcripcional  $CRP$  que directamente regula  $malK$ , y el propio azúcar que activa el factor transcripcional intermedio  $malT$ , a su vez regulado también por  $CRP$ . La función respuesta tiene la forma de una puerta lógica 'AND', y puede ajustarse bien por una función respuesta producto de dos funciones de Hill de activación independientes. Los datos experimentales de la Ref. [69] han sido proporcionados por Anat Bren. C. Función respuesta del gen  $malT$ , que responde de forma efectiva como una puerta  $y$  (sólo se activa si  $cAMP$  tiene un nivel alto de expresión). El ajuste del panel inferior es a una función de Hill simple con una tasa de transcripción basal modulada por el nivel de maltotriosa.

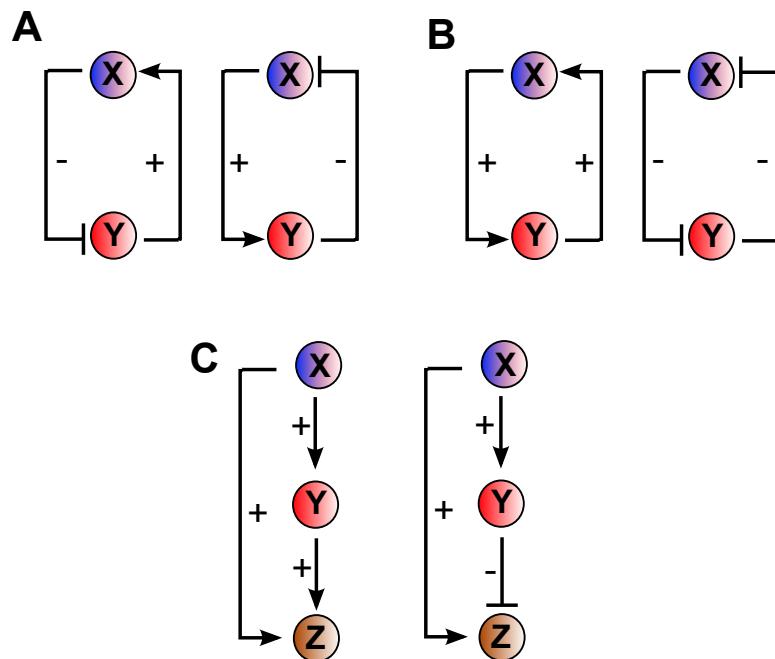
función respuesta de este tipo se puede modelizar matemáticamente por una función de la forma:

$$f_{AND}(x, y) = \text{Hill}_{act}(x) \cdot \text{Hill}_{act}(y) \quad (19.30)$$

donde  $\text{Hill}_{act}$  es la función de Hill simple descrita por la Ecuación 19.27. Las funciones respuesta de genes de *E.coli* controlados por dos inductores fueron medidas experimentalmente de forma sistemática por Alon y colaboradores [69, 91], que encontraron una gran variedad de respuestas, algunas de ellas con parecido claro a puertas lógicas como las representadas en la Figura 19.6B,C. Trabajos teóricos previos predijeron este control transcripcional combinatorio de genes importantes, y sugirieron que la lógica transcripcional puede ser explotada por las células para actuar como ‘computadores moleculares’ [26].

### 19.3.5. Análisis de motivos de red: osciladores, interruptores y generadores de pulsos

Los modelos cinéticos discutidos arriba permiten describir de forma cuantitativa las interacciones entre genes y su dinámica, y son una herramienta muy útil para estudiar la posible función de algunos de los motivos de red o módulos encontrados en las redes de regulación [6, 17]. Entre los motivos más abundantes en redes de regulación se encuentran módulos de dos nodos (genes) con *interacciones de retroalimentación* (*feedback*) (Figura 19.7A,B).



**Figura 19.7:** A: Motivos de dos componentes con retroalimentación negativa. B: Motivos de dos componentes con retroalimentación positiva. C: Motivos de tres componentes especialmente abundantes en redes transcripcionales: *feedforward loops*. El motivo de la izquierda, donde el signo de la interacción directa de *X* con *Z* es igual al signo total de la interacción indirecta, se denomina *feedforward loop* coherente. El motivo de la derecha es un *feedforward loop* incoherente, ya que la interacción directa es positiva y la indirecta a través de la especie *Y* es negativa.

En la Figura 19.7A, las dos interacciones en bucle tienen distinto signo, y por tanto el circuito formado por los genes *X* e *Y* tiene *retroalimentación negativa*, mientras que en la Figura 19.7B las dos

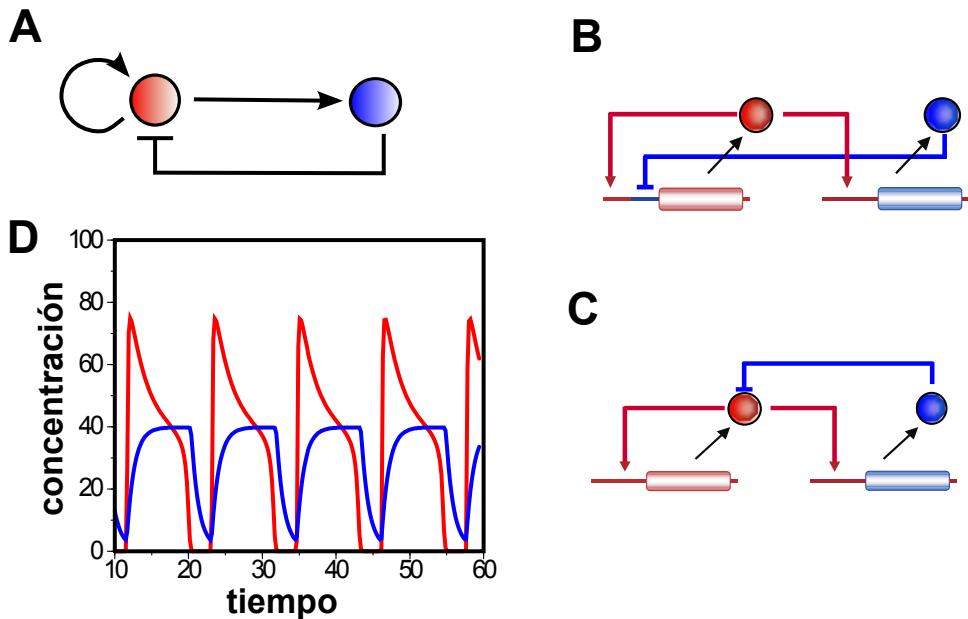
interacciones tienen signo idéntico (activación mutua o inhibición mutua) dando lugar a *retroalimentación positiva*. Por otro lado, se ha encontrado que existe un tipo de motivos sobreabundante en redes transcripcionales [129], formado por tres genes en el que uno actúa como *input* (gen *X*, Figura 19.7C) interaccionando con un *output* (gen *Z*) de forma tanto directa como a través de una especie intermedia *Y*. Este tipo de motivos se denominan *feedforward loops*, y pueden ser *coherentes* (si tanto la interacción directa como la indirecta tienen el mismo signo) o *incoherentes* (interacciones de signo opuesto). Dependiendo del contexto, podemos estar interesados en diferentes propiedades de la respuesta o *output* para caracterizar la función de un motivo de red. Alon y colaboradores se centraron especialmente en la dinámica de la respuesta a un cambio de intensidad o amplitud del *input* [6], investigando los tiempos de respuesta (tiempo que tarda el *output* en alcanzar su nuevo valor de equilibrio cuando se produce un cambio repentino en el *input*). Existen abundantes trabajos que han investigado la función de los motivos de red usando otras propiedades, como la capacidad de adaptarse a cambios continuos del *input* [85], o producir múltiples estados de expresión estables o patrones oscilatorios [28]. Nosotros nos centraremos en este tipo de propiedades que tienen una gran relevancia en la función celular.

### Retroalimentación negativa: osciladores genéticos

Las *oscilaciones periódicas* en los niveles de ciertas proteínas son responsables de importantes procesos fisiológicos y celulares, desde la división celular [113] o los ritmos circadianos que controlan la respuesta de los organismos a los ciclos de luz/oscuridad [54], hasta la codificación de señales que inducen reparación de daño en DNA [116]. Algunos de estos procesos han sido caracterizados molecularmente en un gran detalle, y en todos se han encontrado motivos de retroalimentación negativa entre componentes clave, complementados en muchas ocasiones por retroalimentaciones positivas [102]. Un ejemplo de motivo ‘mínimo’ capaz de producir oscilaciones es el constituido por un sistema de dos genes (activador/represor), en el que el gen activador a su vez tiene autorregulación positiva [58], Figura 19.8A.

Las interacciones esquematizadas en la Figura 19.8A se pueden implementar a nivel genético o molecular de diversas formas. Por ejemplo, tomemos la interacción negativa del gen represor (en azul) al activador. Si el represor es un factor transcripcional del activador, inhibirá la transcripción del gen que codifica la proteína activadora uniéndose al promotor e impidiendo la acción de la polimerasa (represión transcripcional), Figura 19.8B. Existen otros mecanismos posibles de interacción negativa (Ejercicio 1): por ejemplo, las proteínas represora y activadora pueden interaccionar físicamente para formar un complejo que es rápidamente degradado por el proteasoma (represión postraduccional), Figura 19.8C. Aunque ambos mecanismos son capaces de producir oscilaciones periódicas en expresión de proteína (Figura 19.8D y Ejercicio 2), la amplitud, periodo y dinámica de las oscilaciones puede diferir enormemente dependiendo del tipo de represión [58]. En consecuencia, la ‘topología’ de un motivo de red es en muchos casos insuficiente para determinar su función [65], y debemos incluir en el modelo toda la información relevante disponible sobre las interacciones (mecanismos moleculares o funciones de respuesta).

El motivo de retroalimentación negativa descrito aquí no es el único motivo de red sencillo capaz de producir oscilaciones de proteína. De hecho, una simple autorregulación negativa (un motivo de un solo gen) puede dar lugar a oscilaciones en su expresión de proteína si se tienen en cuenta los ‘retardos’ en la interacción debidos a los procesos más lentos de transcripción y traducción [78]. Este mecanismo se ha comprobado experimentalmente en la formación periódica de somitas durante el desarrollo del pez cebra [52]. En uno de los primeros trabajos de *Biología Sintética* (el diseño de redes genéticas artificiales que cumplan una determinada función), se construyó una red de tres genes que se reprimían uno a otro en un motivo circular (lo que constituye una retroalimentación negativa ‘indirecta’ de cada uno de los nodos). Esta pequeña red, a la que se denominó ‘*repressilator*’ [37] puede generar oscilaciones en los niveles de



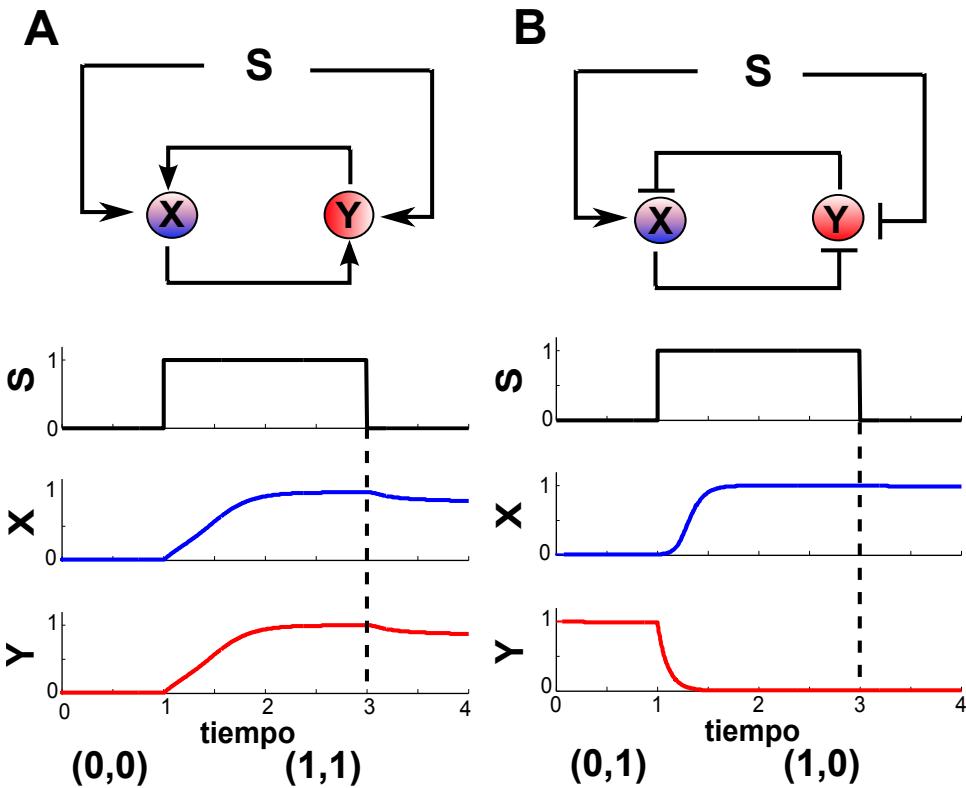
**Figura 19.8:** A: Topología de un motivo de red de dos componentes con retroalimentación negativa y autorregulación positiva del componente activador (en rojo). B, C: implementación en términos de genes y proteínas del motivo en el panel A. En la figura B, el represor (azul) es un factor de transcripción del activador, que inhibe su transcripción (represión transcripcional). En el panel C, el represor se une a la proteína activadora para favorecer su degradación (represión postraduccional). D: Oscilaciones de proteína activadora y represora en un modelo cinético de la dinámica de dicho motivo (correspondiente a represión transcripcional).

expresión de cada uno de los componentes. Una observación interesante es el hecho de que en la mayor parte de los osciladores genéticos caracterizados en detalle, como los responsables de ritmos circadianos o división celular, se observan motivos de activador/represor como el discutido en la Figura 19.8, mientras que topologías como la del *repressilator* no aparecen. Una explicación posible a este hecho viene dada por el efecto de la *variabilidad estocástica* o ‘ruido’ en la dinámica de los osciladores (explicaremos los orígenes de este ruido en la siguiente sección). Un sistema biológico se encuentra siempre sometido a fluctuaciones de sus componentes que pueden alterar su funcionamiento; mientras que en el *repressilator* las oscilaciones son muy variables debido a este ruido [37], un motivo de activador/represor es mucho más robusto a fluctuaciones [15], lo cual puede constituir una ventaja evolutiva para el sistema. De hecho se ha demostrado que la presencia de autorregulaciones o retroalimentaciones positivas coexistiendo con las negativas en osciladores genéticos favorece la robustez de las oscilaciones, tanto a fluctuaciones aleatorias como a cambios en los parámetros bioquímicos [137, 142].

### Retroalimentación positiva: interruptores genéticos

La *retroalimentación positiva* de dos componentes, tanto de activación mutua como de inhibición mutua (Figura 19.7B), son un motivo especialmente frecuente en redes de regulación involucradas en desarrollo y diferenciación celular [31, 59]. Una razón posible para esta abundancia es que este tipo de motivos muestran unas características dinámicas muy especiales, ya que pueden presentar dos patrones de expresión estables simultáneamente y pasar de uno a otro según el contexto de señalización celular. A este tipo de comportamiento se le denomina *biestabilidad*, y se dice que la red regulatoria que lo origina puede funcionar como un ‘interruptor’ genético que alterna entre dos estados diferentes.

Podemos fácilmente imaginar la importancia de un mecanismo de este tipo en redes transcripcionales que controlan la toma de decisiones en procesos de desarrollo y diferenciación, donde las células deben pasar de un estado fenotípico a otro bajo la acción de una señal externa (por ejemplo un morfógeno [9] o una hormona [153]).

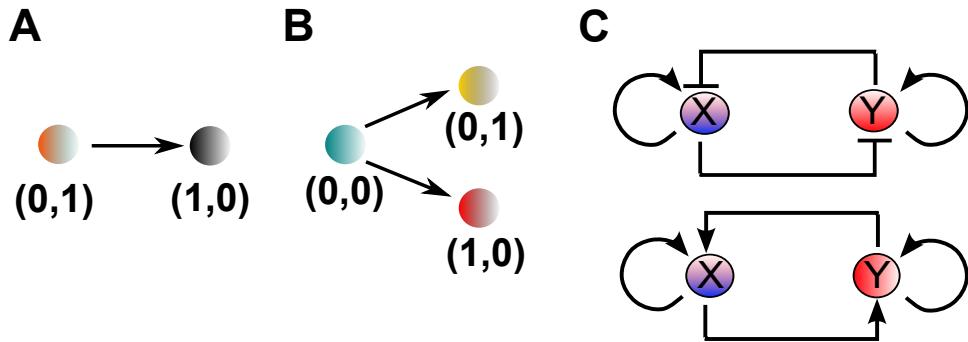


**Figura 19.9:** Motivos de retroalimentación positiva que dan lugar a interruptores genéticos. A: Motivo de activación mutua (panel superior). Dinámica de los componentes  $X$  e  $Y$  bajo el efecto de un pulso de señal  $S$  que activa cualquiera de los dos componentes. B: Motivo de inhibición mutua y evolución temporal de los componentes  $X$  e  $Y$  bajo un pulso de señal  $S$  que o bien activa el componente  $X$  o inhibe el  $Y$ .

