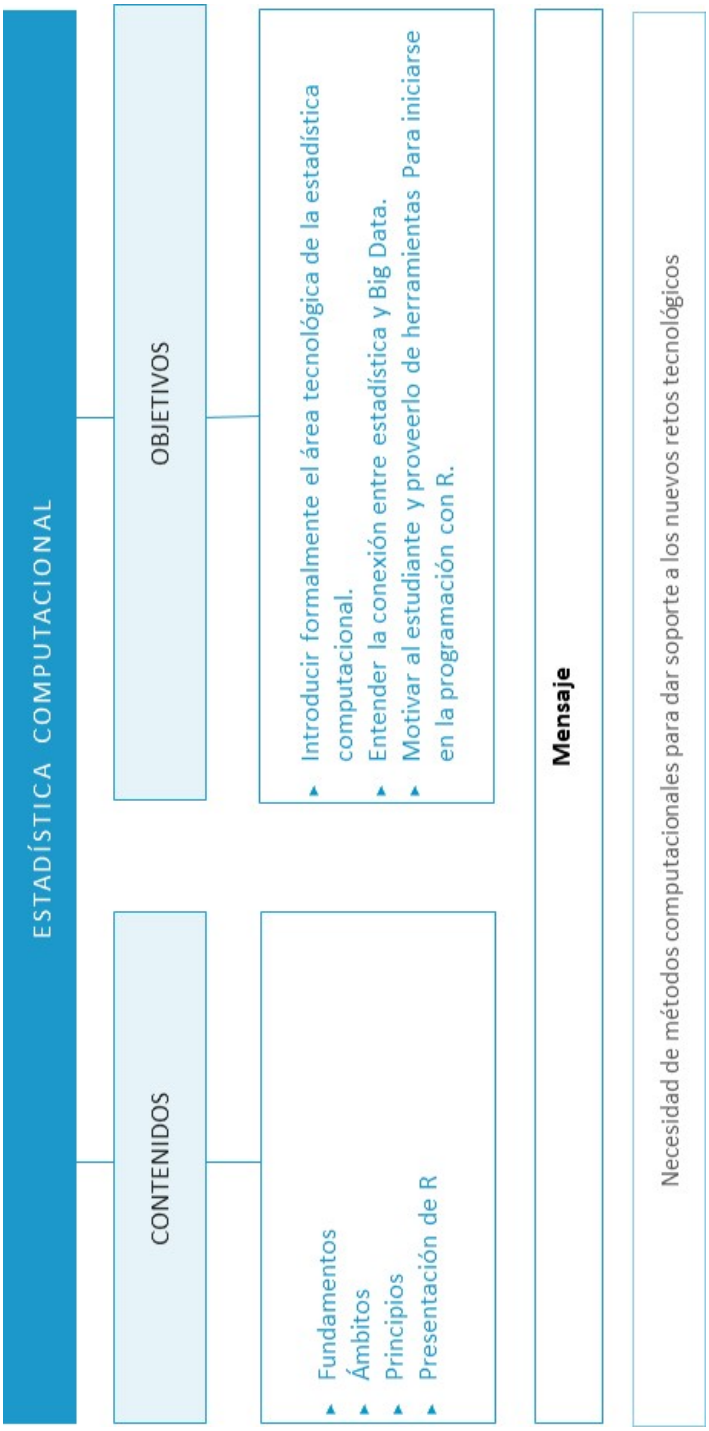
Análisis e Interpretación de Datos

Tema 2. Estadística computacional

Índice

Esquema



2.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las **Ideas clave** que encontrarás a continuación.

En este tema se introduce el área de estadística computacional. Se presenta una visión general del estado en esta ciencia emergente, tras la convergencia de áreas ya consolidadas como las matemáticas, estadística e informática. Tras presentar los **principios básicos** que definen la estadística computacional, se comparte una panorámica sobre las **áreas donde se está aplicando** esta nueva herramienta técnica, incluso podría decirse, áreas técnicas nuevas que se consolidan apoyadas en estrategias de lo que se conoce como **estadística computacional**. Algunas áreas son de reciente incorporación incluso al argot técnico, como inteligencia artificial, minería de datos y *Machine Learning*.

A continuación, se introduce el **lenguaje de programación estadística R** y se presentan sus elementos básicos para un uso a nivel principiante. Estos elementos no pretenden de ninguna manera ser un manual de uso para iniciarse, siquiera, en el *software*, algo imposible de abarcar en solo un apartado de este tema. Lo que sí se pretende es que sea un aliciente para motivaros a iniciar los desarrollos computacionales con R y proveeros con las pautas sobre cómo enfrentar, por un lado, la formulación de un problema estadístico en términos de un algoritmo computacional y, por otro, ser capaz de escribir este algoritmo en el lenguaje R.

Ideas clave

Los objetivos que intentamos alcanzar con este tema son:

- Introducir formalmente el área tecnológica de estadística computacional.
- Entender y ser capaz de pensar en términos de la aplicación de los métodos de la estadística computacional a problemas técnicos que podemos tener hoy en día, haciendo énfasis en los problemas derivados del tratamiento de entornos *Big Data*.
- Presentar el *software* estadístico R.
- Motivar y proveer de herramientas para la programación con R.

La idea central que queremos recalcar es la necesidad que tenemos como profesionales de ser **capaces de controlar el uso de métodos computacionales para resolver problemas estadísticos** si pretendemos superar retos que en estos momentos conciernen a los científicos de datos.

Este conocimiento, la estadística computacional, puede ser entendida como el puente entre ciencias clásicas (matemáticas, estadística) y campos científicos aun en sus inicios (inteligencia artificial, bioinformática, ciberseguridad, medicina personalizada, *machine learning*). Los avances que podamos lograr en estas últimas áreas mencionadas serán dependientes en gran medida del rigor y alcance con que podamos abordar la estadística computacional.

2.2. Principios básicos

La estadística computacional se consolida como Ciencia que se sustenta en la implementación computacional de conceptos, reglas y fórmulas, derivadas de análisis matemáticos y estadísticos utilizados para describir la solución a un problema (ver Figura 1).

De arriba hacia abajo, vemos las bases tecnológicas que fundamentan los desarrollos en estadística computacional y conducen a las aplicaciones tangibles hoy día. Campos como el *Big Data* podrían dan soporte a nuevos retos tecnológicos incluso aun por definir.

Por otro parte, **análisis estadísticos** se refiere a la ciencia de recopilar, discutir, visualizar y analizar datos. Datos que pueden constituir una muestra finita o una porción de un espacio muestral infinito. Esta recopilación, discusión, visualización y análisis de datos se hace, esencial y complementariamente, basándose en métodos matemáticos.

Ideas clave

Hoy en día, nos encontramos ante la necesidad/reto de analizar problemas que derivan del comportamiento de sistemas «grandes» (con muchos datos) incluso muchas interacciones ocultas entre estos datos, que generan información (nuevos datos) y que, en esencia, es lo que subyace al problema central que motiva el desarrollo de la estadística computacional: la necesidad de **entender sistemas *Big Data*** en un entorno intrínsecamente complejo.

