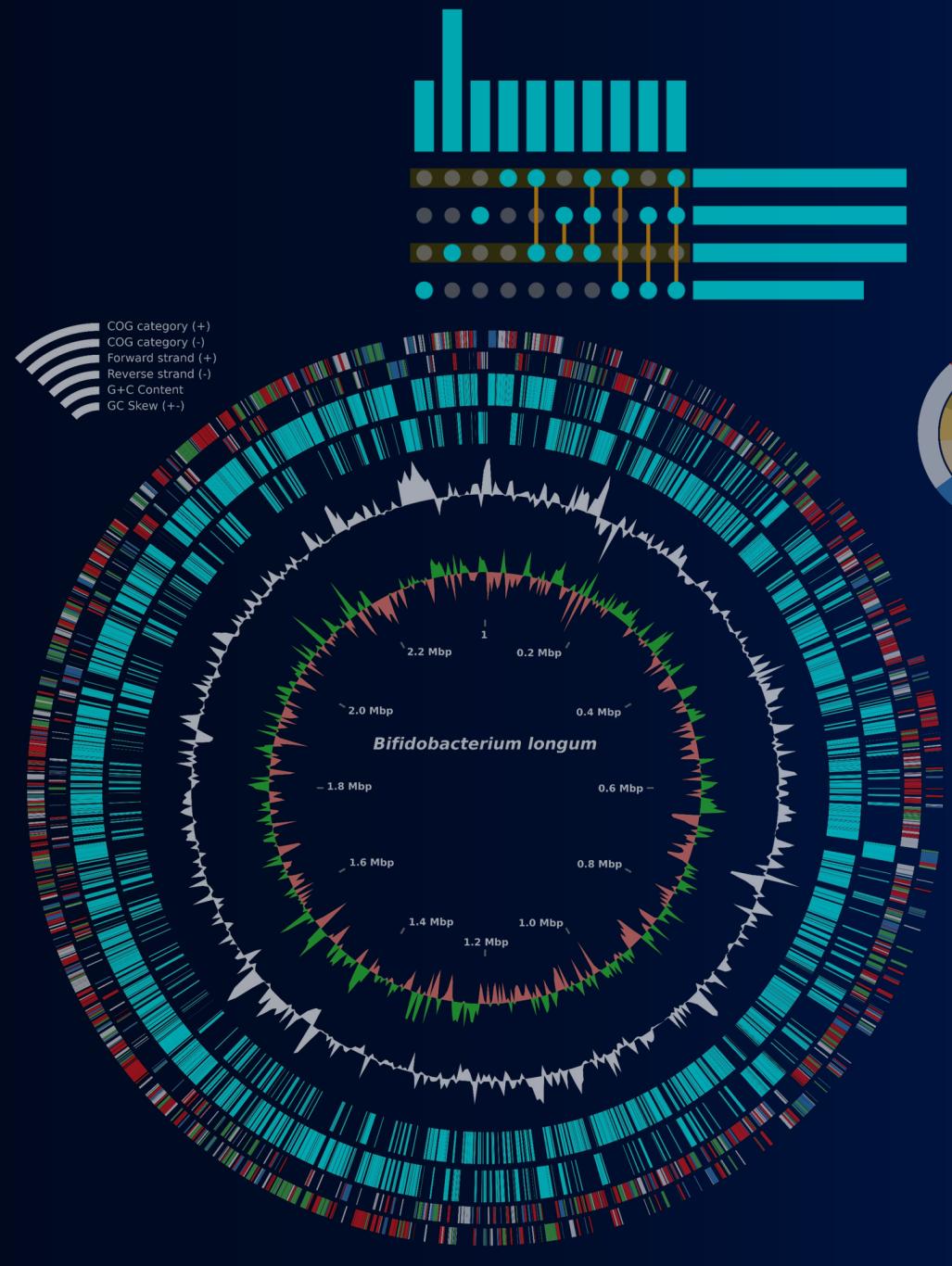


Visualización de Datos con Python para Publicaciones Científicas



GAACACATAGCCGATACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCCGTTTGAACGCGATTTGAATTCTAGCAGCACCGCAAGGAACATGCCAAGAAATTCTGTGC

OBJETIVO GENERAL

Aprender a crear visualizaciones básicas y avanzadas con el lenguaje de programación Python para publicaciones científicas.

OBJETIVOS ESPECÍFICOS

- Implementar técnicas de visualización de datos utilizando bibliotecas de Python, como Matplotlib, Seaborn, y Plotly.
- Implementar el uso de Widgets para controlar visualizaciones.
- Implementar código de programación para crear gráficos de novo.

DURACIÓN Y LUGAR DEL CURSO

Del 6 - 8 de noviembre.

15 horas, 3 sesiones de 5 horas.

Modalidad presencial de las 14:00 - 19:00 horas.