El tipo de dinámica que origina una ‘*decisión celular*’ viene esquematizada en la Figura 19.9 (ver también Ejercicio 3). En el panel A tenemos un *bucle de activación mutua*; si los dos genes que lo componen no están activos (no hay transcripción), el sistema está ‘apagado’ y no hay producción de  $X$  ni de  $Y$ . En notación digital, diríamos que se encuentra en el estado ‘ $(0, 0)$ ’. Imaginemos que llega al sistema un pulso de señal  $S$  (un factor de transcripción, una kinasa, etc.) que puede activar cualquiera de los dos componentes, o los dos simultáneamente. Esta señal inducirá la expresión de un componente que a su vez activará a su complementario, reforzando mutuamente su nivel de expresión hasta que el sistema pasa al estado ‘ $(1, 1)$ ’ o encendido. En el ejemplo de la Figura 19.9A, vemos que cuando el pulso de señal se desconecta (a  $t = 3$ , línea discontinua), el sistema permanece ‘encendido’ y no vuelve al estado original. Este efecto, que es análogo a un mecanismo de *histéresis*, implica que el sistema ‘recuerda’ el estado de expresión en que se encuentra aunque desaparezca la señal que lo ha inducido, es decir, que existe una cierta ‘memoria’ celular. La *memoria celular* es importante en todos los procesos de diferenciación para que, una vez tomada la decisión e iniciado el proceso, la célula termine completamente diferenciada.

En el interruptor de la Figura 19.9B, de *inhibición mutua*, la dinámica es complementaria al motivo

de activación mutua. Si uno de los componentes está expresado, reprimirá a su contrario evitando a su vez la inhibición por éste, de forma que los dos componentes alcanzarán estados de expresión opuestos. Imaginemos que el sistema está inicialmente en el estado '(0, 1)' (el gen *X* inactivo y el *Y* activo). Si en un cierto momento una señal *S* suficientemente intensa activa *X* o inhibe *Y*, puede cambiar el estado inicial a su complementario '(1, 0)', y permanecer en él aunque la señal desaparezca (presentando de nuevo la capacidad de memoria celular). Uno de los primeros interruptores genéticos caracterizados es el que da lugar a la transición entre los estados de lisis/lisogenia del fago Lambda [115], controlada por dos factores de transcripción que se inhiben mutuamente, *cI* y *cro*. En condiciones de lisogenia (virus inactivo e integrado en la bacteria hospedadora), *cI* se encuentra expresado a niveles altos, reprimiendo a *cro* y a una batería de genes que inducen al estado de lisis. Cuando la bacteria hospedadora sufre algún daño (por ejemplo por radiación ultravioleta), se inhibe la expresión de *cI*, liberando a *cro* que mantiene 'apagada' la transcripción de *cI*, favoreciendo la replicación del virus dentro de la célula.



**Figura 19.10:** A: Diferenciación fenotípica por un interruptor biestable con estados  $(0, 1)$  y  $(1, 0)$ . B: Diferenciación fenotípica en dos posibles linajes desde un estado pluripotente inicial. Esta situación correspondería a un interruptor multiestable con estados posibles  $(0, 0)$  (pluripotente),  $(0, 1)$  y  $(1, 0)$ . C: Motivos de dos componentes capaces de producir multiestabilidad.

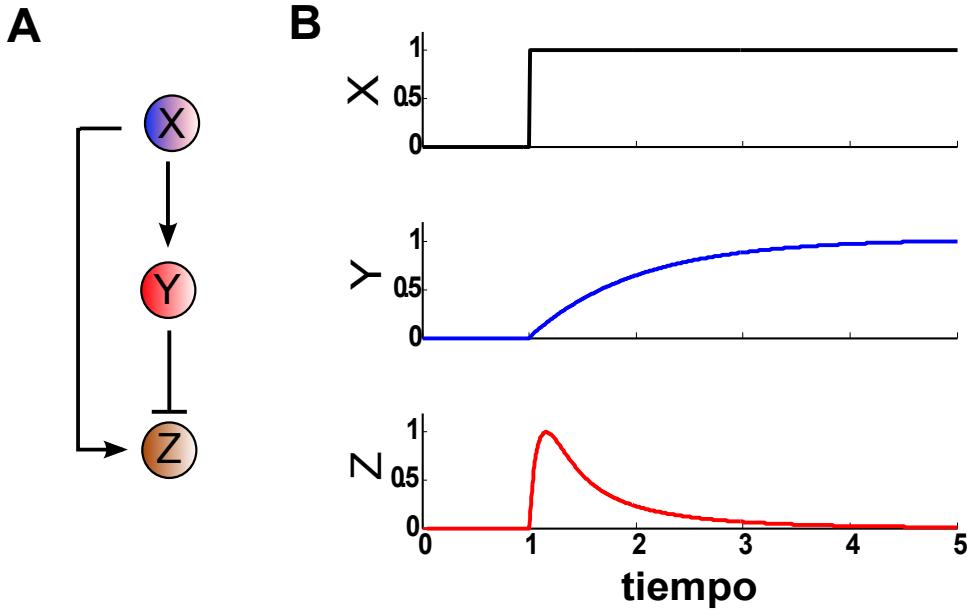
En la Figura 19.9 tenemos ejemplos de interruptores con dos estados de expresión estables, de forma que la célula sólo puede evolucionar a un único fenotipo diferente, Figura 19.10A. En algunas situaciones biológicas, como la diferenciación de células madre pluripotentes a diferentes linajes o tipos celulares, una misma red regulatoria parece controlar la elección de varias alternativas posibles, Figura 19.10B. Un ejemplo claro lo tenemos en hematopoiesis, donde una red de sólo dos factores de transcripción (GATA1 y PU.1) controla la diferenciación de células madre hematopoiéticas en células eritroides o mieloides [55]. En el lenguaje de interruptores genéticos, esto correspondería a una situación en que coexisten al menos tres estados de expresión estables. ¿Es posible la existencia de interruptores genéticos de *dos componentes multiestables*? La respuesta es que bucles de retroalimentación positivos como los estudiados arriba donde los dos componentes además se autorregulan positivamente, Figura 19.10C, pueden generar tres y hasta cuatro estados de expresión estables simultáneamente [59]. De hecho, tanto en el ejemplo mencionado de la red hematopoiética como en muchos otros casos de diferenciación celular, se han encontrado autorregulaciones positivas e interacciones mutuas entre genes clave que controlan el proceso [55, 59].

### **Feed-forward loops: pulsos y adaptación al estímulo**

Los motivos en bucles tipo *feed-forward*, Figura 19.7, aparecen en multitud de genes de la red transcripcional de *E.coli* [129], así como en redes regulatorias de levadura [75] y de humanos [24, 105]. Estos circuitos presentan propiedades dinámicas especiales [86]: por ejemplo, el bucle *feed-forward* coheren-

te con todas las interacciones positivas, Figura 19.7C, puede producir un retardo en la expresión del *output*(especie *Z*) cuando el *input* (componente *X*) se activa [86, 88]. Esto ocurre si la acción combinada de *X* y la especie intermedia *Y* sobre *Z* está controlada por una puerta lógica tipo ‘AND’ (ver Subsección 19.3.4), ya que en este caso ambas activaciones son necesarias para expresar *Z* y la acción indirecta a través de *Y* retarda su expresión. Esta propiedad convierte a este motivo en una especie de *detector de persistencia*: las señales que activan el *input X* deben ser de suficiente duración para llegar a activar el *output Z* a través de la especie intermedia, con lo que sólo señales persistentes producen respuesta. Esta propiedad puede ser de utilidad en redes de regulación sometidas a fluctuaciones, que de esta forma son capaces de distinguir entre ruido y señal verdadera. El bucle *feed-forward* coherente tiene otras propiedades interesantes: es capaz de detectar adecuadamente tanto cambios en amplitud de una señal estacionaria, como un amplio rango de frecuencias en señales oscilatorias [57], siendo además un detector muy robusto a ruido.

El bucle *feed-forward* incoherente con la interacción *Y-Z* negativa (Figura 19.11A) presenta propiedades muy diferentes: acelera el tiempo de respuesta del *output* cuando se activa el *input X* [87], y responde con un pulso a un cambio brusco de la señal, Figura 19.11B. La activación directa del *input X* al *output Z* produce un aumento rápido de la concentración de *Z*, que empieza a decrecer una vez que entra en juego la inhibición indirecta a través de la especie *Y*. La Figura 19.11B también pone de relieve otro comportamiento relevante que pueden generar los bucles *feed-forward* incoherentes: la capacidad de adaptar la respuesta a un estímulo constante (Ejercicio 4). En respuesta a un estímulo o *input*, el sistema cambia el nivel del *output*, pero al cabo de un cierto tiempo se recuperan los niveles existentes antes de la estimulación, aunque el estímulo continúe (línea roja en la Figura 19.11B). La *adaptación* es un mecanismo muy empleado en redes sensoriales y de señalización para aumentar el rango de sensibilidad a diferentes *inputs*. Nuestro propio sentido del olfato funciona de esta manera: en respuesta a un determinado olor, se activa una respuesta neuronal que decae pronto aunque los receptores olfativos que la producen sigan activos, dando posibilidad a que un olor diferente pueda ser detectado por la misma red. En redes de señalización celular, quizás el ejemplo más estudiado de adaptación es el que controla la movilidad en respuesta a un gradiente químico (por ejemplo, de nutrientes para la célula), fenómeno que se conoce como *quimiotaxis*. La bacteria *E. coli*, por ejemplo, se mueve en su medio alternando series de desplazamientos lineales con intervalos erráticos (‘*tumbling*’) en los que elige una nueva dirección de desplazamiento al azar. Modificando la frecuencia del movimiento errático, la bacteria es capaz de moverse hacia ligandos químicos ‘atrayentes’ o alejarse de los ‘repelentes’. En un medio homogéneo en el ligando que activa la red de respuesta quimiotáctica, se sabe que la frecuencia de movimiento errático es insensible a la concentración de ligando [18], y este mecanismo de adaptación es robusto a cambios en las concentraciones de los componentes de la red [7]. Las amebas como *Dictyostelium discoideum* también presentan un mecanismo de adaptación similar, donde un bucle *feed-forward* incoherente controla la respuesta quimiotáctica [76]. Este tipo de motivo no es el único capaz de producir adaptación en la respuesta, de hecho la retroalimentación negativa es el mecanismo operativo en la red de *E. coli*. Un estudio sistemático de todos las redes posibles de tres nodos demostró que sólo dos motivos, la retroalimentación negativa y el bucle *feed-forward* incoherente, son capaces de adaptación perfecta. El motivo *feed-forward* incoherente presenta además otras propiedades dinámicas interesantes: cuando el *input* es oscilatorio, actúa como un detector de ‘banda alta’, favoreciendo la detección de oscilaciones de alta frecuencia [57]. También pueden operar como detectores ‘relativos’, es decir, que generan una respuesta que depende del cambio relativo del *input* respecto a su valor inicial [53]. Este comportamiento se conoce como ‘*ley de Weber*’ en los sistemas sensoriales (por ejemplo en el oído y la visión): el umbral de detección depende del cambio relativo entre estímulo y nivel de fondo (Ejercicio 5).



**Figura 19.11:** A: Motivo de bucle *feed-forward* incoherente. B: Dinámica de la especie intermedia *Y* y el *output Z* frente a un cambio brusco del *input X*.

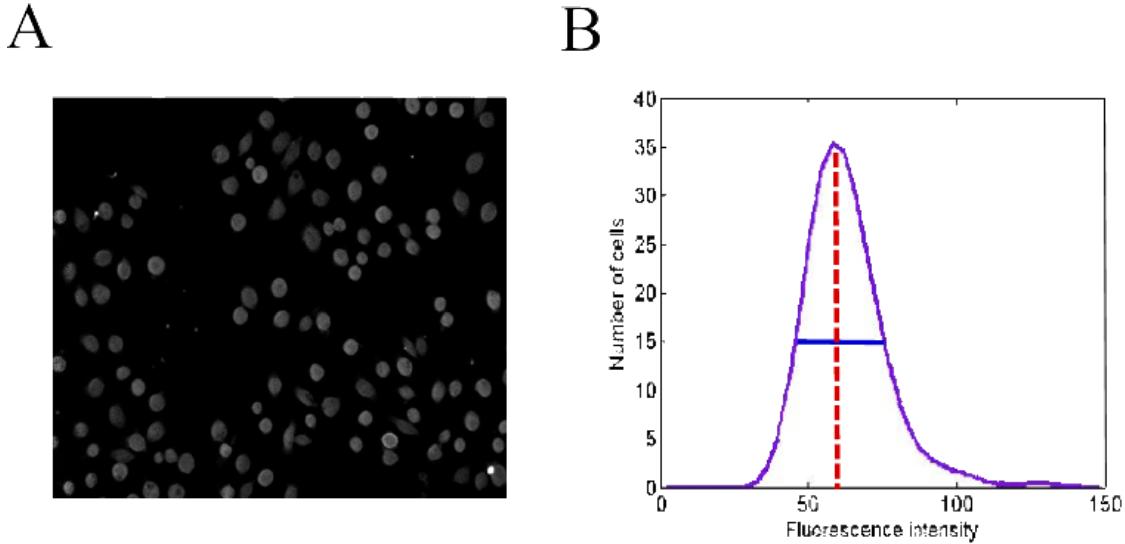
### 19.3.6. Ruido en expresión genética: orígenes y consecuencias

En la descripción de las redes de regulación vista hasta ahora hemos considerado que la expresión de cada gen o nodo de la red es estable o varía de forma continua en el tiempo, de forma que conociendo las interacciones y las funciones de regulación podemos predecir la actividad de cada gen a un tiempo dado. A este tipo de modelos, tanto de lógica booleana como cinéticos basados en ecuaciones diferenciales, se les denomina *deterministas*. La situación real en una célula es bien diferente: dos células de una población clonal, incluso en condiciones idénticas del medio (pH, temperatura, nutrientes, etc.) nunca muestran los mismos niveles de expresión de una proteína (Figura 19.12A), de forma que si cuantificamos la cantidad de proteína de una población clonal en unas condiciones determinadas no obtenemos un valor único, sino una *distribución de valores* centrada en torno a un valor medio y con una desviación estándar (Figura 19.12B, Sección 3.3.3). Esto es lo que se conoce como *variabilidad* o *ruido* en la expresión genética, y en experimentos de fluorescencia como el mostrado en la Figura 19.12 se suele cuantificar por el *coeficiente de variación* (CV) de la distribución de proteína, definido por:

$$CV = \frac{\sigma}{\mu} \quad (19.31)$$

donde  $\sigma$  es la desviación estándar de la distribución y  $\mu$  su valor medio (es decir, cuanto mayor sea la desviación estándar relativa a la media mayor es la variabilidad o ruido en la expresión de esa proteína)

El *ruido en expresión genética tiene múltiples orígenes*: inhomogeneidades del medio celular, diferente estado interno de cada célula (por ejemplo a lo largo de un ciclo celular), variabilidad en el número de moléculas que intervienen en los procesos bioquímicos, etc. Incluso considerando un gen como un sistema aislado dentro de la célula y teniendo en cuenta solamente las reacciones elementales que dan lugar a la expresión del gen, como las descritas en la sección anterior, existe una fuente inevitable de ruido debida a la naturaleza intrínsecamente aleatoria de las reacciones bioquímicas. En las siguientes secciones describiremos en detalle estas reacciones desde un punto de vista físico y matemático, y discutiremos las principales fuentes de variabilidad en expresión genética, la forma de estudiarlas y las



**Figura 19.12:** A: Imagen de inmunofluorescencia de la actividad de proteína *p32* en una población clonal de células HeLa. B: Cuantificación de la cantidad de proteína *p32* en la población: distribución de intensidades de fluorescencia. La media y la desviación estándar de la distribución están marcadas con línea roja discontinua y línea azul respectivamente.

consecuencias que tienen para la célula.

Desde un punto de vista histórico, la idea de que las reacciones elementales que tienen lugar dentro de la célula tienen una naturaleza aleatoria se remonta a los primeros estudios sobre regulación genética en bacterias. Novick y Weiner [103], investigando la producción de la enzima  $\beta$ -galactosidasa por el inductor TMG, observaron que la actividad de enzima en condiciones de inductor a concentraciones bajas o intermedias no se caracterizaba por una distribución continua de valores, sino que todas las células se encontraban únicamente en uno de dos estados posibles: actividad de  $\beta$ -galactosidasa nula o en su nivel máximo (fenómeno que denominaron ‘*all or none*’). Además, era completamente impredecible en qué momento una célula podía pasar del estado inactivo al completamente inducido. Estudiando en detalle el proceso de inducción, Novick y Weiner llegaron a la conclusión de que ‘la transición del estado no inducido al inducido es la consecuencia de un único evento aleatorio’. Con las técnicas actuales que permiten estudiar producción de proteína a nivel de molécula individual, se ha dilucidado recientemente el mecanismo exacto de este fenómeno y el proceso aleatorio que origina la transición [29].