Por tanto, los principios fundacionales de la estadística computacional subyacen en el conocimiento y control de **tres áreas técnicas:**

- **Programación:** nos centraremos en desarrollar, programar, aplicaciones en el *software* estadístico R. Con este contenido se pretenden dar pautas para el trabajo en la solución e implementación en R (es decir, implementación a nivel de principiante) de problemas matemáticos, estadísticos y de programación.

Presentaremos ejemplos prácticos que sirvan de guía en esta iniciación e incluso, y muy importante, atendiendo a lo que comentaremos más adelante como una de las reglas básicas de programación, presentaremos **problemas y soluciones numéricas sencillas** que deben **servir de plantilla para la solución de problemas más complejos**. Esta reutilización de código, concepto propio de la programación numérica y que sin definición estricta avanzamos, debe permitir al analista de datos poder modificar y ajustar ejemplos a sus necesidades derivadas del análisis de casos reales o modelos más complejos.

- **Análisis numérico:** durante la asignatura, veremos y motivaremos al alumno a enunciar problemas que conducen a soluciones en el marco de la estadística computacional. Estos problemas, pueden derivar de situaciones ya anunciadas dentro del entorno *Big Data* y los sistemas complejos. Estas situaciones, a su vez, constituyen la última etapa en el mapeo al análisis numérico de situaciones reales en áreas incluso emergentes hoy en día, como pueden ser la biología, la medicina, ciberseguridad, etc.

Frecuentemente, estos problemas no tienen solución analítica o el resultado exacto, —debido al elevado número de datos, requiere mucho tiempo para su evaluación—. En este sentido, se hace imprescindible el uso técnicas numéricas para aproximar el resultado. Distinción aparte merece el hecho de que los *software*s estadísticos, por ejemplo, R, pueden ayudar en el cómputo de datos, en su arreglo y en su proceso de visualización. Aspectos que también trabajaremos en este curso.

En la siguiente figura se puede ver la propuesta de solución a un problema en el ámbito de análisis de datos para sistemas complejos que implica el uso de estadística computacional.

La solución, de manera global, se describe con **tres pasos fundamentales**:

- Modelado del problema.
- *Propuesta de solución numérica.*
- *Implementación numérica.*
- **Estadística clásica:** en este contexto presentaremos los métodos estadísticos básicos utilizados para describir y analizar **datos univariados,** temas propios de estadística descriptiva e inferencial de datos univariados, y se irán incorporando a los temas en la medida que se desarrolle el curso.

También retomaremos análisis propios de combinatoria y teoría de probabilidades, básicos para progresar en tratamientos estadísticos. Para análisis en sistemas de muchas variables utilizaremos **métodos de regresión lineal y multivariable**. Los modelos de regresión son extremadamente importantes. Por ejemplo, la regresión lineal, herramienta simple pero poderosa para investigar las dependencias lineales y basada en supuestos estrictos de distribución a los modelos de regresión no paramétricos imprescindibles en el análisis de entornos *Big Data*.

Con estos elementos podemos decir que vamos a trabajar con la conjunción de fuerzas que ofrecen los elementos de análisis matemático, estadística y programación, encaminados a resolver problemas de una alta dimensionalidad (es decir, número de datos y complejidad), o sea, estamos en la línea correcta para trabajar en el área de estadística computacional.

2.3. Ámbitos de aplicación

Áreas de aplicación para la estadística computacional… Parodiando mensajes propios del mundo de la programación, podríamos añadir: ***running***.

¿Qué queremos decir con esto?

Estamos ante un **campo nuevo**, con aplicaciones con tendencia a resolver y simplificar problemas numéricos; por ejemplo, hacer funciones similares a calculadoras programables. Pero la estadística computacional, como hemos enunciado, es más que esto, lleva a plantear soluciones a problemas hasta ahora irresolubles y, por tanto, a generar nuevos problemas.

¿En qué áreas estamos viendo este impacto hoy en día? Véase la Figura 1 que resume el actual contexto de trabajo y entremos en más detalle en los siguientes párrafos.

- **Estadística computacional en Biología, bioestadística computacional:** es un área que une ramas ya consolidadas como bioinformática, genómica y biotecnología. Todas estas áreas tomadas como ejemplo muestran hoy en día un nicho de actividad importante para los científicos de datos. Son áreas que lucen por los prometedores avances que usan (y generan) grandes volúmenes y diversos tipos de datos. Por tanto, exigen el desarrollo de metodologías y herramientas eficientes de estadística computacional integradas con conocimiento biológico y algoritmos computacionales, o sea bioestadística computacional.

- ***Big Data* como base en el desarrollo de la medicina**, es decir, informática o ingeniería de datos para biomedicina, medicina personalizada: estas son algunas de las etiquetas que hoy en día se utilizan para sintetizar áreas emergentes de desarrollo. Tratan fundamentalmente del potencial que subyace en la utilización de «muchos datos» para lograr avances en estudios médicos. Por ejemplo, podemos citar los estudios derivados de análisis de genoma humano para la predicción de

comportamientos futuros de los individuos. Esto implica análisis de muestras de genotipos, comparación con datos de otros individuos (cuanto más, mejor). Todo esto es una aplicación más del análisis de datos e incluso lo que se conoce como *machine learning*, aplicado a problemas médicos.

- **Estadística computacional para facilitar trabajos de Ciberseguridad**. En estos momentos el tráfico de datos en el entorno virtual es una realidad y necesidad creciente. Este comportamiento genera acciones derivadas del uso que puedan tener estos datos, por un lado, la buena intención de sacar provecho de estos para fines legítimos u otras intenciones más oscuras encaminadas a utilizar estos datos para generar problemas de ámbito técnico, *malware,* o incluso para desestabilizar determinados entornos, *fakenews*. En cualquier caso, el objetivo es claro, controlar el tráfico de datos y poder proveerlo de seguridad. Para esto es necesario ser capaces de registrar, codificar, las transferencias que ocurren en red. Estos son procesos matemáticos que incluyen análisis dinámicos, contenidos avanzados para este curso, pero que en general ahora necesitamos entender como series de números que evolucionan en el tiempo. La posibilidad de desarrollar métodos seguros en este ámbito requiere de la participación de los estadísticos computacionales además de otros perfiles técnicos.

Con el panorama antes expuesto queremos dejar patente la **necesidad de trabajar en el ámbito de estadística computacional**, basándonos en el hecho de que es un campo reciente y, por tanto, abierto a **aportaciones y oportunidades** desde el punto de vista de contribuciones de desarrollo, además de que en estos momentos ya se ha consolidado como solución a muchos problemas existentes.

2.4. Técnicas básicas de programación

A continuación, queremos sintetizar lo que consideramos serán buenas prácticas en el ámbito de la programación de soluciones a problemas estadísticos.