### Ruido intrínseco: bajo número de copias

Consideremos el proceso más elemental para la producción estable de una proteína  $P$ : traducción constante a una velocidad  $\alpha$  y degradación lineal con tasa  $\delta$ :



Esto es lo que se conoce como proceso de “muerte/nacimiento” (*birth/death*) en el que moléculas individuales se producen y desaparecen a velocidad constante. Si aplicamos la *ley de acción de masas*, tenemos una ecuación lineal para la dinámica de  $P$ :

$$\frac{dP}{dt} = \alpha - \delta \cdot P \quad (19.33)$$

La solución de esta ecuación cuando no hay inicialmente nada de proteína  $P$  es la curva

$$P(t) = \frac{\alpha}{\delta} \left(1 - e^{-\delta t}\right) \quad (19.34)$$

que a tiempos largos da una cantidad constante de proteína (el estado de equilibrio o estacionario)

$$P_{eq} = \frac{\alpha}{\delta} \quad (19.35)$$

Como señalamos en la sección anterior, la ley de acción de masas produce una ecuación determinista en que las variables se miden en unidades de concentración. Valores típicos de las constantes  $\alpha$  y  $\delta$  para la bacteria *E.coli* son  $\alpha = 100 \text{ nM.h}^{-1}$  y  $\delta = 2 \text{ h}^{-1}$ . Prestemos atención al significado de estos valores: para proteínas estables, su tasa de desaparición está gobernada por el tiempo de división celular como  $\delta = \ln 2 / \tau_{cc}$ , donde  $\tau_{cc}$  es el periodo de un ciclo celular. Por tanto  $\delta = 2 \text{ h}^{-1}$  supone que  $P$  es una proteína estable dentro de una bacteria que se divide cada 20 minutos, un valor típico para *E.coli* en un medio rico en nutrientes (ver la Tabla 19.2). Por otro lado, *E.coli* tiene una forma aproximadamente cilíndrica de una micra de diámetro y unas dos micras de largo, lo que nos da un volumen de  $\sim 10^{-15} \text{ l}$ . Por tanto, tenemos la equivalencia aproximada entre concentración y número de moléculas por célula en *E.coli*:

$$1 \text{nM} \sim 1 \text{molécula por célula}, \quad (19.36)$$

es decir, que la proteína  $P$  se produce a una velocidad de unas 100 moléculas por hora. De hecho, valores típicos de números de copia de proteína en bacterias van desde menos de diez hasta varios cientos. Por tanto, las reacciones bioquímicas esquematizadas en la Ec. (19.32) pueden tener lugar a concentraciones muy diluidas. En tales casos, la ley de acción de masas, que es esencialmente una ley macroscópica, no puede aplicarse y debemos recurrir a una *descripción probabilista* o microscópica de las reacciones bioquímicas: es decir, no podemos decir con exactitud qué número de moléculas tendremos a un tiempo determinado, sino que sólo podemos hablar de la *probabilidad* de tener un cierto número de moléculas a un tiempo dado. Además, las reacciones bioquímicas se producen cuando dos moléculas se aproximan e interaccionan con la orientación adecuada. Si el número de estas moléculas es escaso, el tiempo entre reacciones consecutivas es en sí mismo una variable aleatoria. Por tanto, estrictamente, debemos abandonar la descripción determinista (continua en el tiempo) de los modelos cinéticos y sustituirla por una *descripción estocástica* con tiempos de reacción discretos y aleatorios, y reemplazar las concentraciones por probabilidades de densidad de moléculas. Esto requiere una descripción matemática diferente a la de ecuaciones diferenciales ordinarias regidas por la ley de acción de masas.

### 19.3.7. Modelos cinéticos estocásticos: ecuación maestra y ecuación de Langevin

El formalismo matemático riguroso que describe la evolución temporal de la *probabilidad* de tener  $n$  moléculas de proteína  $P$  a un tiempo  $t$  dado (que denotaremos por  $p_n(t)$ ) se conoce como *ecuación maestra* [44, 144] y, al igual que ocurría con las ecuaciones cinéticas (Ecuación 19.14), se puede expresar como un balance entre la ‘pérdida’ de probabilidad  $p_n(t)$  y la ‘ganancia’ de  $p_n(t)$ ,

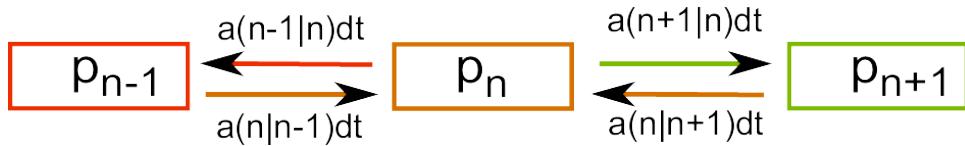
$$\frac{dp_n(t)}{dt} = \text{ganancia de } p_n(t) - \text{pérdida de } p_n(t) \quad (19.37)$$

Por tanto, para escribir la ecuación maestra necesitamos fijarnos en las reacciones que producen pérdida o ganancia de cada una de las especies moleculares involucradas, y tener un medio de obtener las probabilidades de cada reacción a tiempo  $t$ . Una descripción física rigurosa a nivel microscópico de

las reacciones bioquímicas establece que estas probabilidades microscópicas son proporcionales precisamente a las constantes de reacción macroscópicas que aparecen en la ley de acción de masas [48]. Así, para el proceso de muerte/nacimiento descrito por las dos reacciones (Ecuación 19.32), tenemos que

$$\begin{aligned} \text{Prob}(P_{t+dt} = n+1 | P_t = n) &= a(n+1|n)dt \propto \alpha dt \\ \text{Prob}(P_{t+dt} = n-1 | P_t = n) &= a(n-1|n)dt \propto n\delta dt \end{aligned} \quad (19.38)$$

es decir, la probabilidad de pasar de  $n$  moléculas de proteína  $P$  a  $n+1$  a un tiempo  $t$  se puede expresar como  $a(n+1|n)dt$ , donde  $a(n+1|n)$  es la probabilidad de producción de una molécula de  $P$  por unidad de tiempo y es directamente proporcional a la tasa macroscópica de producción  $\alpha$ . De forma similar, la probabilidad de pasar de  $n$  a  $n-1$  moléculas de  $P$  a tiempo  $t$  es proporcional a  $n$  veces la tasa macroscópica de degradación  $\delta$  (ya que tenemos  $n$  moléculas potencialmente degradables). A continuación debemos considerar todas las reacciones discretas que producen pérdida o ganancia de la probabilidad  $p_n(t)$ , (Ecuación 19.37).



**Figura 19.13:** Reacciones de pérdida y ganancia de  $p_n$  junto con sus correspondientes probabilidades.

En nuestro caso tenemos sólo las cuatro posibilidades representadas en la Figura 19.13: en las reacciones de ganancia de probabilidad  $p_n$  (flechas marrones) pasamos de  $n-1$  a  $n$  moléculas de proteína por producción de una molécula, o de  $n+1$  a  $n$  por degradación de una molécula, mientras que en las reacciones de pérdida de probabilidad  $p_n$  (flechas roja y verde) pasamos de  $n$  moléculas a  $n+1/n-1$  por producción/degradación respectivamente. Utilizando las expresiones para las probabilidades de reacción por unidad de tiempo,  $a(n+1|n)$ ,  $a(n-1|n)$ ,  $a(n|n+1)$  y  $a(n|n-1)$  dadas por las ecuaciones (19.38), y tomando el límite  $dt \rightarrow 0$ , la *ecuación maestra* para este proceso simplificado de expresión de proteína se puede escribir como:

$$\begin{aligned} \frac{dp_n(t)}{dt} &= a(n|n-1)p_{n-1}(t) + a(n|n+1)p_{n+1}(t) - [a(n-1|n) + a(n+1|n)] p_n(t) \\ &= \alpha p_{n-1}(t) + (n+1)\delta p_{n+1}(t) - (\alpha + n\delta)p_n(t) \end{aligned} \quad (19.39)$$

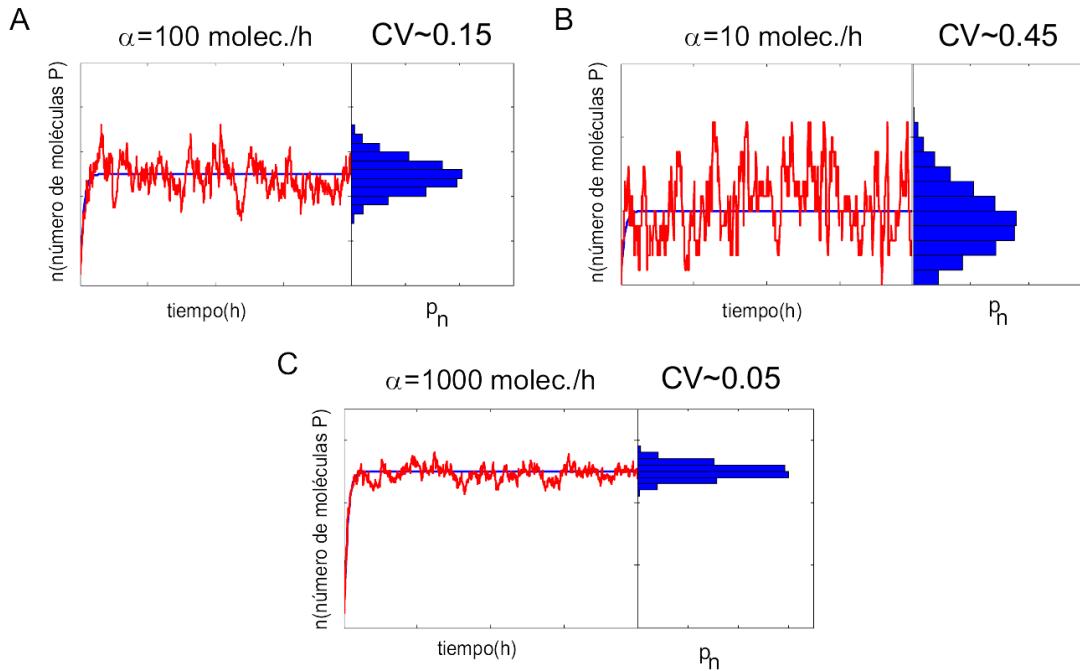
Para sistemas sencillos como es este caso la ecuación maestra se puede resolver analíticamente y obtener una expresión matemática para la probabilidad  $p_n(t)$  [44] (en general nos interesa la distribución de probabilidad estacionaria  $p_n(t \rightarrow \infty)$ ).

Cuando tenemos varias especies moleculares y múltiples reacciones bioquímicas la ecuación maestra no es en general resoluble y debemos recurrir a algoritmos numéricos o aproximaciones para obtener bien  $p_n$ , bien su media y desviación estándar para calcular el nivel de ruido mediante el coeficiente de variación (Ecuación 19.31). Entre los algoritmos más populares para simular de forma numéricamente ‘exacta’ la ecuación maestra se encuentra el *algoritmo original de Gillespie* [50] o alguna de sus variantes [47] (ver la Sección 19.7). Una de las aproximaciones más habituales a la ecuación maestra es la llamada *ecuación de Langevin*, que puede derivarse a partir de la ecuación maestra bajo ciertas aproximaciones [49], y que tiene una forma similar a las ecuaciones cinéticas deterministas dadas por la ley de acción de masas con un término aleatorio adicional que depende de las reacciones de producción/degradación. Para nuestro proceso de nacimiento/muerte tenemos:

$$\frac{dP}{dt} = \alpha - \delta \cdot P + \sqrt{\alpha + \delta \cdot P} \eta(t) \quad (19.40)$$

donde  $\eta(t)$  es una variable aleatoria gaussiana con estadística de ruido blanco,  $\langle \eta(t)\eta(0) \rangle = \delta(t)$ . La ecuación de Langevin se puede resolver numéricamente y en general su solución es mucho más eficiente que la de la ecuación maestra por el algoritmo de Gillespie [2], por lo que suele ser el método de preferencia para describir sistemas estocásticos de muchas variables.

Por analogía con el origen histórico de la ecuación de Langevin para describir movimiento browniano y procesos físicos de difusión [44], al factor que multiplica a la variable aleatoria  $\eta(t)$  y que da la amplitud de las fluctuaciones se le denomina coeficiente de difusión, en nuestro caso  $D \equiv \sqrt{\alpha + \delta \cdot P}$ . Nótese que en general el coeficiente de difusión depende de las variables del sistema y por tanto es también una variable estocástica dependiente del tiempo. Una aproximación teórica adicional a la ecuación de Langevin consiste en *linealizar* la dinámica estocástica alrededor de los valores medios de equilibrio, con lo que resulta un conjunto de ecuaciones lineales con coeficientes de difusión constantes, que pueden ser resueltas analíticamente por métodos estándar. Esto es lo que se conoce como *aproximación de ruido lineal* [36], y ha sido empleada habitualmente para entender las diferentes fuentes de ruido en redes regulatorias [57, 111].



**Figura 19.14:** Trayectorias determinista (azul) y estocástica (roja) simulada con el algoritmo de Gillespie, para el proceso de expresión de proteína  $P$  dado por las ecuaciones (19.32)-(19.33). En el panel de la derecha se representa la probabilidad estacionaria  $p_n$  (probabilidad de tener  $n$  moléculas de proteína una vez que el proceso ha alcanzado el equilibrio). Junto a la distribución  $p_n$  se indica la magnitud del ruido (coeficiente de variación, CV) dado por la Ec. (19.31). A: Tasa de producción  $\alpha = 100 \text{ moléculas/h}$ . B: Tasa de producción  $\alpha = 10 \text{ moléculas/h}$ . C: Tasa de producción  $\alpha = 1000 \text{ moléculas/h}$ . La constante de degradación es igual en todos los casos ( $\delta = 2 \text{ h}^{-1}$ ).

En la Figura 19.14 se muestra la evolución de proteína  $P$  en el tiempo según la ecuación cinética 19.33 (línea azul) y la correspondiente trayectoria estocástica resultante de simular las reacciones (Ecuación 19.32) mediante el algoritmo de Gillespie (línea roja). La correspondiente distribución de probabilidad para  $p_n$  se muestra como un histograma en los paneles de la derecha, junto con el nivel de ruido (coeficiente de variación, CV) dado por dichas distribuciones. Como ya se ha señalado, cuanto menor es el número de moléculas (que modulamos cambiando la tasa de producción de proteína  $\alpha$ ), más

estocástico es el sistema y su nivel de ruido será mayor. Para el sistema con  $\alpha = 10$  moléculas/h, panel B, el nivel medio de proteína por célula es de 5 moléculas, y el coeficiente de variación es elevado (0.45). Una observación interesante es que la dinámica determinista reproduce la evolución del valor medio de proteína, lo que nos dice que las ecuaciones cinéticas son válidas si estamos interesados únicamente en los promedios para todas las especies que reaccionan. De hecho la *ecuación de evolución para los valores medios* de las especies se puede deducir a partir de la ecuación maestra y es idéntica a la cinética dada por la ley de acción de masas [144]. Cuando los valores medios de moléculas son muy altos, como en el panel C, la dispersión alrededor de la media es muy baja y las ecuaciones deterministas son una buena aproximación a la dinámica del sistema. Otra observación es que el coeficiente de variación escala inversamente con el valor medio de proteína,  $\bar{n}$ , como  $CV \sim 1/\sqrt{\bar{n}}$ . Esta ley de escala se puede deducir de la aproximación de ruido lineal mencionada arriba [111] y se ha comprobado que se cumple experimentalmente en la mayoría de los genes de células eucariotas y procariotas [11, 96, 139].

## Modelos booleanos estocásticos

Aunque el formalismo teórico adecuado para estudiar ruido biológico es la ecuación maestra o sus aproximaciones, como la ecuación de Langevin, es posible simular efectos estocásticos debidos a concentraciones bajas de las especies reactivas en modelos lógicos o booleanos como los descritos arriba. Una aproximación posible consiste en cambiar el estado de los nodos de 0 a 1 y viceversa según una probabilidad predefinida [30]. El inconveniente de este método es que no tiene en cuenta la correlación entre los estados o niveles de expresión de las especies reactivas y la probabilidad de cambiar el estado de un nodo debido al ruido (mientras que en la ecuación maestra vimos que las probabilidades de reacción dependen en general de las concentraciones de reactivos). Una alternativa que evita esta desventaja es introducir estocasticidad en las propias funciones booleanas (ver Figura 19.3) en lugar de en el estado de los nodos, definiendo una probabilidad de fallo de la función lógica que dependa del estado de expresión de los *inputs*. De esta forma la probabilidad de transición de un nodo en un instante dado dependerá de la actividad de otros nodos en la red en ese instante, de forma análoga a la descripción estocástica de la ecuación maestra [45].

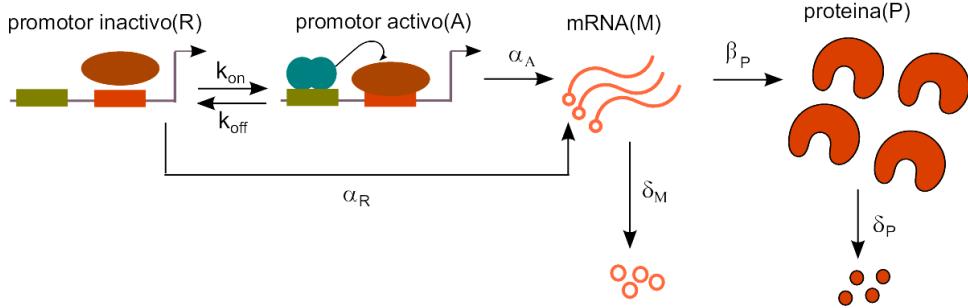
### 19.3.8. Diferentes fuentes de ruido en expresión genética

Consideremos un *modelo de expresión de proteína más realista* que el simple proceso de nacimiento/-muerte de la sección anterior: un gen cuyo promotor se puede encontrar en forma activa o inactiva, que da lugar a transcripción de mRNA que a su vez se traduce en proteína, Figura 19.15). En este caso tenemos cuatro especies moleculares (promotor activo A, promotor inactivo R, mRNA M y proteína P), y siete reacciones (activación/desactivación del promotor  $k_{on}/k_{off}$  respectivamente, transcripción desde el promotor inactivo/activo  $\alpha_R/\alpha_A$ , traducción  $\beta_P$  y degradaciones de mRNA/proteína  $\delta_M/\delta_P$ ).