Conocidas como **«buenas prácticas»**, son extensibles a cualquier lenguaje de programación y en particular nos centraremos en **ponerlas en práctica en los códigos que se desarrollen con R**:

- **Prestar atención a la sintaxis de operaciones,** que se conoce como «expresividad» del lenguaje, lo que implica usar variables, denominaciones, etiquetas, cuadros de texto que ayuden a seguir el código sin necesidad de ser especialista en desarrollo informático.

Por ejemplo, si una variable guarda valores asociados a las notas de unos alumnos, lo más oportuno es denominarla «notas» o análogo. En caso de que los códigos sean extensos, o incluso como praxis general, se recomienda introducir al principio de los códigos cuadros de texto indicando qué significa cada variable utilizada.

- **Seccionar el programa**, esto se hace con la intención de facilitar el proceso de validación de código. Por ejemplo, si es un cálculo que como *input* utiliza las funciones f_1,f_2,f_3, debemos dejar indicado en el código dónde se calculan cada una de estas funciones para poder estructurar, así, el proceso de verificación de código.

- **Facilitar lo que se conoce como «modularidad»**, similar a lo anterior, pero con más implicaciones. Se trata de dividir el programa tanto como sea posible en «módulos». Con estos módulos se gana en el proceso de validación de código, pero también se potencia un aspecto relevante en el ámbito de desarrollo computacional, la posibilidad de «reutilizar» código.

Un ejemplo es el hecho de que necesitemos generar alguna función como parte de la

solución a nuestro problema. Es recomendable, en este caso, salvarla como función independiente, incluso si se trata de un diseño propio, y poder invocarla en futuras realizaciones.

Con esto sentamos unas bases para poder comenzar a programar a **nivel principiante**. Bases abiertas a su desarrollo y que, de forma general, constituyen una guía para perfilar buenas prácticas en la elaboración de *software*, particularmente *software* estadístico en el contexto que trabajamos.

2.5. Presentación del software «R»

R es un programa muy útil para el análisis, representación y visualización de datos. Es abierto (*open source*), gratuito y se puede descargar de Internet.

Accede a la página de descarga a través del aula virtual o desde la siguiente dirección:
https://www.r-project.org/

Aquí se pueden encontrar los ejecutables para los distintos sistemas operativos, pero, además, como corresponde a su definición también podemos tener acceso al código fuente (esto permite algunos estudios avanzados de implementación estadística prácticamente al alcance de cualquier usuario). Para este curso recomendamos inicialmente bajar los ejecutables para el sistema operativo que prefiera el alumno.

A modo de resumen presentamos algunas de las características de R que lo hacen ser el *software* de elección para conducir este curso y que, de hecho, justifican la tendencia al alza en su uso en el gremio de los científicos de datos.

- **Contiene implementaciones para el cálculo de «todas» las herramientas estadísticas** . Aquellas que no se encuentran, dada su especificidad o novedad, suelen ser añadidas por usuarios y agregadas como librerías de libre acceso.
- **Permite el acceso a otros programas de cálculo matemático**. Acceso entendido como compartición de librerías. Algunos programas que se pueden hibridar con R son: **C,C++, Fortran**.
- Es una potente herramienta de cálculo numérico que se basa en potenciar la **programación orientada a objetos** que, a su vez, le concede alta eficiencia para trabajar con distintos formatos de lectura/importación de datos externos.

Con estos elementos recomendamos proceder a la instalación de R y empezar a utilizarlo. Daremos algunas pautas que pueden servir de guía, pero como norma general recomendamos consultar manuales más extensos o ayudas *online* para poder gestionar los problemas de implementación que surjan durante el trabajo.

Estructuras básicas: bases de datos, operadores, funciones y librerías

El objetivo fundamental de un *software* diseñado para estadísticas es ser capaz de leer datos, manipularlos, operar con ellos y guardarlos adecuadamente. Todas estas funcionalidades las desarrolla R. En general R es muy versátil, en cuanto a que puede trabajar con distintos tipos de datos y cambiar de un tipo a otro según convenga.

Para la **lectura de datos**, veremos **cómo podemos leer datos de ficheros externos** y, por comodidad en este ejemplo, **guardarlos en una tabla**.

Primero, crearemos un fichero .txt (es el tipo más genérico de extensión que nos «conecta» con cualquier otra extensión de datos que trabajemos), que, por ejemplo, represente una lista finita de «edad» de un grupo de personas: 10,20,30,10,40,80 y encabezaremos una columna con el nombre: **edad**. Guardamos el fichero como edad.txt en nuestro directorio de trabajo, por ejemplo, PRÁCTICAS.

A continuación, abrimos la consola de R y vamos al directorio de trabajo PRACTICAS. Para esto usamos el comando:

Desde aquí ya podemos cargar el fichero edad.txt y guardarlo en una variable:

Otra manera muy recomendada de guardar datos en R es mediante la **creación de vectores**.

Por ejemplo, en el caso anterior, en lugar de importar una lista de edades pensemos en crear una lista que contga las edades de un grupo análogo al anterior. Creamos en este caso un vector «edad» para guardar esta información. En línea con el ejemplo anterior este vector se crearía así:

Con estas dos estructuras básicas de almacenamiento de datos podemos comenzar a operar en R.

En cuanto a operaciones, R puede funcionar como una **calculadora habitual**. Por ejemplo:

RichText template tag **rawhtml** is not configured

Y así, en esta línea, las operaciones algebraicas ya conocidas.

Para calcular algunas **funciones tipo exponenciales**, se pueden introducir directamente por pantalla, pues son de las que el programa carga por defecto. En línea con el desarrollo del código, se han implementado funciones más complicadas, incluso algunos estadísticos específicos para determinados contextos.

Todo esto lleva a que sea necesario, antes de escribir el código, **investigar las librerías que contienen las funciones** que necesitamos y dar al programa la orden de cargarlas antes de interpretar nuestro código.

Veamos un ejemplo:

En estadística suele ser usual el proceso de codificación de variables durante una recogida de datos. Por ejemplo, preguntar por «práctica de deporte» y tener respuesta «sí» o «no». Para trabajar, puede ser útil transformar estas variables a numéricas, ejemplo «1» y «0» respectivamente. Este proceso se designa con el término anglosajón *recoding*.

¿Por qué es útil pensar en un término anglosajón? Porque con los programas que solemos trabajar hoy en día, los diseños de código suelen corresponder a estructuras anglosajonas y porque la manera de proceder para el investigador ante la duda de cómo introducir una funcionalidad, la opción inicial suele ser la búsqueda de bibliografía. En este caso, la búsqueda o respuesta a la pregunta: ¿existe alguna función que haga lo que necesito? Si buscamos sobre «R *software* recoding», inmediatamente nos dirigirá al uso de la función ***recode*** con ejemplos como el que mostramos a continuación.

Supongamos que declaramos una lista de datos sobre práctica deportiva:

Queremos recodificar estas variables como se indica en el párrafo anterior. Entonces, debemos cargar la librería «car» y utilizar la función «recode». Esta información se obtiene investigando en la documentación del programa.

RichText template tag **rawhtml** is not configured

Esta es la **rutina de trabajo**: fijar el problema que necesitamos resolver, buscar qué hay ya implementado en el *software* que nos puede ayudar a escribir el algoritmo de solución (generalmente esto se refiera buscar funciones implementadas que estén implícitas en la solución al problema). Finalmente, como estrategia de programación para principiantes se

recomienda acceder a ejemplos y mapearlos a el caso particular que se esté tratando.

Ideas clave

Representación de datos: variables categóricas y variables numéricas

El análisis estadístico necesita manipular datos. Datos que, por su origen, pueden tener distinta forma o lo que en R denominamos «clase». Trabajaremos con dos clases de estadísticos: **categóricos o nominales** y **numéricos o cuantitativos**.

Mostraremos a modo de ejemplo cómo obtener **parámetros descriptivos** para cada uno de estos tipos de variables.

Supongamos que tenemos una variable categórica que guarda información sobre alumnos «aprobados» y «suspensos»:

Esta variable «notaUnir» guarda una lista con la información de 20 «aprobados» y 10 «suspensos». En la definición, y para no repetir el valor categórico (20 y 10 veces respectivamente), utilizamos de forma recurrente la función propia de R **rep** (del término anglosajón *reproduce*). Utilizando otra función, summary, podemos obtener una descripción de esta variable:

RichText template tag **rawhtml** is not configured

De manera análoga, con la función summary, podemos obtener los principales parámetros descriptivos para una variable numérica. Supongamos ahora que este es el caso de la nueva variable notaUnir,

RichText template tag **rawhtml** is not configured

Nótese que esta función provee de algunos estadísticos, pero no de todos, como es de esperar. El cálculo de aquellos que no se devuelven debe hacerse con funciones específicas que, como ya se comentó, hay que investigar para llegar a su forma a partir de la documentación del programa.

En casos más específicos o de interés técnico en los que puedan surgir situaciones en las que no tengamos implementadas las funciones necesarias, se desarrollará el algoritmo para «definir una nueva función», en caso de que necesitemos utilizarla de manera recurrente o dividir el problema, de forma tal que nos lleve a trabajar sobre funciones conocidas.

Ideas clave

Tabulación de variables

En apartados anteriores hemos trabajado ya con tablas simples de una columna. A continuación, discutiremos aspectos más generales relacionados con las potencialidades que ofrece R en este contexto para el almacenamiento de datos.

Por medio de R podemos manipular **diferentes formatos de ficheros de bases de datos**, por ejemplo:

Para **importar estos ficheros** se usarán comandos tipo y y comandos y **para guardarlos,** una vez que hayan sido modificados. Con las tablas se puede operar como con los vectores a nivel algebraico.

Existen **librerías especializadas** para la edición de tablas, colocación de etiquetas, títulos y otros aspectos estilísticos. A nivel de programación es necesario invocar esta librería al comenzar al escribir el *script* para poder usar las funciones que nos interesan y se comentan en este punto.

Como ejemplo para este curso recomendamos la librería **expss y el uso de la función .**

Ideas clave

Gráficas básicas

Existen diversas maneras de representar datos gráficamente. En este curso nos centraremos en los **histogramas, diagramas de sectores y barras acumuladas**. A la hora de hacer un gráfico es imprescindible, desde el primer momento, poder acceder a las funciones para su correcto etiquetado, denominar ejes, regiones gráficas o lo que se necesite en este sentido. Un descuido en estos aspectos puede llevar a la incomprensión del gráfico.

Te recomendamos escribir un *script* para cada uno de los gráficos planteados, suponiendo que seguimos trabajando sobre la variable «nota» que contiene una lista resumida con información sobre resultados académicos:

RichText template tag **rawhtml** is not configured

Al introducir estas instrucciones aparece un diagrama de barras, un diagrama de sectores y un histograma. Sin embargo, hay imprecisiones importantes en cuanto al estilo e incluso contenidos. Para esto se deben introducir parámetros asociados al etiquetado y estructura de los datos para cada caso. A continuación, en los temas siguientes se mostrarán ejemplos para trabajar con las estructuras introducidas en este tema.

Ideas clave

Perspectivas

Como hemos intentado mostrar, R se puede usar en cualquier problema estadístico y para trabajar con cualquier tipología de datos.

En este curso nos limitamos a mostrar el uso de R para la realización de los análisis estadísticos que alcanzamos a estudiar a nivel principiante. Esto incluye análisis descriptivo de datos y algunos mínimos sobre visualización. A nivel superior, se puede indicar que R es útil, además, para realizar operaciones entre bases de datos de manera análoga a la forma en que trabajan algunos *software*s funcionales que se ocupan del tema de la lógica relacional.

Con R podemos incluso desarrollar modelos de aprendizaje automático y modelado matemático avanzado, altamente demandados en temas de inteligencia artificial y *machine learning*.

Características del lenguaje R: flexible, reproducible, código abierto e interfaces controlables a través de línea de comandos.

- El hecho de ser **flexible,** como hemos venido comentando, está relacionado con la capacidad que tiene para ofrecer una solución numérica a cualquier problema estadístico que necesitemos. Su amplio número de funciones implementadas deben satisfacer las demandas a nivel de principiante. En términos de investigación computacional, se puede necesitar desarrollar nuevas funciones y esto es posible con la estructura de código que tenemos disponible.
- **Código reproducible.** Como anunciamos en el apartado de «buenas prácticas», si escribimos un código claro (legible) podemos reutilizar el código para distintas bases de datos.
- El hecho de ser un **código abierto** permite siempre identificar errores o introducir mejoras en los procesos de desarrollo ya implementados. Es importante destacar que esto es un esfuerzo comunitario, como otros códigos abiertos, y en este sentido todas las aportaciones son bienvenidas.
- El poder trabajar con **líneas de comandos** da un poder superior al usuario que no se limita a «activar funcionalidades», sino que puede mejorarlas y entrar a perfilar su codificación en función de sus especificidades.

Como **limitaciones al trabajo con R** podemos señalar que el método de trabajo no suele ser intuitivo y se presenta como un acto de investigación en cuanto al desarrollo.

El usuario encuentra un espacio vacío al comenzar un *script* y a partir de ahí, como hemos indicado, el trabajo se centra en identificar ¿qué necesito?, buscar referencias previas de estudios en R y, finalmente, con esta información, explorar en la documentación propia del programa detalles más finos, como pueden ser las sintaxis de las funciones a utilizar.

A nivel de ***hardware***, R posee algunas **limitaciones de memoria**. De momento encontramos dos limitaciones fundamentales que pueden afectar el trabajo en entornos *Big Data*:

- **Acceso a la memoria operativa:** debido a que R trabaja en la memoria operativa, si por algo perdemos el cálculo, aquello que no hayamos guardado es susceptible de perderse. Por tanto, en cálculos extensos es recomendable ir guardando en el disco periódicamente.