La dinámica de mRNA y proteína a alto número de moléculas se puede aproximar por la ley de acción de masas. Si asumimos que las reacciones de activación/desactivación del promotor son rápidas respecto a la transcripción, traducción y degradaciones, podemos representar la evolución temporal de mRNA y proteína por el sistema de ecuaciones diferenciales (Ejercicio 6 y [68]):

$$\begin{aligned} \frac{dM}{dt} &= \frac{k_{on}}{k_{on} + k_{off}} \cdot \alpha_A + \frac{k_{off}}{k_{on} + k_{off}} \cdot \alpha_R - \delta_M \cdot M \\ \frac{dP}{dt} &= \beta_P \cdot M - \delta_P \cdot P \end{aligned} \tag{19.41}$$

Nótese que en esta ocasión la producción de  $P$  depende de la cantidad de mRNA  $M$ , que es una variable estocástica. Estas dos especies están por tanto acopladas y la variabilidad de una afectará a



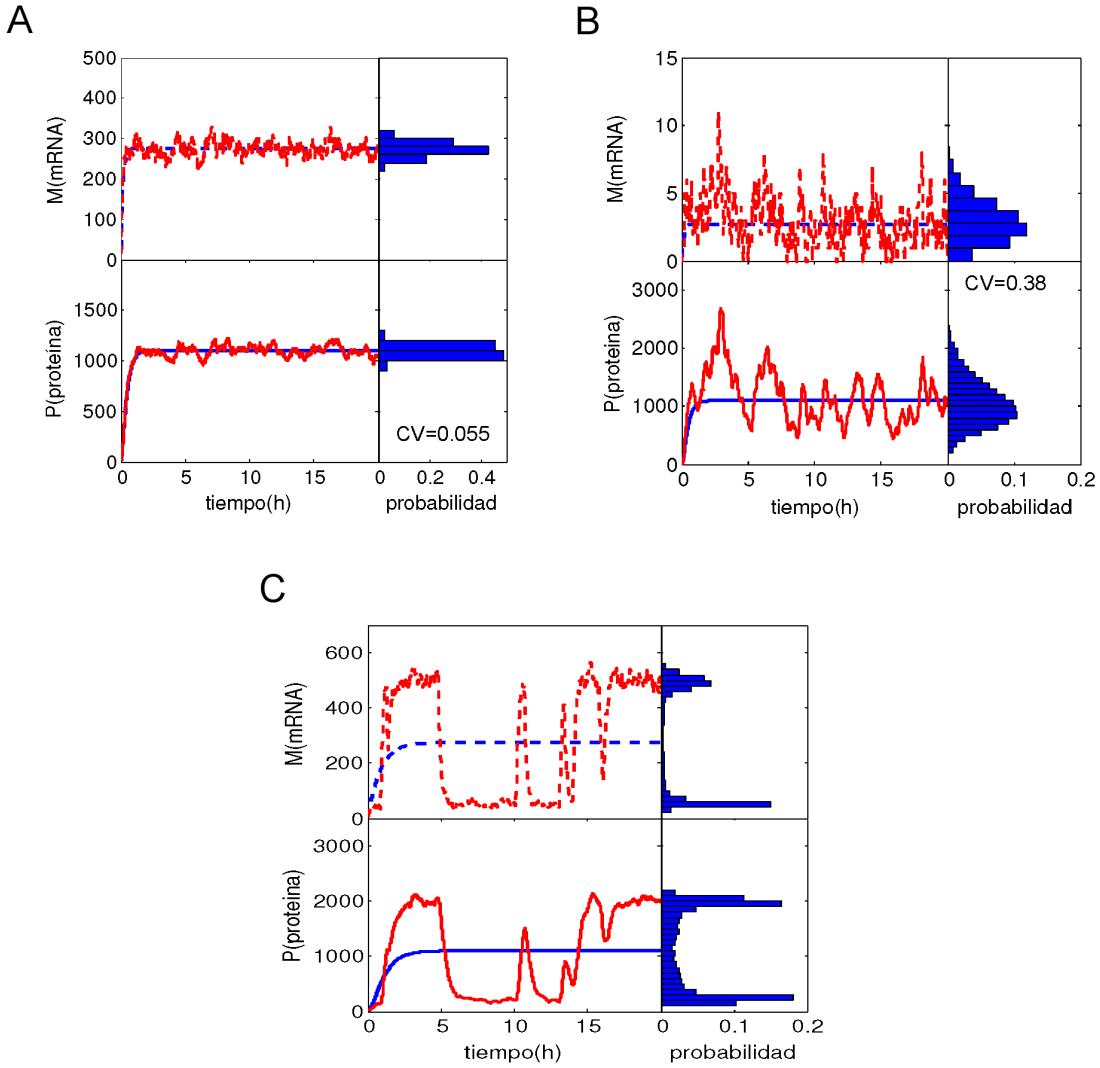
**Figura 19.15:** Esquema de expresión genética donde un promotor activo puede pasar a inactivo y viceversa (con tasas de activación/desactivación  $k_{on}/k_{off}$  respectivamente), dando lugar a transcripción de mRNA (con tasas  $\alpha_R/\alpha_A$  dependiendo de que haya transcripción desde el promotor activo o inactivo ( $\alpha_R \ll \alpha_A$ ), que a su vez traduce proteína con una tasa  $\beta_P$ . Tanto mRNA como proteína se degradan con velocidades  $\delta_M$  y  $\delta_P$  respectivamente.

la otra. En la Figura 19.16A, tenemos las trayectorias de mRNA y proteína y sus correspondientes distribuciones de probabilidad a alto número de moléculas (con un promedio de  $\sim 1000$  proteínas y  $\sim 300$  mRNAs). El ruido en proteína, como se espera de los resultados de la sección anterior, es muy bajo. En la Figura 19.16B hemos mantenido el mismo número elevado de proteínas en promedio, pero hemos disminuido mucho la cantidad de moléculas de mRNA ( $\sim 3$ ), haciendo 100 veces más pequeñas las tasas de transcripción  $\alpha_A$  y  $\alpha_R$ . El resultado es que el ruido resultante en mRNA se propaga a la producción de proteína mediante un mecanismo conocido como ‘ráfagas de traducción’ (*translational bursting*) [140]. Si la tasa de traducción es alta, cada mRNA dará en promedio un número alto de proteínas y este mecanismo amplificará el ruido debido al bajo número de copias de mRNA. Usando la aproximación de ruido lineal [111, 140] se puede demostrar que el coeficiente de variación es directamente proporcional al parámetro:

$$b = \frac{\beta_P}{\delta_M}, \quad (19.42)$$

que se conoce como *parámetro de ráfaga* (*burst parameter*) y da el número de proteínas producidas por mRNA (en el ejemplo de la Figura 19.16B  $b=200$ ). En uno de los primeros trabajos experimentales que midieron ruido en expresión genética en células individuales identificaron dichas ‘ráfagas de traducción’ como un mecanismo importante en células procariotas [108].

Imaginemos ahora que la activación y desactivación del promotor es un proceso lento (por tanto las ecuaciones de 19.41 no son válidas y debemos tener en cuenta explícitamente la dinámica para A y R). Podemos modelar este proceso disminuyendo las tasas de activación/desactivación  $k_{on}/k_{off}$ . En la Figura 19.16C hemos reducido los tiempos de activación/desactivación a aproximadamente un evento por hora ( $k_{off} = k_{on} = 0.01 \text{ min.}^{-1}$ ). En este caso lo que ocurre es que la producción de mRNA tiene lugar en ráfagas, como un proceso digital o binario, en el que durante ciertos intervalos de tiempo apenas hay producción de mRNA (el promotor se encuentra inactivo) mientras que en otros intervalos los niveles de mRNA alcanzan su valor máximo. Esto se refleja en la *distribución de proteína*, que puede llegar a ser *bimodal* (con dos máximos, panel derecho de la Figura 19.16C), de forma que en una misma población celular coexisten dos subpoblaciones, una con niveles muy bajos de proteína y otra con niveles máximos. Este mecanismo de ruido o variabilidad en expresión genética se conoce como ‘ráfagas transcripcionales’, y su magnitud viene gobernada por un parámetro similar al ‘parámetro de ráfaga’ descrito arriba, que da la eficiencia transcripcional o número promedio de mRNAs transcritos durante el tiempo que el promotor se encuentra activo,  $b = \alpha_A/k_{off}$ . Uno de los primeros estudios que investigaron ruido en expresión de genes en células eucariotas (en levadura) identificó el mecanismo de



**Figura 19.16:** Trayectorias determinista (azul) y estocástica (roja) para mRNA y proteína (en líneas discontinua y continua respectivamente), según el esquema de reacciones de la Figura 19.15. A: Con dinámica rápida de A y R, ver Ec. (19.41), y alto número de moléculas de mRNA y proteína. Los valores de los parámetros son  $k_{on}=10 \text{ min.}^{-1}$ ,  $k_{off} = 10 \text{ min.}^{-1}$ ,  $\alpha_A = 50 \text{ moléculas/min.}$ ,  $\alpha_R = 5 \text{ moléculas/min.}$ ,  $\beta_P = 0.2 \text{ min.}^{-1}$ ,  $\delta_M = 0.1 \text{ min.}^{-1}$ ,  $\delta_P = 0.05 \text{ min.}^{-1}$ . En el panel de la derecha se representan las distribuciones de número de mRNAs y proteínas, junto al coeficiente de variación (CV) de la distribución de proteína. B: Ruido debido a ráfagas de traducción. El número de mRNAs se ha reducido 100 veces disminuyendo la transcripción ( $\alpha_A = 0.5 \text{ moléculas/min.}$ ,  $\alpha_R = 0.05 \text{ moléculas/min.}$ ). El resto de los parámetros como en el panel A. C: Ruido y bimodalidad debidos a ráfagas transcripcionales: la dinámica de activación/desactivación del promotor es lenta,  $k_{on} = k_{off} = 0.01 \text{ min.}^{-1}$ . El resto de los parámetros como en el panel A.

ráfagas transcripcionales como una fuente probable de variabilidad en producción de proteína [21]. De hecho, trabajos experimentales más recientes han demostrado la existencia de distribuciones bimodales de proteína debidas únicamente a la activación/desactivación estocástica de un promotor [141].

Los mecanismos de ruido descritos aquí se conocen como ‘*ruido intrínseco*’, que afecta de forma diferente a cada gen y depende exclusivamente de la naturaleza estocástica de las reacciones bioquímicas

involucradas en la expresión de proteínas. Podemos imaginar fácilmente que existen otras fuentes de variabilidad celular, como diferencias en la progresión del ciclo celular, inhomogeneidades del entorno o cambios en las concentraciones locales y estado de otras proteínas reguladoras, polimerasas, etc., que pueden afectar de manera global a muchos de los genes de una misma célula. Este tipo de variabilidad es lo que se conoce como ‘*ruido extrínseco*’, y puede jugar un papel tan relevante como el ruido intrínseco [120]. El ruido intrínseco que afecta a un gen se puede discriminar del ruido extrínseco usando dos proteínas fluorescentes de diferente color controladas por el mismo promotor y equidistantes del origen de replicación [38].

La identificación de los mecanismos que originan variabilidad celular y sus consecuencias son áreas de investigación muy activas [112] (ver Sección 19.6). Procesos celulares fundamentales como la diferenciación celular [82], la apoptosis [135] o la resistencia de células tumorales a quimioterapia [25] tienen componentes estocásticos. A estos avances en el conocimiento y la relevancia biológica del ruido celular ha contribuido sin duda el desarrollo de técnicas experimentales de célula y molécula individual, que permiten explorar variabilidad en niveles de proteína a escala genómica [96], e incluso la cuantificación precisa de número de moléculas de mRNA y proteína a nivel global [139]. Las técnicas de fluorescencia de molécula individual, tales como FISH (*fluorescence in situ hybridization*) han permitido diseccionar en gran detalle la contribución real de los mecanismos de ruido discutidos arriba a la variabilidad en expresión genética [134].

## 19.4. Modelos metabólicos basados en restricciones

En este capítulo se han tratado, por el momento, dos aproximaciones teóricas: las redes complejas y los modelos matemáticos detallados. Por un lado, las redes complejas han contribuido enormemente a entender la estructura global de los sistemas biológicos, a partir de “catálogos” masivos de datos biológicos obtenidos experimentalmente. Pero el análisis de grafos normalmente carece del componente dinámico crucial para comprender en su totalidad los sistemas biológicos. Los modelos matemáticos detallados, por otro lado, se centran fundamentalmente en este componente dinámico, pero se hacen prácticamente inmanejables en sistemas con un gran número de componentes relevantes y de relaciones entre éstos. Debido a estas limitaciones, es extremadamente difícil dotarse de modelos de calidad capaces de predecir el comportamiento de un sistema biológico en unas condiciones determinadas.

En este sentido, uno de los modelos biológicos usados con más éxito son los denominados *modelos metabólicos basados en restricciones* (MBR). El objetivo de estos modelos consiste en predecir el comportamiento de un determinado metabolismo en términos de actividad de cada una de las reacciones presentes en una situación dada. Estos modelos se basan primeramente en el conocimiento experimental de las enzimas presentes y las estequiométrías de las reacciones asociadas para generar una reconstrucción metabólica *in silico*. A esta reconstrucción se le aplican una serie de asunciones biológicamente realistas, generalmente explicitadas como restricciones (por ejemplo, estableciendo límites realistas a las tasas de reacción, o imponiendo el balance de masas), estableciendo un conjunto de estados posibles de ese metabolismo. Finalmente, un algoritmo de optimización establece cuál de estos estados maximiza una función objetivo biológicamente realista, dada la composición nutricional del medio.

El primer paso en la *modelización* consiste en representar la red metabólica de una célula (Figura 19.17A) mediante una *matriz estequiométrica* que contiene los coeficientes estequiométricos de todas las reacciones (filas) para todos los metabolitos (columnas). A partir de esta matriz se define un sistema de ecuaciones diferenciales que describen la dinámica de cada metabolito (Figura 19.17C). La primera restricción importante en el modelo se impone igualando estas ecuaciones a cero. Es decir, no se permite la acumulación de ningún metabolito en la célula, lo cual se denomina *balance de masas*.

Una restricción adicional impone unas tasas de flujo máxima y mínima para cada reacción. En su conjunto, el balance de masas y los límites de flujo delimitarán un *espacio de soluciones* a nuestro sistema de ecuaciones, que comprenderá todas las combinaciones de flujos posibles para las reacciones de un sistema metabólico (Figura 19.17E).

El objetivo del modelo consiste en predecir, para determinada composición nutricional del medio (que define el *input*), cual es la combinación de flujos concreta que maximiza el *output* (por ejemplo, la tasa de producción de biomasa, o crecimiento). La incorporación de las funciones '*input*' y '*output*' en el modelo implica la definición de *pseudo-reacciones* que representarán la entrada y salida de componentes. Las pseudo-reacciones de entrada, o 'fuente', suelen tener la forma → *nutriente*. Definiendo el límite superior de flujo (de manera análoga a las demás reacciones), definiremos la cantidad del nutriente presente en el medio. De manera análoga, la pseudo-reacción '*output*' suele tener una estructura de tipo  $aA + bB + \dots + zZ \rightarrow$ , es decir, extrae del sistema aquellos metabolitos cuya producción metabólica queremos maximizar (o minimizar). Por ejemplo, si queremos predecir la tasa de crecimiento (que es la función objetivo usada con más frecuencia) debemos utilizar como output una '*función biomasa*', que simula la extracción de los componentes químicos necesarios para el crecimiento celular (aminoácidos para biosíntesis de proteínas, nucleótidos, etc.).

La puesta a punto de la *función biomasa* requiere un conocimiento preciso de a) la cantidad de cada molécula sintetizada por el sistema metabólico necesaria para generar una unidad de biomasa, y b) la cantidad de energía necesaria para los procesos biosintéticos considerados de manera implícita. Generalmente, esta información se obtiene a través de estimaciones experimentales. Pero la construcción de una función biomasa precisa y fiable no constituye una tarea nada fácil. Por ejemplo, es necesario considerar no sólo los componentes y la energía necesaria para los *procesos principales* (replicación, síntesis de proteínas, etc), sino también para *procesos secundarios*, como la reparación de daños en el DNA. También es fundamental tener en cuenta algunos componentes que son esenciales en muy pequeñas cantidades, como determinadas *vitaminas* o *cofactores*. Pero además, el establecimiento de una función biomasa adecuada plantea preguntas de carácter fundamental: *¿qué trata de hacer el organismo en un ambiente dado?* Aunque parece claro que el crecimiento es uno de los objetivos principales, pueden existir matices importantes, que dependerán en gran parte de su historia evolutiva [131]. Dado que las predicciones finales del modelo serán altamente sensibles a la forma precisa que tenga esta función biomasa, se han desarrollado multitud de herramientas adicionales que han permitido refinar su estimación hasta obtener resultados satisfactorios [40].