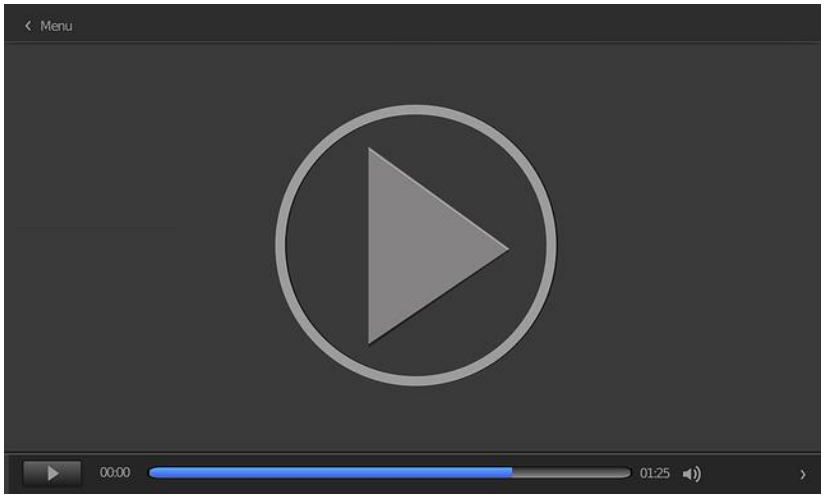
- **Trabajo con extensas bases de datos:** como comentamos anteriormente, R tiene limitaciones de trabajo con algunas funciones que colapsan si las bases de datos superan determinadas dimensiones. Para resolver esto se usan paquetes auxiliares que en esencia se ocupan de dividir estas bases de datos tanto como se necesario y utilizar conectores para operar de forma segura con ellas.

En general, este es el panorama de R en el entorno de estadística computacional.

Consideramos que la principal motivación para adentrarse en este contexto de desarrollo es la posibilidad de no ser un usuario de un código oscuro, sino ser capaz de usar y crear, incluso, aportaciones a un campo emergente como es el de la estadística computacional que hemos presentado en este tema.

Estadística computacional: inicio a la programación con R

En este vídeo vamos a introducir el concepto de estadística computacional y la programación con R como medio para llevarla a la práctica.



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=50489bf2-4dcf-41b3-a63d-acbd00aaf3eb

Primeros pasos con R

Asumiendo que tienes instalado R y RStudio, abre el IDE de RStudio y en un «R script» nuevo «Ctrl + Shift + N» escribe el siguiente código. En este ejercicio, es importante que leas los comentarios que hay en el propio código para comprender lo que hace cada instrucción.