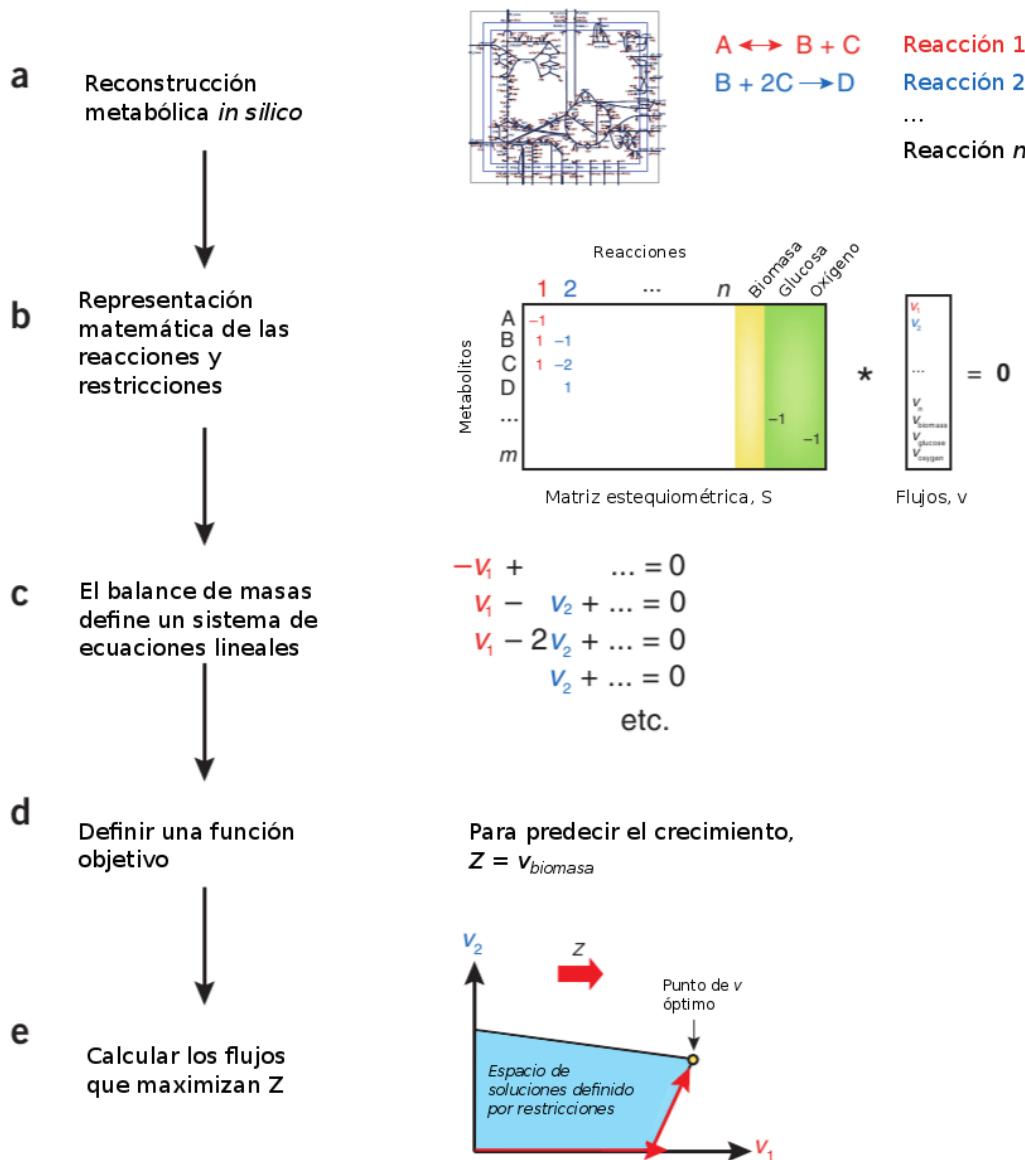
Una vez definido el espacio de soluciones para una determinada reconstrucción metabólica, las restricciones asociadas a éste y las funciones *input* y *output*, *¿cuál de estas soluciones maximiza la función objetivo?* Debido a la linealidad de las ecuaciones del sistema, el espacio de soluciones posibles tiene la forma de politopo - es decir, un poliedro en tantas dimensiones como flujos posibles existen, delimitado por las restricciones aplicadas. Es posible demostrar matemáticamente que la solución que optimiza cualquiera de las ecuaciones presentes siempre se encontrará en uno de los vértices (o aristas) de este espacio de soluciones. En la práctica, tener que "buscar" sólo entre los vértices, y no entre todos los puntos del politopo, reduce drásticamente el tiempo computacional requerido para hallar la solución, hasta el punto de convertir el problema en computacionalmente viable. Este método de optimización se denomina *programación lineal*.

En resumen, los *modelos metabólicos basados en restricciones* nos permiten conocer, para un determinado conjunto de reacciones (metabolismo), el flujo a través de cada una de ellas que optimiza una función objetivo para unas condiciones externas dadas. Como veremos más adelante, este tipo de predicciones pueden resultar muy útiles a la hora de entender, por ejemplo, la función de los componentes del sistema (por ejemplo, genes o metabolitos) en un contexto global. Sin embargo, es preciso señalar también una serie de limitaciones cruciales. En primer lugar, el modelo asume que el organismo tiene

la capacidad de utilizar de manera óptima su metabolismo. En otras palabras, asumimos que todos los componentes que consideramos de manera implícita (por ejemplo, los sistemas de señalización, regulación de la expresión génica y de la actividad enzimática) tienen la capacidad inherente de responder de la manera más eficaz posible a cualquier situación nutricional. En muchos casos, esto parece relativamente aceptable, por ejemplo, en unas condiciones nutricionales parecidas a las que se dan en el ambiente natural del organismo [70]. Pero los organismos no suelen disponer de una plasticidad infinita, y es de esperar que las predicciones del modelo diverjan significativamente de la realidad cuanto más “extraña” sea la composición de nutrientes medio externo. Esta limitación del modelo ha sido demostrada experimentalmente de manera elegante [70]. Dicho experimento parte de la sobreestimación que la predicción del modelo hace del crecimiento de *E. coli* en glicerol. Los autores hipotetizaron que el organismo no ha evolucionado mecanismos eficaces para maximizar su crecimiento en glicerol como única fuente de carbono, dado que presumiblemente nunca ha encontrado un ambiente similar en la naturaleza. Para demostrarlo, cultivaron en el laboratorio una cepa de *E. coli* en glicerol durante 40 días (aproximadamente 700 generaciones), en condiciones de presión selectiva fuerte sobre su tasa de crecimiento. El resultado fue que las cepas evolucionadas habían mejorado su crecimiento en glicerol, y se acercaban a la predicción del modelo.

Una limitación adicional de estos modelos está relacionada con su propia naturaleza como proceso de optimización. A saber, *¿qué es lo que realmente busca hacer un organismo en un ambiente dado?* Maximizar su crecimiento puede ser frecuentemente una buena aproximación, pero este no tiene por qué ser siempre el caso. De hecho, en evolución, parece ser frecuente la necesidad de optimizar dos o más objetivos enfrentados entre sí. En este tipo de situaciones suelen existir *diferentes conjuntos de fenotipos óptimos* (por ejemplo, diagramas de Pareto). Un ejemplo proviene precisamente del metabolismo. Una hipótesis es que, dado que los organismos viven en ambientes cambiantes, no sólo tienden a ajustar sus flujos metabólicos cada vez que cambia el ambiente sino también a variar estos flujos lo menos posible, dado que variarlos puede implicar un gran coste en expresión génica. La plausibilidad de este razonamiento fue puesta a prueba mediante una variante del modelo llamada *minimización del ajuste metabólico* (MOMA en sus siglas en inglés)[127]. Este método identifica, tomando determinada solución óptima como referencia, la solución posible más cercana bajo unas condiciones alternativas. El estudio en cuestión primero optimizó el crecimiento de *E. coli* mediante análisis de balance de flujo (FBA). Seguidamente, delecionó los genes del modelo uno a uno, y buscó en cada caso la solución usando MOMA. El resultado fue que MOMA tuvo un mejor rendimiento que FBA a la hora de predecir qué genes son esenciales. Más recientemente, un estudio experimental basado en medición de flujos metabólicos generalizó este resultado [126], demostrando que tanto la maximización del crecimiento como la minimización del ajuste metabólico juegan un papel central en la determinación del fenotipo.

A pesar de sus limitaciones (algunas relativamente caracterizadas), los modelos basados en restricciones son, como hemos visto, una herramienta muy útil en la exploración de la función metabólica. Un claro ejemplo es su potencial utilidad en ciencia aplicada, por ejemplo a la hora de mejorar cepas para la producción de compuestos de interés industrial [74]. Pero también pueden proporcionar pistas muy útiles en investigación básica, como veremos en la sección siguiente.



**Figura 19.17:** Formulación de un problema de análisis de balance de flujos. A) Una red metabólica es una lista de reacciones metabólicas y sus coeficientes estequiométricos. B) Esta reconstrucción es convertida en un modelo matemático en forma de una matriz (llamada  $S$ ) en la que cada fila representa un metabolito, y cada columna una reacción. La función biomasa está incorporada a la matriz en forma de reacción (columna amarilla), simulando los metabolitos consumidos por el crecimiento celular. Las reacciones de intercambio con el medio externo (columnas verdes) se usan para representar el flujo de metabolitos, como la glucosa o el oxígeno, hacia dentro o hacia fuera de la célula. C) En el estado estacionario, el flujo a través de cada reacción está dado por  $Sv = 0$ , definiendo un sistema de ecuaciones lineales. Como los modelos grandes contienen más reacciones que metabolitos, suele haber más de una solución posible a estas ecuaciones. D) La solución de estas ecuaciones para predecir el flujo máximo requiere la definición de una función objetivo  $Z = c^T v$ , donde  $c$  es un vector indicando la contribución de cada reacción  $v$  al objetivo. (En la práctica, cuando se maximiza sólo una reacción, tal como la reacción 'biomasa',  $c$  es un vector formado por ceros, excepto la propia reacción biomasa, que es un 1). E) Usamos un algoritmo de programación lineal para encontrar la distribución de flujos que maximiza la función objetivo dentro del espacio de soluciones permitidas (región azul), definida por los límites impuestos por el balance de masas y los límites superior e inferior de reacción. La flecha roja indica la manera de proceder del algoritmo de programación lineal, identificando el punto óptimo en un arista o vértice del espacio de soluciones. Adaptado de [107]).

## 19.5. Robustez en los sistemas biológicos

Una de las cuestiones fundamentales en biología, y particularmente, en biología evolutiva, consiste en entender la *relación existente entre el genotipo, el fenotipo y el ambiente*. El genotipo es la información hereditaria contenida en el genoma que, en un determinado ambiente y como resultado de un proceso de desarrollo, resulta en un fenotipo. En el último siglo hemos aprendido mucho acerca de estas relaciones. Por poner sólo un ejemplo, conocemos el código mediante el cual se almacena la información sobre la secuencia proteica, así como los detalles moleculares de la traducción de este código a proteínas que, a su vez, determinarán en gran medida el fenotipo (Ver Sección 7.4). Estos (y otros) avances sugieren que, por fin, entendemos los detalles del proceso molecular que traduce la información genética en componentes biológicamente activos, es decir, el mecanismo subyacente a la relación genotipo-fenotipo.

No obstante, muchas preguntas siguen sin respuesta. No todos los genes influyen en el fenotipo de igual manera, en igual cantidad y en las mismas condiciones ambientales. Además, el efecto de cada gen sobre el fenotipo depende de manera crucial de su interacción con el resto de componentes. En este contexto, una de las observaciones más sorprendentes es la relativa *invariabilidad del fenotipo*, es decir, su robustez frente a mutaciones, cambios ambientales o fluctuaciones aleatorias en la cantidad de moléculas implicadas en su desarrollo. Ya a mediados del siglo XX, Waddington y Schmalhausen advirtieron este fenómeno, atribuyéndolo a un proceso de “canalización” [124, 125, 146, 147]. Argumentaban que es el propio proceso de desarrollo el que “amortigua” variaciones fenotípicas que podrían ser producidas por diversas fluctuaciones ambientales o variantes genéticas, de manera que los organismos acaban desarrollando un fenotipo relativamente invariante (por ejemplo, los mismos órganos en las mismas posiciones, los mismos tipos celulares discretos, etcétera). Estos autores sostenían que la canalización ha evolucionado debido a una *presión selectiva estabilizante* (es decir, que tiende a eliminar desviaciones fenotípicas). La apreciación clave fue que, además del aspecto genético, es necesario considerar el aspecto *epigenético* (*epi* - sobre y *genético*, es decir, que se construye “sobre la genética”) y su influencia sobre la constitución del fenotipo. Dicho aspecto epigenético surge de las interacciones que se producen entre los productos de los genes.

El trabajo de estos dos autores ha podido ser retomado mucho más tarde, en las últimas décadas, gracias al mejor conocimiento de la genética molecular y del desarrollo, así como del avance tecnológico que ha aportado sofisticadas herramientas de análisis experimental. En 1998, S. Rutherford y S. Lindquist publicaron un trabajo en el que proponían la proteína Hsp90 como capacitador evolutivo (o “agente de canalización”) [123]. Hsp90 es una chaperona molecular, cuyo papel consiste en asistir el correcto plegamiento de otras proteínas. Entre sus dianas se encuentran algunos reguladores del ciclo celular y proteínas participantes en rutas de señalización relacionadas con el desarrollo. En este artículo se demostró que la inhibición de la actividad de dicha proteína durante el desarrollo (por mutaciones o por inhibición química) resulta en una diversidad fenotípica mucho mayor de la observada en ausencia de estas perturbaciones. De manera importante, esto demuestra cómo determinantes que no son ni genéticos ni ambientales pueden influir en el fenotipo (en este caso, en su robustez).

Otros experimentos también han puesto de manifiesto que el alcance y la ubicuidad de la robustez biológica son mucho mayores de lo que se pensaba. La importancia de esta robustez se ilustra de manera muy clara por el hecho de que en la levadura *Saccharomyces cerevisiae* el 80 % de los genes que codifican proteínas funcionales pueden ser eliminados sin comprometer la viabilidad del organismo [46]. En esta sección abordaremos el estudio de la robustez biológica frente a mutaciones desde la perspectiva y la metodología de la Biología de Sistemas. Trataremos en detalle dos ejemplos: la redundancia y la robustez distribuida. Desde el punto de vista biológico, se hará especial énfasis en las fuerzas que permiten la aparición y la estabilidad evolutiva de estos mecanismos. Desde el punto de vista metodológico, trataremos dos aproximaciones típicamente utilizadas en biología de sistemas: los modelos matemáticos

y los computacionales.

### 19.5.1. Un mecanismo básico de robustez: la redundancia por duplicación génica

Uno de los mecanismos de robustez más sencillos con los que cuentan los organismos es la *redundancia por duplicidad génica*. Las duplicaciones génicas son eventos mutacionales relativamente frecuentes en la evolución de los genomas ( $\sim 1$  duplicación por gen y por  $10^6$  años [84]). Tras la duplicación, las mutaciones en una de las copias dejarán de tener efecto, dada la existencia de una “copia de seguridad” adicional que preservará la función. Imaginemos que determinada función puede ser llevada a cabo en un organismo por dos genes redundantes. La pérdida de dicha función ocasiona una pérdida de ‘fitness’ (aptitud,  $s$ ). Pero dado que dos genes codifican para dicha función, la eliminación de uno de ellos no conllevará ningún efecto, mientras la eliminación de ambos a la vez originará una pérdida  $s$ . Este efecto, denominado *epistasis negativa*, fue observado en una alta proporción en los más de 60 pares de genes duplicados testados en *Saccharomyces cerevisiae*, viniendo a confirmar el aumento de la robustez fenotípica por duplicaciones génicas.

Resulta bastante más complicado entender cómo se mantienen los duplicados de manera estable a lo largo del tiempo evolutivo ([145]). Dado que cualquier gen está sometido a una carga mutacional deletérea, para su manutención ésta ha de ser contrarrestada por una presión selectiva que elimine de la población estas mutaciones desventajosas. Esto se conoce como *balance mutación-selección* (ver Capítulo 9). Sin embargo, la redundancia camufla el carácter deletéreo de estas mutaciones. En otras palabras, en una situación de redundancia, las mutaciones que produzcan pérdida de función de una de las copias se comportarán de forma esencialmente neutral a efectos de dinámica poblacional: a largo plazo, una de las copias se inactivará por acumulación y fijación de mutaciones deletéreas, haciendo desaparecer la redundancia.

Esta situación puede examinarse de forma cuantitativa mediante un sencillo modelo [156]. Consideraremos una población haploide compuesta por dos genotipos. El primero posee dos genes que codifican para la misma función, mientras que el segundo sólo posee uno. La tasa de mutación es  $u$  por gen y generación, mientras que el efecto deletéreo de la pérdida de función es  $s$ . En cada generación, un individuo con el genotipo redundante tendrá una probabilidad de pérdida de función de  $u^2$ , con un *fitness* resultante de  $1 - s$ . Tras una generación, por tanto, el *fitness* esperado del primer genotipo será  $f_1 = u^2(1 - s) + 1(1 - u^2) = 1 - su^2$ . Para el segundo genotipo, el *fitness* esperado será  $f_2 = u(1 - s) + 1(1 - u) = 1 - su$ . Aunque aparentemente  $f_1 > f_2$ , en una población finita dos genotipos son efectivamente neutrales si la diferencia de *fitness*  $\delta f = f_1 - f_2 = su(1 - u) \approx su$  es menor que  $1/N_e$ , siendo  $N_e$  el tamaño de población efectivo. Esto es, la condición para el mantenimiento selectivo de la redundancia es  $su > 1/N_e$ , o  $N_e su > 1$ . Incluso para genes esenciales ( $s = 1$ ), es difícil satisfacer esta desigualdad, ya que para la gran mayoría de especies se cumple que  $N_e u < 1$ . Por ejemplo, *Saccharomyces cerevisiae* tiene un tamaño de población efectivo de  $\sim 10^7$  y una tasa de mutación  $\sim 4 \times 10^{-4}$ , lo cual conlleva  $su = 0,4$ .

Este modelo apunta, esencialmente, a que *la redundancia por duplicación es evolutivamente inestable*. La fijación de mutaciones deletéreas en régimen neutral se produce porque la propia redundancia las protege de su exposición fenotípica y selectiva. Parece razonable pensar, entonces, que la estabilidad sólo puede darse si los genes redundantes poseen, además, características adicionales que confieren una ventaja adaptativa extra para cada uno de ellos. Esta presencia de múltiples funciones se denomina *pleiotropía* [104]. Otro escenario posible implica diferentes tasas de mutación y/o eficiencias funcionales. En el caso de que una de las copias tenga mayor tasa de mutación, ésta será la que tienda a desaparecer (dado que la tasa de fijación de mutaciones neutrales depende exclusivamente de la tasa de mutación). No obstante, si esta copia lleva a cabo su función de manera más eficiente, la redundancia podrá conservarse de manera estable [104]). En este caso, una de las copias se mantendrá por su mayor

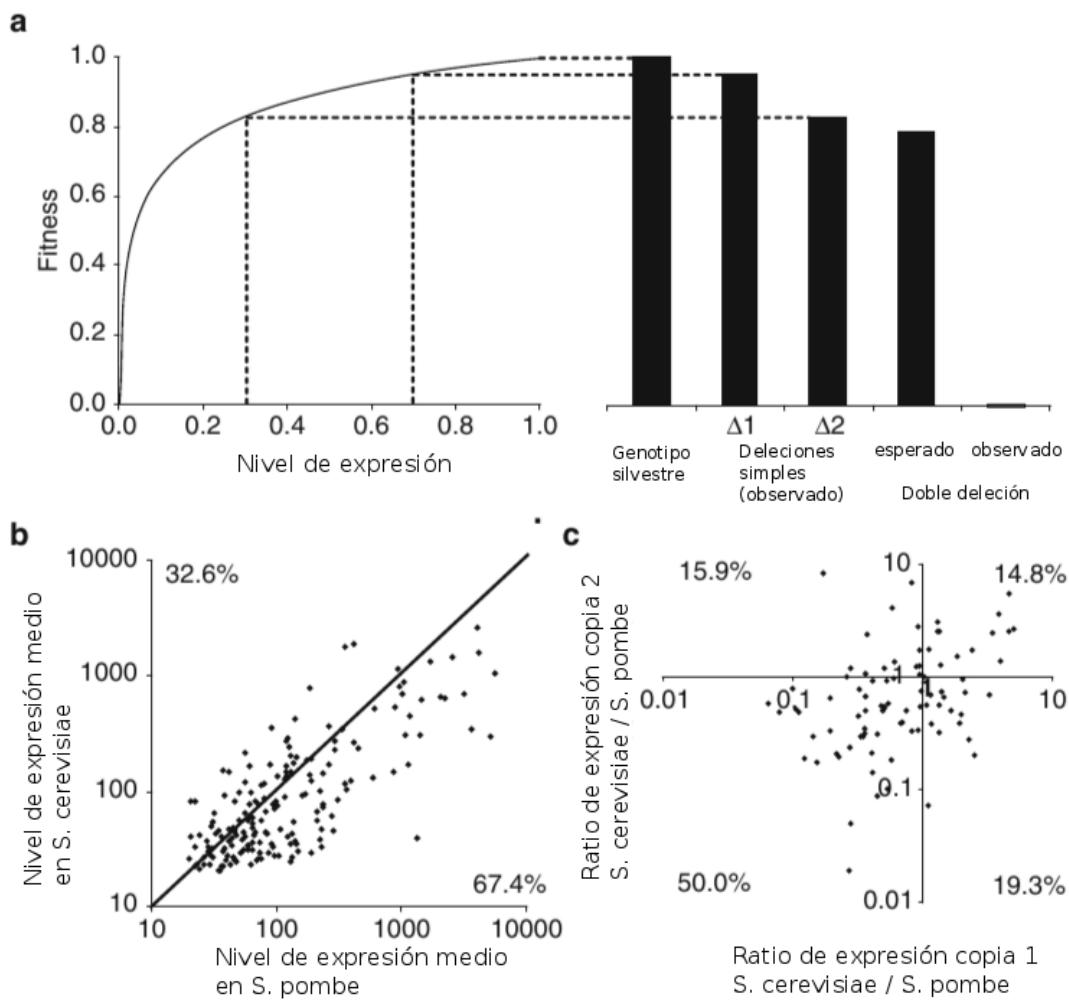
eficiencia funcional; la otra, por su menor tasa de mutación, se mantendrá en los genotipos con la primera copia mutada.

Paradójicamente, la propia robustez, causada por la redundancia (a corto plazo), es la responsable de que la redundancia sea inestable (a largo plazo). Para que ésta sea estable parece necesaria la existencia de un nivel determinado de asimetría. Para ser preservada, cada copia debe aportar una ventaja adicional. En otras palabras, la redundancia puede ser estable sólo si es parcial, y será estable gracias a la parte no redundante. Es decir, de alguna forma, estamos haciendo “trampa”. Pero hasta ahora sólo nos hemos preguntado ‘cómo’ deben ser los genes para que su redundancia sea estable. Otro factor que puede ser importante es el ‘cuánto’. Imaginemos que, tras la duplicación, los genes implicados reducen su nivel de expresión hasta el nivel original antes de la duplicación. En este caso, la pérdida de uno de los alelos conllevaría una pérdida de *fitness*, lo cual posibilitaría la preservación selectiva de la redundancia. Ahora, ¿no se está perdiendo con esto la robustez que se originaría con la redundancia? La respuesta es que sí, pero sólo parcialmente. Esto se debe a que el *fitness* generalmente no depende de manera lineal del nivel de expresión [73] (Figura 19.18A); una reducción del 50 % en el nivel de expresión implica una reducción de *fitness* menor que 50 %. En consecuencia, aunque perder una copia conlleve cierta desventaja, esta sería mucho menor que en caso de que sólo haya una; por tanto, sí que existe cierto grado de robustez. El efecto de reducción del nivel de expresión en genes duplicados ha sido, de hecho, observado experimentalmente. De los 227 genes de *Schizosaccharomyces pombe* que habían experimentado duplicación en *Saccharomyces cerevisiae*, dos tercios habían reducido significativamente su nivel de expresión en este último organismo [117] (Figura 19.18B, C). Por tanto, sí parece que estos aspectos relacionados con el nivel de expresión constituyen un mecanismo importante para la preservación de pares de genes redundantes.

### 19.5.2. Robustez distribuida en el metabolismo

Aún siendo un mecanismo de robustez importante, existen numerosos casos en los que la redundancia no puede explicar la robustez observada. Pongamos por caso la glucosa deshidrogenasa en *E. coli*, una enzima de la ruta de las pentosas fosfato. Una de las principales funciones de esta ruta es producir NADPH, equivalente reductor utilizado en numerosos procesos de biosíntesis. A pesar de que la función de este enzima sea sin duda importante, su eliminación no produce efectos fenotípicamente observables. Si bien esta robustez no puede atribuirse a genes redundantes capaces de llevar a cabo la misma reacción, los cambios que se producen en el resto del sistema son notables. Por ejemplo, la actividad del ciclo de los ácidos tricarboxílicos se incrementa de forma considerable a fin de producir un exceso de NADH; y éste es a su vez transformado en NADPH gracias a un incremento en la actividad transhidrogenasa. Por otro lado, la producción de determinados precursores biosintéticos, otra de las funciones de la ruta de las pentosas fosfato, sigue siendo posible gracias a la entrada de componentes glicolíticos por otro punto en esta ruta. Este caso ejemplifica un tipo de robustez que no se debe a fenómenos de redundancia génica *sensu stricto*, sino más bien a la naturaleza distribuida de los procesos biológicos: en determinadas circunstancias, componentes genuinamente diferentes del sistema pueden asumir las mismas funciones [149].

¿En qué mecanismos se origina, y como se preserva a lo largo de la evolución, la robustez derivada de propiedades globales del sistema? Estas cuestiones pueden estudiarse de manera directa usando modelado metabólico basado en restricciones [151]. Como hemos tratado en el presente capítulo, estos modelos nos permiten examinar la función de reacciones concretas de un organismo en diferentes contextos nutricionales y genéticos. Concretamente, hacen posible examinar de forma detallada los efectos de la delección de un gen sobre el resto del sistema y las re-estructuraciones que se producen como consecuencia, responsables de la robustez observada a nivel fenotípico.



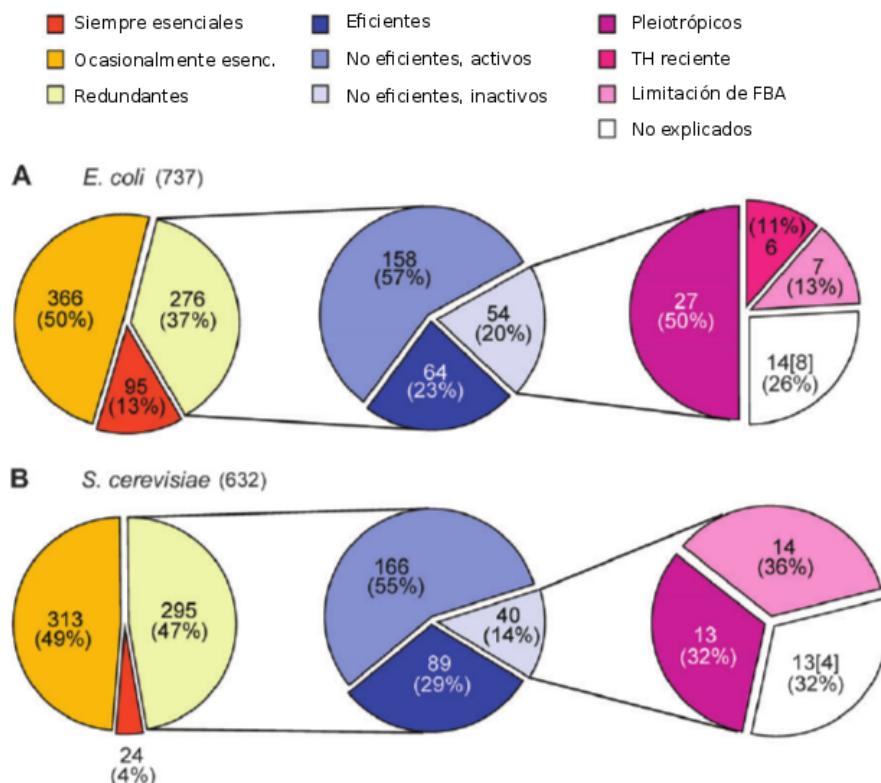
**Figura 19.18:** La reducción en el nivel de expresión puede preservar la preservación evolutiva de duplicados funcionalmente redundantes. A) Dado que el *fitness* es una función cóncava del nivel de expresión, se observa epistasis negativa entre los duplicados con nivel de expresión reducido. En este ejemplo, observamos el caso extremo de letalidad sintética. El *fitness* esperado de la doble delección se obtiene mediante un modelo de efectos multiplicativos. B) Cada punto representa el nivel de expresión medio de los duplicados en *Saccharomyces cerevisiae*, y el nivel de expresión del gen ortólogo en *Schizosaccharomyces pombe*. Se muestra el porcentaje de puntos debajo y encima de la diagonal. C) Ratios de expresión entre *S. cerevisiae* y *S. pombe* para cada par de ortólogos (dos a uno). Adaptado de [117]).

La *primera hipótesis* que plantean estos autores para explicar la robustez frente a la delección de algunas reacciones es que éstas sean esenciales sólo de forma “ocasional” en la naturaleza, donde la composición nutricional del ambiente es variable. Teóricamente, se puede demostrar que aunque la ventaja conferida por determinado gen se manifieste sólo circunstancialmente, en general bastará para que sea preservado en la población [151]. Usando análisis de balance de flujos (FBA), las reacciones que son siempre esenciales pueden ser fácilmente identificadas como aquellas necesarias para el crecimiento en un ambiente que contenga todas las fuentes de carbono utilizables. Este tipo de genes constituyen, sin embargo, una parte muy pequeña del total (Figura 19.19).

Para evaluar la posibilidad de que el resto de genes sean esenciales solamente de manera ocasional, es

necesario simular distintas condiciones con disponibilidad limitada de nutrientes. Por ejemplo, condiciones ambientales con sólo una fuente de carbono, o condiciones generadas con una selección aleatoria de éstas, que simularán los ambientes encontrados por el organismo en la naturaleza. Con una cantidad suficiente de condiciones evaluadas podremos estimar cuántas reacciones se comportan como esenciales de manera ocasional. Este número resulta significativo: tanto en *E. coli* como en *S. cerevisiae*, constituye aproximadamente el 50 % de las reacciones metabólicas.

Aún así, más de un tercio de las reacciones siguen siendo aparentemente prescindibles (Figura 19.19). Una razón posible es que estas reacciones sean más eficientes que sus alternativas en la realización de determinada función metabólica. Esta eficiencia puede detectarse como una disminución de la producción de biomasa tras su eliminación, al ser la alternativa menos eficiente. Los análisis llevados a cabo muestran que las reacciones “eficientes” constituyen más de la mitad de estas reacciones prescindibles. La mayor eficiencia de ciertas reacciones también puede explicar su preservación evolutiva desde un punto de vista teórico (ver arriba).



**Figura 19.19:** Número y fracción de genes esenciales y no esenciales en las redes metabólicas de A) *E. coli* y B) *S. cerevisiae*. Los diagramas de la izquierda y del centro muestran varias explicaciones para la existencia de reacciones no esenciales. Una reacción es siempre u ocasionalmente esencial o si lo es en todas o sólo en algunas condiciones. Las reacciones redundantes, es decir, no esenciales, no lo son en ninguna condición. Una reacción redundante es eficiente si su delección implica una reducción de *fitness* en al menos una condición. Es “inactiva” si es menos efectiva que otras en todas las condiciones, pero tiene flujo en al menos una. En determinadas condiciones (considerando minimización del ajuste metabólico) la delección de esta reacción comportara una reducción inmediata en *fitness* que puede resultar en su conservación evolutiva. Otras posibles causas de conservación, como los artefactos resultantes de la modelización, la existencia de pleiotropía o transferencia horizontal reciente, se muestran en los diagramas de la derecha. Adaptado de [151]).

Los análisis anteriores fueron llevados a cabo usando *modelos basados en restricciones*. El método de optimización usado por este protocolo implica una completa libertad al reorganizar los flujos metabólicos para llegar a la solución óptima. Como hemos señalado, esto implica la asunción de que la célula dispone de una maquinaria regulatoria extremadamente eficaz, lo cual no es una asunción del todo realista. Experimentalmente, se ha demostrado que las células alcanzan primero una situación subóptima, pudiendo evolucionar hasta el óptimo predicho por FBA en caso de que el ambiente se mantenga constante [64]. Pero el genotipo mutante puede desaparecer de la población por su menor capacidad competitiva antes de tener suficiente tiempo para esta evolución. Una posibilidad es que la importancia de algunos genes radique precisamente en esta primera situación subóptima. Para simular este escenario, se puede utilizar un tipo de modelo metabólico basado en restricciones llamado “*minimización del ajuste metabólico*” (MOMA, en sus siglas en inglés). MOMA encuentra una solución sub-óptima minimizando la diferencia entre la distribución de flujos del metabolismo portador de la delección y la distribución óptima de flujos en el metabolismo silvestre. Como hemos visto, que la supresión de una reacción no produzca en ningún ambiente una disminución de la biomasa predicha por FBA quiere decir que existe una alternativa al menos igual de eficiente. Si, por el contrario, produce disminución de biomasa solamente usando MOMA, esto implica que dicho gen está activo en esos ambientes, y además su pérdida puede ocasionar una desventaja competitiva (al menos inicialmente). Esta desventaja puede ser también una razón para la preservación de estas reacciones en el metabolismo.

¿Qué ocurre con el resto de reacciones redundantes, que no son esenciales, eficientes, y ni siquiera activas en el fenotipo silvestre, en ningún ambiente? Una posibilidad es que estén codificadas por *genes pleiotrópicos*, es decir, que codifican para varias reacciones. Si alguna de las reacciones codificadas por un gen pleiotrópico sí perteneciera a uno de los anteriores grupos, esta asociación podría explicar la preservación de las otras reacciones no eficientes ni activas. Una posibilidad adicional es que los genes que codifican estas reacciones se hayan transferido horizontalmente de manera reciente, lo cual se observa para algunas reacciones de *E. coli*, donde la transferencia horizontal sucede frecuentemente. Por último, algunas reacciones pueden ser preservadas por mecanismos imposibles de detectar con FBA, como el contexto regulatorio, las temperaturas extremas, etcétera.

## **19.6. Lecturas adicionales**

### **Introducción a las redes complejas**

Existen multitud de libros introductorios al mundo de las redes complejas, entre ellos, cabe destacar:

- 
- 

Además, muchas revistas de divulgación se han acercado a este campo. Un artículo dedicado a las redes libres de escala y que además es accesible gratuitamente en internet es:

- 

### **Análisis de motivos en redes de regulación**

- 
- 

### **Osciladores bioquímicos**

- 

### **Ruido en expresión genética**

Hay excelentes trabajos de revisión sobre el papel del ruido biológico y la variabilidad en importantes procesos celulares, algunos son:

- 
- 
- 
-

## 19.7. Herramientas computacionales

Programa	Descripción y URL
Cytoscape	Software gratuito para dibujar y visualizar redes complejas. Existen versiones para Linux, Mac y Windows. <a href="http://www.cytoscape.org">http://www.cytoscape.org</a>
Pajek	Software gratuito. Además de dibujar redes complejas, tiene implementados diversos algoritmos para analizarlas. Aunque se desarrolló específicamente para Windows, se puede usar también en Linux y Mac. <a href="http://pajek.imfm.si/doku.php?id=pajek">http://pajek.imfm.si/doku.php?id=pajek</a>
Genetic Network Analyzer	Para el análisis de modelos booleanos [16]. <a href="http://www-helix.inrialpes.fr/gna">http://www-helix.inrialpes.fr/gna</a>
XPPAUT	Para el análisis de sistemas dinámicos y solución de sistemas de ecuaciones diferenciales [39]. <a href="http://www.math.pitt.edu/~bard/xpp/xpp.html">http://www.math.pitt.edu/~bard/xpp/xpp.html</a>
Berkeley Madonna	Modelado y análisis de sistemas dinámicos. <a href="http://www.berkeleymadonna.com">http://www.berkeleymadonna.com</a>
Dizzy	Para la simulación de procesos estocásticos [119]. <a href="http://magnet.systemsbiology.net/software/Dizzy">http://magnet.systemsbiology.net/software/Dizzy</a>
COBRA	Para modelos metabólicos basados en restricciones. Disponible para Python y como librería para Matlab. Ver también el protocolo para el uso de COBRA. <a href="http://opencobra.sourceforge.net">http://opencobra.sourceforge.net</a>

**Tabla 19.3:** Algunos programas útiles para el trabajo con redes biológicas.

## 19.8. Ejercicios

### Modelos cinéticos

**Ejercicio 1. Represión post-transcripcional por sRNAs.** En los mecanismos de regulación estudiados en este capítulo, hemos supuesto que un gen se activa o reprime por la acción de un factor de transcripción que favorece o inhibe la producción de mRNA actuando sobre el promotor de un gen. Tanto en eucariotas como en procariotas, existen otros tipos de mecanismos de regulación. Entre ellos, quizás el más común es la regulación post-transcripcional por RNAs no codificantes: por ejemplo, por microRNAs (miRNAs) en eucariotas [41] o por *small non-coding RNAs* (sRNAs) en bacterias [136]. En ambos casos, una cadena de RNA corta no codificante ( $\sim 20 - 200\text{nts}$ ) se une a un mRNA diana y bien favorece su degradación o evita su traducción, aunque también existen casos de regulación positiva [56]. La regulación post-transcripcional puede producir funciones de regulación diferentes de las funciones de Hill sigmoidales estudiadas en este capítulo.