Una vez copiado, prueba a ejecutar el *script* línea a línea. Para ello, posiciona el cursor en la primera línea y utiliza la opción «Run» o «Ctrl + Enter».

```
##################################################
############### Aprende a usar R desde RStudio #####
##################################################

# A) Para poner comentarios utiliza el símbolo '#' sin las comillas simples

##################################################
# B) Utiliza CTRL + SHIFT + C para comentar o descomentar un bloque de comentarios
# Esta línea es un comentario (Posiciona el cursor aquí y Utiliza CTRL + SHIFT + C)

##################################################
# C) Con el comando CTRL+ENTER se ejecutan las líneas de código seleccionadas
print('Esto es un primer mensaje por pantalla')
print('Esto es otro mensaje por pantalla')

##################################################
# D) Si no se selecciona nada, con el comando anterior se ejecuta 'línea por línea'
print('Esto es un segundo mensaje por pantalla')
print('Esto es otro mensaje por pantalla')
```

```
##################################################
# E) Con el comando CTRL+SHIFT+ENTER se ejecuta el script completo ::::: NO LO HAGAS :)

#############################################################################################
################ Utiliza estructuras de datos en R ##########################################
#############################################################################################

# Es recomendable establecer un directorio de trabajo para R, en dicho directorio estarán todos los ficheros
de tu proyecto.
#Indica aquí tu propia ruta
MAIN_DIR = 'C:/R-Proy/'
#Establece el directorio
setwd(MAIN_DIR)
#comprueba el directorio de trabajo actual
getwd()

################ Utiliza variables ##################
# asigna el valor a una variable
y = 2
# Comprueba su tipo
class(y)

z = 2.3
class(z)

a = as.integer(2)
class(a)

a = as.integer(2.8)
class(a)
print(a)

b = 'str'
class(b)

#variables lógicas True/False
b = T
c = F
class(b)
class(c)

################ Utiliza vectores ##################
vector1 = c(1,2,3,4,5,6,7,8,9,0)
class(vector1)
length(vector1)

# ¿Cuál es el tamaño de la variable a?
length(a)

vector1 = c(3:15)
print(vector1)

vector2 = c(2.1, 2.8, 3, 4.1, 55.2)
class(vector2)
# ¿Qué ocurre a continuación?
print(vector1 + vector2)

# ¿y ahora?
vector1 = c(1:5)
# ¿Funciona? ¿Qué ocurrió?
print(vector1 + vector2)
```

```
# Cadenas...
vector3 = c('str1', 'str2', 'str3')
class(vector3)

vector4 = c(8, 7.9, 'str2', 6, T, F)
class(vector4)
# Ten presente: todas las componentes del vector deben ser del mismo tipo -> cohercion.
# ¿Qué ocurre a continuación?
print(vector4)

vector5 = c(F, F, T, T, T)
class(vector5)

################# Utiliza matrices ##################
# Debes crearlas a partir de vectores
vector1 = c(1:6)
print(vector1)
length(vector1)

m1 = matrix(vector1, ncol = 3, nrow = 2, byrow = F)
class(m1)
print(m1)

# Tamaño de matrix
length(m1)
dim(m1)
nrow(m1)
ncol(m1)

# Crea matrices a partir de vectores de distintas formas
m1 = matrix(vector1, ncol = 2, nrow = 2, byrow = F)
print(m1)

m1 = matrix(vector1, ncol = 2, nrow = 2, byrow = T)
print(m1)

m1 = matrix(vector1)
print(m1)
dim(m1)

m1 = matrix(vector1, ncol = 5, nrow = 6, byrow = T)
print(m1)

# Seguro habrá concluido que como las matrices se construyen a partir de vectores, estas solo aceptan un tipo
de dato en sus componentes.
m2 = matrix(c('str',9, T, F, 8.1))
print(m2) #coercion

################# Utiliza factores ##################
# Al igual que las matrices, se deben crear a partir de vectores.
# Los factores son útiles para manipular variables categóricas
vector1 = c(1:6)
f1 = factor(vector1)
class(f1)
print(f1)
levels(f1)
class(levels(f1))

f2 = factor(c(1,2,3,5,7))
print(f2)
levels(f2)
```

```
levels(f2) = c('a1', 'a2', 'a3', 'a5', 'a6')
print(f2)

################# Utiliza dataframes ##################
# Es una de las principales estructuras de datos de R
# Puedes crear dataframe a partir de vectores, matrices y factores.
# Del dataframe, cada columna es un vector y por ello, los dataframes deben tener el mismo tipo de dato por
COLUMNA.

df1 = data.frame(m1)
class(df1)
print(df1)

# La longitud no es el número de elementos, sino el número de columnas.
length(df1)
length(m1)

dim(df1)
nrow(df1)
ncol(df1)

# Utiliza nombres de filas y columnas (vectores)
colnames(df1) # == names(df1)
rownames(df1)

# Establece nombres de columnas o filas a tu dataframe
colnames(df1) = c('A1', 'A2', 'A3', 'A4', 'A5')
rownames(df1) = c('R1','R2','R3','R4','R5', 'R6')
print(df1)

# Más utilidades para los dataframes
mtcars
head(mtcars, 5)
tail(mtcars)

# Consulta para qué se utiliza la función str()
str(mtcars)

################# Utiliza Listas ##################
# En R, las listas son las estructuras de dato con mayor jerarquía.
# Una lista puede contener variables, vectores, matrices, factores, dataframes o incluso listas anidadas.
# Se podría afirmar que un dataframe es como una lista de vectores.

l1 = list(vector1, m1, f1, df1)
print(l1)
class(l1)
str(l1)
str(l1, max.level = 1)

# Establece nombres en tus listas
l2 = list('Object1' = vector1, 'Object2' = m1, 'Object3' = f1, 'Object4' = df1)
names(l2)
str(l2, max.level = 1)

l2 = list(vector1,m1,f1,df1)
str(l2, max.level = 1)
names(l2) = c('element1', 'element2', 'element3', 'element4')
str(l2, max.level = 1)

#####################################################
############### Operaciones Lógicas ################
```

```
##################################################

# AND: &
T & F
# OR: |
T | F

################ Utiliza variables en operaciones lógicas ##################
a = 0
a > 3
a == 2
a != 4
!(a == 2)

################ Utiliza vectores en operaciones lógicas ##################
vector1 = c(1:4)
vector1 > 2
!(vector1 > 3)
vector1 %in% c(2,3)

################ Utiliza matrices en operaciones lógicas ##################
m1
m1 > 1
!(m1 > 2)
m1 %in% c(5,6)

###########################################################################################
################## Selección de valores y porción de datos (Slice) ########################
###########################################################################################

################ Selección de valores en vectores ##################
vector1 = c(1:5)
vector1[5]

# Visualiza todos los elementos
vector1[-2]

# En R el índice comienza en 1 (en python que empieza en 0)
vector1[0]
vector1[c(1,6)]

# Es equivalente a v1[1:3]
vector1[c(1:3)]

# Visualiza todos los elementos
vector1[-c(1:3)]

# IMPORTANTE, APRENDER A USAR
# Comparación de valores en vectores
# Se pueden utilizar máscaras de booleanos
vector1[vector1>3]
vector1[!(vector1>3)]
vector1[vector1 %in% c(1,5)]
vector1[!(vector1 %in% c(2,3))]

# Utiliza nombre de componentes
names(vector1)
names(vector1) = paste('n', seq(1,length(vector1)), sep = '_')

# ¿Qué hace la instrucción anterior?
print(vector1)
```

```
vector1[c('n_1', 'n_2')]

# Utiliza una condición "x" para recuperar "y" valores de un vector.
# Tener en cuenta que "OJO "x" e "y" deben tener el mismo tamaño.
names(vector1)[vector1>2]
vector1[names(vector1) == 'n_4']

################## Selección de valores en matrices ##################
print(m1)

# Los elementos tienen posición i,j en la matriz (i = fila; j = columna)
m1[2,1]

# Recupera un vector de la matriz
m1[2:5, 4]

# Recupera una submatriz
m1[2:5, 1:4]

# Utiliza idx de 1 a length(m1)
m1[30]

# Lleva a cabo comparación de valores, pero ten presente que esto retorna un vector, no una matriz.
m1[m1>3]
m1[m1 %in% c(2,5)]

################## Selección de valores en dataframe ##################
# La operativa es similar que en las matrices, pero además permite usar nombres de columnas
mtcars[1:5, c('cyl', 'wt')]

# Fila completa
mtcars[5,]

# Utiliza posición de columna
mtcars[4]
class(mtcars[4])
mtcars[,4]
class(mtcars[,4])

# Utiliza nombres de columna
mtcars['wt']
class(mtcars['wt'])

# Utiliza $ para referenciar las columnas
mtcars$wt
class(mtcars$wt)

# Utiliza condición en alguna(s) variable
mtcars[(mtcars$gear == 3),]
mtcars[(mtcars$carb == 3 & mtcars$mpg > 10),]
#obs: más adelante veremos como filtrar dataframes de mejor manera

################## Selección de valores en Listas ##################
# Ten en cuenta que [[ ]] y $ son equivalentes

l2[1]
class(l2[1])

l2[[1]]
class(l2[[1]])

print(l2)
```

```
# ¿Recuerdas de donde salió el nombre "element4"?
l2$element4$A2
l2$element4$A3

l2[1:3]

# Utiliza la selección anidada
l2[[4]]
l2[[4]]['A1']
l2[[4]][['A1']]
l2$element4$A4
```

Avanza un poco más con R

Asumiendo que tienes instalado R y RStudio, abre el IDE de RStudio y en un «R script» nuevo «Ctrl + Shift + N» escribe el siguiente código. En este ejercicio, también es importante que leas los comentarios que hay en el propio código para comprender lo que hace cada instrucción.

Una vez copiado, prueba a ejecutar el *script* línea a línea. Para ello, posiciona el cursor en la primera línea y utiliza la opción «Run» o «Ctrl + Enter».

```
##################################################
############## Aprende a usar R desde RStudio #####
##################################################

######################################################
############## Sentencias condicionales ##############
######################################################

############## Sentencia IF ##############
condition = T
if (condition) {
print('...Se cumple la condición...')
}

############## Sentencia IF ELSE ##############
condition = (6 == 2)

if (condition) {
print('Se cumple la condición')
} else {
print('NO se cumple la condición ')
}

############## Sentencia IF ELSE IF ELSE ##############
q_flow = 35
if (q_flow > 90) {
print('nº mayor a 90')
} else if (q_flow < 90 & q_flow > 10) {
print('nº menor a 90 y mayor a 10')
} else {
print('nº menor a 10')
}

############## Condiciones y Vectores ##############
```

```
vector1 = c(1:6)
condition = vector1 > 3
condition
if (condition) {
print('La condición se cumple')
}

# Alternativas para utilizar los vectores como parte de una condición
# or: a todos los componentes
any(condition)
# and: a todos los componentes
all(condition)

#####################################################
###################### Bucles #######################
#####################################################

############## While ##############
contador = 0
while (contador < 6) {
print(contador)
contador = contador+1
}

############## While + break ##############
contador = 0
while (T) {
print(contador)
contador = contador+1
if (contador == 2){
break
}
}

############## While + next ##############
contador = 0
while (T) {
if (contador == 2) {
# Finaliza el bucle
contador = contador+1
next
}
print(contador)
contador = contador+1
if (contador == 5){
break
}
}

############## FOR ##############
# Se utiliza principalmente para iterar vectores
for (var in 5:10) {
print(var)
}

var_values = c('cadena1', 'cadena2', 'cadena3')
for (var in var_values) {
print(paste('El valor de var es:', var))
}

# next y break se pueden utilizar en los bucles FOR, aunque es menos frecuente.
for (var in 1:10) {
```

```
if (var == 2){
next
}
print(var)
if (var == 5){
break
}
}

############## Instrucción REPEAT ##############
# Es similar a while(True), se recomienda utilizar junto a 'break'
contador = 0
repeat {
if (contador == 2) {
#Finaliza el bucle
contador = contador+1
next
}
print(contador)
contador = contador+1
if (contador == 5){
break
}
}

# Es posible usar bucles anidados, incluso combinando los 3 tipos anteriores: while-for-repeat.

#####################################################
################### Uso de Packages ###################
#####################################################

#importar paquetes: library() - require()
library(dplyr)
package_is_in = require(dplyr)
print(package_is_in)

library(MeInventoUnPaquete)
package_is_in = require(MeInventoUnPaquete)
print(package_is_in)

#instalar paquetes
package_name = 'dplyr'
install.packages(package_name)

#lista de paquetes instalados
installed.packages()
class(installed.packages())
rownames(installed.packages())
package_name %in% rownames(installed.packages())

#####################################################
################## Utilities ####################
#####################################################

str(mtcars)
class(read.csv)
args(read.csv)
help(read.csv)
#?read.csv
?dplyr

# paste0 == paste(sep = '')
```

```
paste('Primera parte del str', 'segunda parte del str', sep = ', ')

u1 = 2
u2 = 4

# No se recomienda lo siguiente:
paste('El primer número es:', u1, 'y el segundo es:', u2)

# Esto es lo recomendado:
sprintf('El primer número es: %s y el segundo es: %s', u1, u2)
```