Considera las siguientes reacciones bioquímicas para un RNA mensajero ( $m$ ), otro sRNA que interacciona con el anterior ( $s$ ) y una proteína  $p$ :

a) Producción:



b)  $s$  y  $m$  se asocian para formar un complejo  $c$ , que también puede disociarse:



c) Degradaciones:



Usando la ley de acción de masas, escribe un sistema de ecuaciones diferenciales para la evolución temporal de cada una de las especies ( $m, s, c$  y  $p$ ). Suponiendo que las reacciones de asociación/disociación son rápidas y  $c$  está en equilibrio, elimina la variable  $c$  y escribe las nuevas ecuaciones para las variables  $s$  y  $m$ . ¿Qué acción tiene la especie  $s$  respecto a la proteína  $p$ , la activa o la reprime?.

Calcula el valor en el estado de equilibrio de dichas variables en función de los parámetros.

Dibuja la concentración de proteína ( $p$ ) en función de su tasa de transcripción  $\alpha_m$ . Observa qué ocurre cuando  $\alpha_m = \alpha_s$ , y discute las diferencias entre esta función respuesta y las funciones de regulación transcripcional explicadas en el capítulo (ver Levine et al. [77]).

## Análisis de motivos en redes de regulación

**Ejercicio 2. Osciladores genéticos de dos componentes.** Representa las interacciones de los esquemas de la Figura 19.8B,C por reacciones bioquímicas como las descritas en las ecuaciones (19.9)-(19.13). Asume que tanto la proteína activadora como la represora forman dímeros cuando actúan como factores transcripcionales (Figura 19.8B), y que ambas se unen como monómeros para formar un complejo degradable en la represión postraduccional (Figura 19.8C). Suponiendo que las reacciones de dimerización y unión/desunión a los promotores son rápidas y están en equilibrio, plantea un sistema de ecuaciones diferenciales para la evolución temporal de las proteínas activadora y represora y sus correspondientes RNAs mensajeros de la Figura 19.8B. Deduce el sistema de ecuaciones análogo para el esquema de la Figura 19.8C y discute sus diferencias con el anterior. ¿Qué esquema crees que dará periodos de oscilación más cortos y por qué?. Suponiendo que la degradación de mRNA es mucho más rápida que la de proteína, usa la aproximación cuasi-estacionaria para reducir el sistema a sólo dos variables.

Usa algún *software* de simulación de sistemas dinámicos (XPPAUT o Berkeley Madonna, ver Herramientas computacionales) para representar la evolución temporal de las concentraciones de proteínas, y estudia en qué condiciones los dos esquemas darán oscilaciones. Sugerencia: Referencia [58].

**Ejercicio 3. Interruptores genéticos multiestables.** En la Figura 19.10C se representan dos sistemas de dos componentes con retroalimentación y autorregulaciones positivas. Representa las interacciones de dichos sistemas como reacciones bioquímicas, en las que cada proteína ( $X$  o  $Y$ ) se puede dimerizar y actuar como un factor de transcripción del promotor complementario o de su propio promotor. Aplica aproximaciones cuasi-estacionarias para los mRNA y reduce a un modelo de dos ecuaciones para la dinámica en la concentración de proteínas, en las que el signo y la fuerza de cada interacción venga representada por un solo parámetro. Como el promotor de cada gen está regulado por su propia proteína y la proteína complementaria, particulariza al caso de que los sitios de unión solapan completamente (puerta lógica ‘XOR’) y supón que todas las fuerzas de interacción simétricas en los componentes  $X$  e  $Y$ . Usando un *software* de análisis de sistemas dinámicos (Sección 19.7), estudia en función de la fuerza de autorregulación positiva en qué condiciones los dos sistemas pueden presentar más de dos estados de equilibrio estables. Sugerencia: Referencia [59].

**Ejercicio 4. Adaptación perfecta.** Hay redes genéticas que pueden adaptar su respuesta a señales (el sistema muestra una respuesta transitoria a un cambio en la señal, aunque su estado de equilibrio es independiente de la fuerza de la señal). Nuestro propio sentido del olfato funciona de esta manera: un estímulo olfativo induce un cambio en la actividad de los receptores olfativos, pero esta actividad pronto retorna a los valores originales incluso si el estímulo persiste. Uno de los módulos más simples que logran adaptación es el siguiente:

$$\begin{aligned}\frac{dR}{dt} &= k_1S - k_2X \cdot R \\ \frac{dX}{dt} &= k_3S - k_4X\end{aligned}\tag{19.46}$$

donde  $S$  es la señal y  $R$  el *output*(respuesta) del sistema. Haz un dibujo esquemático con las diferentes especies moleculares ( $S$ ,  $R$  y  $X$ ) como nodos y con las correspondientes interacciones y su signos. ¿Existe algún tipo de retroalimentación?. Calcula el estado de equilibrio del *output*, ¿por qué el sistema muestra adaptación?.

Haz un programa para resolver las ecuaciones anteriores y pinta las trayectorias en función del tiempo para  $k_1 = k_2 = 2$ ,  $k_3 = k_4 = 1$ , y para  $S$  una función escalón que cambia bruscamente entre 0 y 1.

**Ejercicio 5. Percepción de estímulo relativa o ley de Weber.** La percepción de algunos estímulos sensoriales, como la intensidad sonora o luminosa, obedece aproximadamente la ley de Weber (Ernst Weber, 1834, quien la postuló para explicar sus experimentos sobre la percepción de peso):

$$\frac{\Delta S_{min}}{S_f} = c$$

donde  $\Delta S_{min}$  es el mínimo cambio detectable en el estímulo,  $S_f$  es el nivel ‘de fondo’(*background*) del estímulo, y  $c$  es una constante. Esta ley nos dice que algunos estímulos no son percibidos en términos absolutos sino relativos, es decir, que percibimos su cambio en magnitud relativo a la intensidad del estímulo de ‘fondo’(pensad en el volumen de voz que tenéis que emplear para ser escuchados por alguien cercano si estáis en una iglesia, o si estáis en un bar con la música alta). Uno de los circuitos genéticos más sencillos que pueden dar lugar a este tipo de detección ‘relativa’ es el dado por las siguientes ecuaciones (Ref. [53]):

$$\begin{aligned}\frac{dY}{dt} &= \beta_1 X - \alpha_1 Y \\ \frac{dZ}{dt} &= \beta_2 \frac{X}{Y} - \alpha_2 Z,\end{aligned}\tag{19.47}$$

donde  $X$  es una señal (por ejemplo la cantidad de un factor de transcripción),  $Y$  una especie intermedia y  $Z$  el *output*. ¿Qué tipo de interacciones hay entre los nodos  $X$ ,  $Y$  y  $Z$ ? Usando los modelos básicos de regulación, escribe una subrutina para simular la respuesta de una proteína  $Z$  a un factor de transcripción  $X$  con una activación simple ( $X \rightarrow Z$ ). Supón que  $X$  es una función escalón que pasa de  $X_0 = 1$  a  $X_1 = 5$  a un tiempo dado. Observa cómo cambia  $Z$  en función del tiempo. Ahora pinta  $Z$  en función del tiempo para  $X_0 = 10$ ,  $X_1 = 50$ . Haz lo mismo para las variables ( $Y, Z$ ) del sistema de ecuaciones (19.47) usando la misma función escalón en  $X$  (Figura 2B de Goentoro et al.[53]). Discute similitudes y diferencias con el sistema del problema anterior sobre adaptación.

## Ruido en expresión genética

**Ejercicio 6.** Usando el esquema de expresión genética dado en la Figura 19.15, y la ley de acción de masas, plantea un sistema de ecuaciones diferenciales ordinarias para la dinámica de las cuatro especies moleculares (promotor activo A, promotor inactivo R, mRNA M y proteína P). Suponiendo que la dinámica de activación/desactivación del promotor es más rápida que el resto de las reacciones, usa la aproximación quasi-estacionaria y la conservación de la cantidad total de promotor ( $A + R = 1$ ), para deducir las Ecs. (19.41). Programa el algoritmo de Gillespie original [50] o usa un *software* de solución de sistemas estocásticos como Dizzy (Sección 19.7) para representar las trayectorias estocásticas de mRNA( $M$ ) y proteína ( $P$ ) en las condiciones de las Figuras 19.16A, B. Compara con las soluciones del sistema determinista, Ecs. (19.41). Obtén las distribuciones en cantidad de proteína (propagando trayectorias a tiempos largos) y calcula el coeficiente de variación cambiando la tasa de degradación del mRNA. Representa el coeficiente de variación en función del ‘parámetro de ráfaga’, Ec. (19.42) y compara con la expresión teórica dada en la Referencia [140].

## 19.9. Bibliografía

- [1] G. K. Ackers, A. D. Johnson, and M. A. Shea. Quantitative model for gene regulation by lambda phage repressor. *Proc Natl Acad Sci U S A*, 79(4):1129–33, feb 1982.
- [2] D. Adalsteinsson, D. McMillen, and T. C. Elston. Biochemical network stochastic simulator (bionets): software for stochastic modeling of biochemical networks. *BMC Bioinformatics*, 5:24, mar 2004.
- [3] J. Aguirre, J. M. Buldú, M. Stich, and S. C. Manrubia. Topological structure of the space of phenotypes: The case of RNA neutral networks. *PLoS ONE*, 6:e26324, 2011.
- [4] R. Albert. Scale-free networks in cell biology. *Journal of cell science*, 118:4947–57, 2005.
- [5] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRCpress, 2006.
- [6] U. Alon. Network motifs: theory and experimental approaches. *Nat Rev Genet*, 8(6):450–61, 2007.
- [7] U. Alon, M. G. Surette, N. Barkai, and S. Leibler. Robustness in bacterial chemotaxis. *Nature*, 397(6715):168–71, jan 1999.
- [8] G. Bagler and S. Sinha. Assortative mixing in protein contact networks and protein folding kinetics. *Bioinformatics*, 23:1760–1767, 2007.
- [9] N. Balaskas, A. Ribeiro, J. Panovska, E. Dessaoud, N. Sasai, K. M. Page, J. Briscoe, and V. Ribes. Gene regulatory logic for reading the sonic hedgehog signaling gradient in the vertebrate neural tube. *Cell*, 148(1-2):273–84, jan 2012.
- [10] G. Balazsi, A. van Oudenaarden, and J. J. Collins. Cellular decision making and biological noise: from microbes to mammals. *Cell*, 144(6):910–25, 2011.
- [11] A. Bar-Even, J. Paulsson, N. Maheshri, M. Carmi, E. O’Shea, Y. Pilpel, and N. Barkai. Noise in protein expression scales with natural protein abundance. *Nat Genet*, 38(6):636–43, jun 2006.
- [12] A. L. Barabasi and E. Bonabeau. Scale-free networks. *Sci Am*, 288(5):60–9, 2003.
- [13] A. L. Barabási and Z. N. Oltvai. Network biology: understanding the cell’s functional organization. *Nature reviews. Genetics*, 5(2):101–13, 2004.
- [14] A.-L. Barabási and D. J. Watts. *The Structure and Dynamics of Networks*:. Princeton University Press, Princeton, 1 edition edition, May 2006.
- [15] N. Barkai and S. Leibler. Circadian clocks limited by noise. *Nature*, 403(6767):267–8, jan 2000.
- [16] G. Batt, B. Besson, P. E. Ciron, H. de Jong, E. Dumas, J. Geiselmann, R. Monte, P. T. Monteiro, M. Page, F. Rechenmann, and D. Ropers. Genetic network analyzer: a tool for the qualitative modeling and simulation of bacterial regulatory networks. *Methods Mol Biol*, 804:439–62, 2012.
- [17] S. Ben-Tabou de-Leon and E. H. Davidson. Modeling the dynamics of transcriptional gene regulatory networks for animal development. *Dev Biol*, 325(2):317–28, jan 2009.
- [18] H. C. Berg and P. M. Tedesco. Transient response to chemotactic stimuli in escherichia coli. *Proc Natl Acad Sci U S A*, 72(8):3235–9, aug 1975.
- [19] L. Bintu, N. E. Buchler, H. G. Garcia, U. Gerland, T. Hwa, J. Kondev, T. Kuhlman, and R. Phillips. Transcriptional regulation by the numbers: applications. *Curr Opin Genet Dev*, 15(2):125–35, apr 2005.
- [20] L. Bintu, N. E. Buchler, H. G. Garcia, U. Gerland, T. Hwa, J. Kondev, and R. Phillips. Transcriptional regulation by the numbers: models. *Curr Opin Genet Dev*, 15(2):116–24, apr 2005.
- [21] W. J. Blake, M. KAErn, C. R. Cantor, and J. J. Collins. Noise in eukaryotic gene expression. *Nature*, 422(6932):633–7, apr 2003.
- [22] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10:1742–5468, P10008, 2008.
- [23] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang. Complex networks: Structure and dynamics. *Phys. Rep.*, 424, 2006.

- [24] L. A. Boyer, T. I. Lee, M. F. Cole, S. E. Johnstone, S. S. Levine, J. P. Zucker, M. G. Guenther, R. M. Kumar, H. L. Murray, R. G. Jenner, D. K. Gifford, D. A. Melton, R. Jaenisch, and R. A. Young. Core transcriptional regulatory circuitry in human embryonic stem cells. *Cell*, 122(6):947–56, sep 2005.
- [25] A. Brock, H. Chang, and S. Huang. Non-genetic heterogeneity—a mutation-independent driving force for the somatic evolution of tumours. *Nat Rev Genet*, 10(5):336–42, may 2009.
- [26] N. E. Buchler, U. Gerland, and T. Hwa. On schemes of combinatorial transcription logic. *Proc Natl Acad Sci U S A*, 100(9):5136–41, apr 2003.
- [27] J. M. Buldú, R. Bajo, F. Maestú, N. Castellanos, I. Leyva, and et al. et al. Reorganization of functional networks in mild cognitive impairment. *PLoS ONE*, 6:e19584, 2011.
- [28] Z. Burda, A. Krzywicki, O. C. Martin, and M. Zagorski. Motifs emerge from function in model gene regulatory networks. *Proc Natl Acad Sci U S A*, 108(42):17263–8, oct 2011.
- [29] P. J. Choi, L. Cai, K. Frieda, and X. S. Xie. A stochastic single-molecule event triggers phenotype switching of a bacterial cell. *Science*, 322(5900):442–6, oct 2008.
- [30] M. I. Davidich and S. Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One*, 3(2):e1672, 2008.
- [31] E. H. Davidson. Emerging properties of animal gene regulatory networks. *Nature*, 468(7326):911–20, dec 2010.
- [32] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, 2005.
- [33] J. Dunne, R. Williams, and N. Martinez. Food-web structure and network theory: The role of connectance and size. *Proc. Natl. Acad. Sci. USA*, 99:12917–12922, 2002.
- [34] V. Eguíluz, D. Chialvo, G. Cecchi, M. Baliki, and A. Apkarian. Scale-Free Brain Functional Networks. *Physical Review Letters*, 94:1–4, 2005.
- [35] A. Eldar and M. B. Elowitz. Functional roles for noise in genetic circuits. *Nature*, 467(7312):167–73, 2010.
- [36] J. Elf and M. Ehrenberg. Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome Res*, 13(11):2475–84, nov 2003.
- [37] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8, jan 2000.
- [38] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–6, aug 2002.
- [39] B. Ermentrout. *Simulating, Analyzing, and Animating Dynamical Systems*. Society for Industrial and Applied Mathematics, 2002.
- [40] A. M. Feist and B. O. Palsson. The biomass objective function. *Current opinion in microbiology*, 13(3):344–349, 2010.
- [41] A. S. Flynt and E. C. Lai. Biological principles of microRNA-mediated regulation: shared themes amid diversity. *Nat Rev Genet*, 9(11):831–42, nov 2008.
- [42] S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, 2010.
- [43] S. Gama-Castro et al. Regulondb version 7.0: transcriptional regulation of escherichia coli k-12 integrated within genetic sensory response units (gensor units). *Nucleic Acids Res*, 39(Database issue):D98–105, jan 2011.
- [44] C. W. Gardiner. *Handbook of stochastic methods*. Springer-Verlag, Berlin, 1985.
- [45] A. Garg, K. Mohanram, A. Di Cara, G. De Micheli, and I. Xenarios. Modeling stochasticity and robustness in gene regulatory networks. *Bioinformatics*, 25(12):i101–9, jun 2009.
- [46] G. Giaever, A. M. Chu, L. Ni, C. Connelly, L. Riles, S. Veronneau, S. Dow, A. Lucau-Danila, K. Anderson, B. Andre, et al. Functional profiling of the saccharomyces cerevisiae genome. *Nature*, 418(6896):387–391, 2002.
- [47] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The journal of physical chemistry A*, 104:1876–1889, 2000.

- [48] D. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.
- [49] D. Gillespie. The chemical langevin equation. *The journal of chemical physics*, 113(1):297–306, 2000.
- [50] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.
- [51] L. Giot, J. S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y. L. Hao, C. E. Ooi, B. Godwin, E. Vitols, and et al. et al. A protein interaction map of *Drosophila melanogaster*. *Science*, 302:1727–1736, 2003.
- [52] F. Giudicelli, E. M. Ozbudak, G. J. Wright, and J. Lewis. Setting the tempo in development: an investigation of the zebrafish somite clock mechanism. *PLoS Biol*, 5(6):e150, jun 2007.
- [53] L. Goentoro, O. Shoval, M. W. Kirschner, and U. Alon. The incoherent feedforward loop can provide fold-change detection in gene regulation. *Mol Cell*, 36(5):894–9, dec 2009.
- [54] A. Goldbeter. Computational approaches to cellular rhythms. *Nature*, 420(6912):238–45, nov 2002.
- [55] T. Graf and T. Enver. Forcing cells to change lineages. *Nature*, 462(7273):587–94, dec 2009.
- [56] R. Guantes, B. Cayrol, F. Busi, and V. Arluison. Positive regulatory dynamics by a small noncoding rna: speeding up responses under temperature stress. *Mol Biosyst*, 8(6):1707–15, jun 2012.
- [57] R. Guantes, J. Estrada, and J. F. Poyatos. Trade-offs and noise tolerance in signal detection by genetic circuits. *PLoS One*, 5(8):e12314, 2010.
- [58] R. Guantes and J. F. Poyatos. Dynamical principles of two-component genetic oscillators. *PLoS Comput Biol*, 2(3):e30, mar 2006.
- [59] R. Guantes and J. F. Poyatos. Multistable decision switches for flexible control of epigenetic differentiation. *PLoS Comput Biol*, 4(11):e1000235, nov 2008.
- [60] N. Guelzim, S. Bottani, P. Bourgine, and F. Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature genetics*, 31(1):60–3, 2002.
- [61] A. Gursoy, O. Keskin, and R. Nussinov. Topological properties of protein interaction networks from a structural perspective. *Biochemical Society transactions*, 36(Pt 6):1398–403, 2008.
- [62] P. Holme and J. Saramäki. Temporal networks. *Phys. Rep.*, 519:97–125, 2012.
- [63] M. Huxham, S. Beaney, and D. Raffaelli. Do parasites reduce the chances of triangulation in a real food web? *Oikos*, 76:284–300, 1996.
- [64] R. U. Ibarra, J. S. Edwards, and B. O. Palsson. *Escherichia coli k-12* undergoes adaptive evolution to achieve in silico predicted optimal growth. *Nature*, 420(6912):186–189, 2002.
- [65] P. J. Ingram, M. P. H. Stumpf, and J. Stark. Network motifs: structure does not determine function. *BMC Genomics*, 7:108, 2006.
- [66] H. Jeong, S. Mason, A. L. Barabási, and Z. N. Oltvai. Fethality and centrality in protein networks. *Nature*, 411:41–42, 2001.
- [67] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, oct 2000.
- [68] M. Kaern, T. C. Elston, W. J. Blake, and J. J. Collins. Stochasticity in gene expression: from theories to phenotypes. *Nat Rev Genet*, 6(6):451–64, jun 2005.
- [69] S. Kaplan, A. Bren, A. Zaslaver, E. Dekel, and U. Alon. Diverse two-dimensional input functions control bacterial sugar genes. *Mol Cell*, 29(6):786–92, mar 2008.
- [70] K. J. Kauffman, P. Prakash, and J. S. Edwards. Advances in flux balance analysis. *Current opinion in biotechnology*, 14(5):491–496, 2003.
- [71] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J Theor Biol*, 22(3):437–67, mar 1969.
- [72] K. Klemm and S. Bornholdt. Stable and unstable attractors in boolean networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 72(5 Pt 2):055101, nov 2005.

- [73] F. A. Kondrashov and E. V. Koonin. A common framework for understanding the origin of genetic dominance and evolutionary fates of gene duplications. *Trends in Genetics*, 20(7):287–290, 2004.
- [74] S. Y. Lee, D.-Y. Lee, and T. Y. Kim. Systems biotechnology for strain improvement. *TRENDS in Biotechnology*, 23(7):349–358, 2005.
- [75] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in *saccharomyces cerevisiae*. *Science*, 298(5594):799–804, oct 2002.
- [76] A. Levchenko and P. A. Iglesias. Models of eukaryotic gradient sensing: application to chemotaxis of amoebae and neutrophils. *Biophys J*, 82(1 Pt 1):50–63, jan 2002.
- [77] E. Levine, Z. Zhang, T. Kuhlman, and T. Hwa. Quantitative characteristics of gene regulation by small rna. *PLoS Biol*, 5(9):e229, sep 2007.
- [78] J. Lewis. Autoinhibition with transcriptional delay: a simple mechanism for the zebrafish somitogenesis oscillator. *Curr Biol*, 13(16):1398–408, aug 2003.
- [79] D. Li, I. Leyva, J. A. Almendral, I. Sendina-Nadal, J. M. Buldú, S. Havlin, and S. Boccaletti. Synchronization interfaces and overlapping communities in complex networks. *Phys. Rev. Lett.*, 101:168701, 2008.
- [80] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang. The yeast cellcycle network is robustly designed. *Proc Natl Acad Sci U S A*, 101(14):4781–6, apr 2004.
- [81] S. Li and et al. et al. A map of the interactome network of the metazoan *C. elegans*. *Science*, 303:540–543, 2004.
- [82] R. Losick and C. Desplan. Stochasticity and cell fate. *Science*, 320(5872):65–8, apr 2008.
- [83] N. M. Luscombe, M. M. Babu, H. Yu, M. Snyder, S. A. Teichmann, and M. Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431(7006):308–12, 2004.
- [84] M. Lynch. *The origins of genome architecture*. Sinauer Associates, 2007.
- [85] W. Ma, A. Trusina, H. El-Samad, W. A. Lim, and C. Tang. Defining network topologies that can achieve biochemical adaptation. *Cell*, 138(4):760–73, aug 2009.
- [86] S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proc Natl Acad Sci U S A*, 100(21):11980–5, oct 2003.
- [87] S. Mangan, S. Itzkovitz, A. Zaslaver, and U. Alon. The incoherent feed-forward loop accelerates the response-time of the gal system of *escherichia coli*. *J Mol Biol*, 356(5):1073–81, mar 2006.
- [88] S. Mangan, A. Zaslaver, and U. Alon. The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. *J Mol Biol*, 334(2):197–204, nov 2003.
- [89] N. Martinez. Artifacts or attributes? Effects of resolution on the Little Rock Lake food web. *Ecological Monographs*, 61:367–392, 1991.
- [90] S. Maslov, K. Sneppen, and N. York. Specificity and stability in topology of protein networks. *Nature*, pages 1–17, 2002.
- [91] A. E. Mayo, Y. Setty, S. Shavitt, A. Zaslaver, and U. Alon. Plasticity of the cis-regulatory input function of a gene. *PLoS Biol*, 4(4):e45, apr 2006.
- [92] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824, 2002.
- [93] J. M. Montoya, S. L. Pimm, and R. V. Solé. Ecological networks and their fragility. *Nature*, 442:259–264, 2006.
- [94] J. M. Montoya and R. V. Solé. Small world patterns in food webs. *J. Theor. Biol.*, 214:405–412, 2002.
- [95] M. K. Morris, J. Saez-Rodriguez, P. K. Sorger, and D. A. Lauffenburger. Logic-based models for the analysis of cell signaling networks. *Biochemistry*, 49(15):3216–24, apr 2010.
- [96] J. R. S. Newman, S. Ghaemmaghami, J. Ihmels, D. K. Breslow, M. Noble, J. L. DeRisi, and J. S. Weissman. Single-cell proteomic analysis of *s. cerevisiae* reveals the architecture of biological noise. *Nature*, 441(7095):840–6, jun 2006.

- [97] M. Newman. *Networks: An Introduction*. OUP Oxford, Oxford ; New York, Mar. 2010.
- [98] M. E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20):1–4, 2002.
- [99] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2002.
- [100] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.
- [101] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.
- [102] B. Novak and J. J. Tyson. Design principles of biochemical oscillators. *Nat Rev Mol Cell Biol*, 9(12):981–91, 2008.
- [103] A. Novick and M. Weiner. Ezyme induction as an all-or-none phenomenon. *Proc Natl Acad Sci U S A*, 43(7):553–66, jul 1957.
- [104] M. A. Nowak, M. C. Boerlijst, J. Cooke, and J. M. Smith. Evolution of genetic redundancy. *Nature*, 388(6638):167–171, 1997.
- [105] D. T. Odom, N. Zizlsperger, D. B. Gordon, G. W. Bell, N. J. Rinaldi, H. L. Murray, and T. L. Volkert. Control of pancreas and liver gene expression by hnf transcription factors. *Science*, 303(5662):1378–81, feb 2004.
- [106] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. Palsson. A comprehensive genome-scale reconstruction of escherichia coli metabolism–2011. *Mol Syst Biol*, 7:535, 2011.
- [107] J. D. Orth, I. Thiele, and B. Ø. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.
- [108] E. M. Ozbudak, M. Thattai, I. Kurtser, A. D. Grossman, and A. van Oudenaarden. Regulation of noise in the expression of a single gene. *Nat Genet*, 31(1):69–73, may 2002.
- [109] A. Palacín, L. A. Rivas, C. Gómez-Casado, J. Aguirre, and *et al.* The involvement of thaumatin-like proteins in plant food cross-reactivity: A multicenter study using a protein microarray. *PLoS ONE*, 7:e44088, 2012.
- [110] S. Pandey, R.-S. Wang, L. Wilson, S. Li, Z. Zhao, T. E. Gookin, S. M. Assmann, and R. Albert. Boolean modeling of transcriptome data reveals novel modes of heterotrimeric g-protein action. *Mol Syst Biol*, 6:372, jun 2010.
- [111] J. Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–8, jan 2004.
- [112] L. Pelkmans. Using cell-to-cell variability—a new era in molecular biology. *Science*, 336(6080):425–6, apr 2012.
- [113] J. R. Pomerening, S. Y. Kim, and J. E. Ferrell. Systems-level dissection of the cell-cycle oscillator: bypassing positive feedback produces damped oscillations. *Cell*, 122(4):565–78, aug 2005.
- [114] P. Pons and M. Latapy. Computing communities in large networks using random walks. *J. of Graph Alg. and App.*, 10:191–218, 2006.
- [115] M. Ptashne. *A genetic switch: gene control and phage Lambda*. Blackwell Science, Oxford, 1986.
- [116] J. E. Purvis, K. W. Karhohs, C. Mock, E. Batchelor, A. Loewer, and G. Lahav. p53 dynamics control cell fate. *Science*, 336(6087):1440–4, jun 2012.
- [117] W. Qian, B.-Y. Liao, A. Y.-F. Chang, and J. Zhang. Maintenance of duplicate genes and their functional redundancy by reduced expression. *Trends in Genetics*, 26(10):425–430, 2010.
- [118] A. Raj and A. van Oudenaarden. Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell*, 135(2):216–26, 2008.
- [119] S. Ramsey, D. Orrell, and H. Bolouri. Dizzy: stochastic simulation of large-scale genetic regulatory networks. *J Bioinform Comput Biol*, 3(2):415–36, apr 2005.
- [120] J. M. Raser and E. K. O’Shea. Control of stochasticity in eukaryotic gene expression. *Science*, 304(5678):1811–4, jun 2004.
- [121] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science (New York, N.Y.)*, 297(5586):1551–5, 2002.
- [122] M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *PNAS*, 104:7327, 2007.

- [123] S. L. Rutherford and S. Lindquist. Hsp90 as a capacitor for morphological evolution. *Nature*, 396(6709):336–342, 1998.
- [124] I. Schmalhausen. Organism as a whole in individual and historical development. *Akad. Nauk SSSR, Moscow*, 1938.
- [125] I. Schmalhauzen. *Factors of Evolution: The Theory of Stabilizing Selection*. Blakiston Company, 1949.
- [126] R. Schuetz, N. Zamboni, M. Zampieri, M. Heinemann, and U. Sauer. Multidimensional optimality of microbial metabolism. *Science*, 336(6081):601–604, 2012.
- [127] D. Segre, D. Vitkup, and G. M. Church. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences*, 99(23):15112–15117, 2002.
- [128] M. A. Shea and G. K. Ackers. The or control system of bacteriophage lambda. a physical-chemical model for gene regulation. *J Mol Biol*, 181(2):211–30, jan 1985.
- [129] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nat Genet*, 31(1):64–8, may 2002.
- [130] M. S. Sherman and B. A. Cohen. Thermodynamic state ensemble models of cis-regulation. *PLoS Comput Biol*, 8(3):e1002407, 2012.
- [131] O. Shoval, H. Sheftel, G. Shinar, Y. Hart, O. Ramote, A. Mayo, E. Dekel, K. Kavanagh, and U. Alon. Evolutionary trade-offs, pareto optimality, and the geometry of phenotype space. *Science*, 336(6085):1157–1160, 2012.
- [132] A. L. Slusarczyk, A. Lin, and R. Weiss. Foundations for the design and implementation of synthetic genetic circuits. *Nat Rev Genet*, 13(6):406–20, 2012.
- [133] B. Snijder and L. Pelkmans. Origins of regulated cell-to-cell variability. *Nat Rev Mol Cell Biol*, 12(2):119–25, 2011.
- [134] L.-H. So, A. Ghosh, C. Zong, L. A. Sepúlveda, R. Segev, and I. Golding. General properties of transcriptional time series in escherichia coli. *Nat Genet*, 43(6):554–60, jun 2011.
- [135] S. L. Spencer, S. Gaudet, J. G. Albeck, J. M. Burke, and P. K. Sorger. Non-genetic origins of cell-to-cell variability in trail-induced apoptosis. *Nature*, 459(7245):428–32, may 2009.
- [136] G. Storz, J. Vogel, and K. M. Wassarman. Regulation by small rnas in bacteria: expanding frontiers. *Mol Cell*, 43(6):880–91, sep 2011.
- [137] J. Stricker, S. Cookson, M. R. Bennett, W. H. Mather, L. S. Tsimring, and J. Hasty. A fast, robust and tunable synthetic gene oscillator. *Nature*, 456(7221):516–9, nov 2008.
- [138] R. Tanaka. Scale-rich metabolic networks. *Phys. Rev. Lett.*, 94:1168101, 2005.
- [139] Y. Taniguchi, P. J. Choi, G.-W. Li, H. Chen, M. Babu, J. Hearn, A. Emili, and X. S. Xie. Quantifying e. coli proteome and transcriptome with single-molecule sensitivity in single cells. *Science*, 329(5991):533–8, jul 2010.
- [140] M. Thattai and A. van Oudenaarden. Intrinsic noise in gene regulatory networks. *Proc Natl Acad Sci U S A*, 98(15):8614–9, jul 2001.
- [141] T.-L. To and N. Maheshri. Noise can induce bimodality in positive transcriptional feedback loops without bistability. *Science*, 327(5969):1142–5, feb 2010.
- [142] T. Y.-C. Tsai, Y. S. Choi, W. Ma, J. R. Pomerening, C. Tang, and J. E. Ferrell. Robust, tunable biological oscillations from interlinked positive and negative feedback loops. *Science*, 321(5885):126–9, jul 2008.
- [143] J. J. Tyson, K. C. Chen, and B. Novak. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr Opin Cell Biol*, 15(2):221–31, 2003.
- [144] N. G. van Kampen. *Stochastic processes in physics and chemistry*. Elsevier, Amsterdam, 2004.
- [145] T. Vavouri, J. I. Semple, and B. Lehner. Widespread conservation of genetic redundancy during a billion years of eukaryotic evolution. *Trends in Genetics*, 24(10):485–488, 2008.
- [146] C. Waddington. *The strategy of the genes: a discussion of some aspects of theoretical biology*. Allen & Unwin, 1957.
- [147] C. H. Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150(3811):563–565, 1942.

- [148] A. Wagner. The yeast protein interaction network evolves rapidly and contains few redundant duplicate genes. *Mol. Biol. Evol.*, 18:1283–1292, 2001.
- [149] A. Wagner. Distributed robustness versus redundancy as causes of mutational robustness. *Bioessays*, 27(2):176–188, 2005.
- [150] A. Wagner and D. A. Fell. The small world inside large metabolic networks. *Proceedings. Biological sciences / The Royal Society*, 268(1478):1803–10, 2001.
- [151] Z. Wang and J. Zhang. Abundant indispensable redundancies in cellular metabolic networks. *Genome Biology and Evolution*, 1:23–33, 2009.
- [152] S. Wuchty, E. Ravasz, and A. L. Barabási. The Architecture of Biological Networks, in T.S. Deisboeck, J. Yasha Kresh and T.B. Kepler (eds.). *Complex Systems in Biomedicine* (Kluwer Academic Publishing, New York, 2003).
- [153] W. Xiong and J. E. Ferrell. A positive-feedback-based bistable 'memory module' that governs a cell fate decision. *Nature*, 426(6965):460–5, nov 2003.
- [154] S.-H. Yook, Z. N. Oltvai, and A. L. Barabási. Functional and topological characterization of protein interaction networks. *Proteomics*, 4(4):928–42, 2004.
- [155] A. Zaslaver, A. E. Mayo, R. Rosenberg, P. Bashkin, H. Sberro, M. Tsalyuk, M. G. Surette, and U. Alon. Just-in-time transcription program in metabolic pathways. *Nat Genet*, 36(5):486–91, May 2004.
- [156] J. Zhang. Genetic redundancies and their evolutionary maintenance. In O. S. Soyer, editor, *Evolutionary Systems Biology*, volume 751 of *Advances in Experimental Medicine and Biology*, pages 279–300. Springer New York, 2012.