Practica con R los conceptos estudiados

Asumiendo que tienes instalado R y RStudio, abre el IDE de RStudio y en un «R script» nuevo «Ctrl + Shift + N» escribe el siguiente código.

```
rm(list=ls())
##################Tema 1#############################################

requiredPackages <- c("arsenal", "car", "corrplot", "gapminder","dplyr","DescTools", "foreign", "e1071",
"expss", "GGally", "ggplot2", "haven", "knitr","plotly", "remotes", "summarytools","ggridges","table1",
"tableone", "tidyverse", "SmartEDA")

sesion1 <- function(pkg){
new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
if (length(new.pkg))
install.packages(new.pkg, dependencies = TRUE)
sapply(pkg, require, character.only = TRUE)
}

sesion1(requiredPackages)
#####################################################
##LOAD DATA
#Data Lending Club -https://www.kaggle.com/wordsforthewise/lending-club. Factores que determinan el Default en
los créditos. Modelo de riesgo
Datalc<-read.csv("https://raw.githubusercontent.com/millerjanny/Custom_UNIR/main/Data_LendingClub.csv")
# Exploración inicial
View(Datalc)
head(Datalc)
glimpse(Datalc)
names(Datalc)
str(Datalc)
dim(Datalc)
sapply(Datalc, function(x) sum(is.na(x)))

#####################################################
#Tabla de frecuencias variable categórica
table(Datalc$home_ownership_n)
freq(Datalc$home_ownership_n, style = "rmarkdown")

#Tabla de frecuencias variable continua
Datalc$int_rate_cat <- factor(cut(Datalc$int_rate,
breaks=nclass.Sturges(Datalc$int_rate),include.lowest=TRUE))
freq(Datalc$int_rate_cat, style = "rmarkdown")
#####################################################

#Pie chart
#Somos malos para juzgar el tamaño de los ángulos, que es lo que requieren los gráficos de tarta.
```

```
#Cambiar los colores de las porciones de la tarta hace que diferentes porciones parezcan más grandes o
pequeñas.
#Girar la tarta hace que las diferentes porciones parezcan más grandes o más pequeñas.
#Cuando hay pocas categorías, desperdician espacio.
#Cuando hay muchas categorías, son ilegibles.

df <- as.data.frame(table(Datalc$home_ownership_n)/length(Datalc$home_ownership_n))
colnames(df) <- c("class", "relative_freq")
pie <- ggplot(df, aes(x = "", y=relative_freq, fill = factor(class))) +
geom_bar(width = 1, stat = "identity") +
theme(axis.line = element_blank(),
plot.title = element_text(hjust=0.5)) +
labs(fill="class",
x=NULL,
y=NULL,
title="Pie Chart of Home ownership",
caption="Source: Lending Club")
pie + coord_polar(theta = "y", start=0)

#Barras
ggplot(data=Datalc, aes(home_ownership_n)) +
geom_bar(aes(y = (..count..)/sum(..count..))) +
ylab("Percent")+
geom_text(aes( label = scales::percent((..count..)/sum(..count..), accuracy = 0.01), y=
(..count..)/sum(..count..),accuracy = 0.01), stat= "count", vjust = -.5)+
scale_y_continuous(labels = scales::percent)

#comparación de Barras por variable cualitativa
ggplot(data=Datalc, aes(x=home_ownership_n)) +
geom_bar(aes(y = (..count..)/sum(..count..)*100),fill="steelblue") +
facet_wrap(~Default)+
ylab("Percent")

#Barras por variable cualitativa en el mismo gráfico
ggplot(Datalc, aes(x=home_ownership_n)) +
geom_bar(aes(y =(..count..)/sum(..count..)*100,accuracy = 0.01, fill = Default),
position = "dodge")+
ylab("Percent")

#Barras apiladas-variable cualitativa
ggplot(Datalc, aes(x = home_ownership_n, fill = Default)) +
geom_bar(aes(y =(..count..)/sum(..count..)*100,accuracy = 0.01))+
ylab("Percent")

#Histograma-Poligono de frecuencias-continua
ggplot(Datalc, aes(int_rate)) +
geom_histogram(mapping=aes(x=int_rate, y=..count../sum(..count..)*100),color="white",fill="darkblue",
bins=nclass.Sturges(Datalc$int_rate))+
geom_freqpoly(mapping=aes(x=int_rate, y=..count../sum(..count..)*100),
bins=nclass.Sturges(Datalc$int_rate))+
ggtitle("Distribution of Interest rate") +
xlab("Int Rate") +
ylab("Percent")

#Grafico de frecuencias acumuladas
ggplot(Datalc, aes(int_rate))+
geom_histogram(aes(y=cumsum(..count../sum(..count..)*100)),bins=nclass.Sturges(Datalc$int_rate),color='white',fill="darkblue")+

stat_bin(aes(y=cumsum(..count../sum(..count..)*100),bins=nclass.Sturges(Datalc$int_rate)),geom="line",color="green")+

xlab("Int Rate") +
ylab("Percent")
```

```
#comparar Grafico de frecuencias acumuladas por variable cualitativa
ggplot(Datalc, aes(int_rate, color = Default)) +
stat_ecdf(geom = "point")+
ylab("Relative frequency")

#Comparar distrbuciones de variable cuanti por variable cualitativa-crestas
ggplot(Datalc, aes(x = int_rate, y = Default)) +
geom_density_ridges(aes(fill = Default)) +
scale_fill_manual(values = c("#00AFBB", "#E7B800"))

ggplot(Datalc, aes(x = 'int_rate', y = 'home_ownership_n')) +
geom_density_ridges_gradient(aes(fill = ..x..), scale = 3, size = 0.3) +
scale_fill_gradientn(colours = c("#0D0887FF", "#CC4678FF", "#F0F921FF"),
name = "Interest rate")+
labs(title = 'Distribution of Interest rate by Home ownership')

######################################################################
#El paquete gapminder contiene un fichero de datos de población, esperanza de vida y renta per cápita de los
países del mundo entre 1952 y 2007.
#La fundación Gapminder es una organización sin fines de lucro con sede en Suecia que promueve el desarrollo
global mediante el uso de estadísticas.
library(gapminder)
# Descripción de variables
# country: factor with 142 levels
# continent: factor with 5 levels
# year: 1952-2007
# lifeExp: life expectancy at birth
# pop: total population
# gdpPercap: per-capita GDP

#Gráfico de dispersión
gap=data.frame(gapminder)
ggplot(gap, aes(y=lifeExp, x=log(gdpPercap))) +
geom_point()+
geom_smooth(method=lm)

#Gráfico de linea-variable temporal
spain <- gapminder %>%
filter(country == "Spain")

ggplot(spain, aes(x = year, y = lifeExp)) +
geom_line(color = "#0099f9", size = 1) +
geom_point(color = "#0099f9", size = 2) +
labs(title = "Average life expectancy in Spain",
subtitle = "Data from 1952 to 2007",
caption = "Source: Gapminder dataset") +
theme(plot.title = element_text(color = "#0099f9", size = 20, face = "bold", hjust = 0.5),
plot.subtitle = element_text(size = 13, face = "bold", hjust = 0.5),
plot.caption = element_text(face = "italic", hjust = 0))
```

Prueba a ejecutar el script anterior siguiendo estas indicaciones:

- Ejecuta cada línea de código, posiciona el cursor en la primera línea y utiliza la opción «Run» o «Ctrl + Enter».
- Observa la «Consola», la pestaña Environment y la pestaña Plot cuando ejecutes cada línea de código, especialmente cuando dibujes un gráfico.
- Encuentra la definición de cada función de R y comparte aquellas funciones que no conozcas en el foro de la asignatura.

Ideas clave

- Repasa con R todos los conceptos vistos hasta ahora.

Sobre la relación entre Estadística, Informática y Big Data

Ferrero, R. y López, J. L. (s.f.). La estadística en la era del *Big Data*. *Data science* [Blog].

Recomendamos, especialmente, la lectura del apartado «El Universo del Big Data en Expansión», donde se muestra con datos cuantitativos la necesidad tangible hoy en día de desarrollar métodos computacionales para abordar problemas estadísticos.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web: https://www.maximaformacion.es/blog-dat/la-estadistica-en-la-era-del-big-data/

Ejemplo de aplicación del análisis estadístico con métodos computacionales y usando software R a un problema real

Recomendamos la revisión global a este trabajo: *Basic principles in Biostatistics: likelihood and statistical thinking*. Ejemplo práctico que muestra la elección de un problema real, desarrollo de un modelo estadístico, elección de un *software* de cómputo y cálculos numéricos para dar solución al problema.

Este trabajo es una guía, incluso una plantilla, de lo que debe ser el trabajo del analista de datos y puede servir incluso para mostrar posibles líneas de trabajo en las actividades que se propondrán durante el curso.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web: https://borishejblum.science/html/m2phds-basics/biostatistics_basics_mlepracticals#1_motivational_example

---

Bibliografía

Hey, T., Tansley, S. y Tolle, K. (2009). The Fourth Paradigm: Data-intensive Scientific Discovery. *Microsoft Research* [Web].
Disponible en: https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/

R Development Core Team (ed.). R manuals. *Cran* [Web].
Disponible en: https://cran.r-project.org/manuals.html

Test

1. R soporta datos de tipo numérico en sus bases de datos:

A. Verdadero.

B. Falso.

C. Solo si se introducen como tipo .txt.

D. Ninguna de las propuestas es correcta.

2. R soporta datos de tipo categórico en sus bases de datos:

A. Verdadero.

B. Falso.

C. Solo si van acompañados de algún valor numérico.

D. Ninguna de las propuestas es correcta.

3. Histogram() es la etiqueta para desarrollar una función que elabore histogramas en un algoritmo desarrollado con R:

A. Incorrecto.

B. Correcto.

C. Falta colocar las etiquetas para completar el histograma.

D. Ninguna de las propuestas es correcta.

4. En estos momentos, R es un *software* que ofrece soporte ilimitado a la solución de problemas estadísticos en el entorno *Big Data*.

A. Sí, pero con limitaciones.

B. Nos impone la necesidad de trabajar para evitar problemas de asignación de memoria.

C. Posibilita el uso de funciones de código abierto para optimizar los recursos de memoria.

D. Todas las respuestas anteriores son correctas.

5. ¿Por qué puede ser relevante la irrupción del código R en temas de ciberseguridad?

A. Facilita el tratamiento de muchos datos.

B. Prima la lógica de los programas y la capacidad creativa del desarrollador a los mecanismos de control internos.

C. Posibilidad de paralelización de procesos al tener las estructuras modularizadas.

D. Todas las propuestas anteriores son correctas.

6. Uno de los objetivos básicos de la programación es la capacidad de desarrollar código que sea reutilizable:

A. Verdadero, pero no aplicable al contexto estadístico donde cada código debe limitarse a un problema específico.

B. Verdadero, extensible al área de la estadística donde se pretenden crear códigos generalistas que puedan ser utilizados sobre distintos escenarios.

C. Falso, siempre se debe empezar el código de cero al implementar un problema.

D. Ninguna de las anteriores.

7. Sobre el uso de la programación por módulos en R:

A. Facilita la reutilización de código.

B. Permite detectar errores (*bugs*) en un proceso de validación de código.

C. Hace el código más expresivo.

D. Todas las anteriores son correctas.

8. R no permite compartir librerías con otros lenguajes:

A. Verdadero, asociado a la seguridad propia del lenguaje.

B. Verdadero, en la línea de garantizar un uso matemáticamente correcto de los datos.

C. Falso, las librerías se pueden compartir con otros lenguajes de programación.

D. Ninguna de las anteriores.

9. ¿Puede R trabajar con varios tipos de ficheros de datos?

A. Sí, siempre que sean almacenables como .txt.

B. Sí, puede trabajar con varios tipos de ficheros, ejemplo .txt, ,csv.

10. ¿Puede un solo código R tratar simultáneamente variables categóricas y numéricas?

A. Sí, es algo estándar.

B. No, una u otra, nunca simultáneamente. Puede dar errores en el proceso de compilación.

C. No, deben transformase a uno u otro tipo y elegir un tipo para cada código.

D. Ninguna de las anteriores es correcta.