Aula Multimedia de la Coordinación de Educación Continua.

Descanso: de 16:00 – 16:30 horas.

PORCENTAJE MÍNIMO DE ASISTENCIA PARA OBTENER CONSTANCIA

80% de asistencia.

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
with open(carpeta+'GO_BP.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(proceso):
```

```
        if len(proceso) == e+1:
```

```
            fq.write(i+'\t'+proceso[i])
```

```
        else:
```

```
            fq.write(i+'\t'+proceso[i]+'\n')
```

```
with open(carpeta+'GO_MF.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(funcion):
```

```
        if len(funcion) == e+1:
```

```
            fq.write(i+'\t'+funcion[i])
```

```
        else:
```

```
            fq.write(i+'\t'+funcion[i]+'\n')
```

```
with open(carpeta+'GO_CC.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(componente):
```

```
        if len(componente) == e+1:
```

```
            fq.write(i+'\t'+componente[i])
```

```
        else:
```

Programa del curso

<https://eduardo1011.my.canva.site/curso3-2024>

The image shows a course landing page with a dark background featuring a blue and purple abstract wave pattern. At the top left is a photo of a 'SOY uam' sign. The title 'VISUALIZACIÓN DE DATOS CON PYTHON PARA PUBLICACIONES CIENTÍFICAS' is centered in large white font. At the bottom left are the Python logo and the Jupyter logo. On the right side, there are four white rounded rectangular buttons with black outlines and text: 'TEMARIO', 'CÓDIGOS', 'DESCARGAS', and 'NOTEBOOKS'. A large amount of code text is visible in the upper right corner.

TEMARIO

CÓDIGOS

DESCARGAS

NOTEBOOKS

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i][1] == 'proceso'}
```

```
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i][1] == 'funcion'}
```

```
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i][1] == 'componente'}
```

```
with open(carpeta + '/GO_BP.txt', 'w') as fq:
```

Sesión 1

Python.
Jupyter Notebook.
Variables.
Operadores booleanos.
Listas y tuplas (list, tuple).
Diccionarios (dict).
DataFrame (Pandas).
Condicionales if, elif y else.
Bucle for.

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
with open(carpeta+ '/GO_BP.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(proceso):
        if len(proceso) == e+1:
            fq.write(i+'\t'+proceso[i])
        else:
            fq.write(i+'\t'+proceso[i]+'\n')
with open(carpeta+ '/GO_MF.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(funcion):
        if len(funcion) == e+1:
            fq.write(i+'\t'+funcion[i])
        else:
            fq.write(i+'\t'+funcion[i]+'\n')
with open(carpeta+ '/GO_CC.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(componente):
        if len(componente) == e+1:
            fq.write(i+'\t'+componente[i])
        else:
```

Sesión 2

Integración de estructuras.

Visualizaciones básicas con Matplotlib.

Visualizaciones básicas con Seaborn.

Gráficos personalizados para publicaciones.

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
with open(carpeta + '/GO_BP.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(proceso):
```

```
        if len(proceso) == e+1:
```

```
            fq.write(i + '\t' + proceso[i])
```

```
        else:
```

```
            fq.write(i + '\t' + proceso[i] + '\n')
```

```
with open(carpeta + '/GO_MF.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(funcion):
```

```
        if len(funcion) == e+1:
```

```
            fq.write(i + '\t' + funcion[i])
```

```
        else:
```

```
            fq.write(i + '\t' + funcion[i] + '\n')
```

```
with open(carpeta + '/GO_CC.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(componente):
```

```
        if len(componente) == e+1:
```

```
            fq.write(i + '\t' + componente[i])
```

```
        else:
```

Sesión 3

Mapas Genómicos de bacterias.
Datos Metagenómicos.
Introducción a la visualización con Plotly.

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i]}
```

```
with open(carpeta + '/GO_BP.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(proceso):
```

```
        if len(proceso) == e+1:
```

```
            fq.write(i + '\t' + proceso[i])
```

```
        else:
```

```
            fq.write(i + '\t' + proceso[i] + '\n')
```

```
with open(carpeta + '/GO_MF.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(funcion):
```

```
        if len(funcion) == e+1:
```

```
            fq.write(i + '\t' + funcion[i])
```

```
        else:
```

```
            fq.write(i + '\t' + funcion[i] + '\n')
```

```
with open(carpeta + '/GO_CC.txt', 'w') as fq:
```

```
    fq.write('GO\tTerm\n')
```

```
    for e, i in enumerate(componente):
```

```
        if len(componente) == e+1:
```

```
            fq.write(i + '\t' + componente[i])
```

```
        else:
```

Introducción

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}

with open(carpeta+ '/GO_BP.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(proceso):
        if len(proceso) == e+1:
            fq.write(i+'\t'+proceso[i])
        else:
            fq.write(i+'\t'+proceso[i]+'\n')

with open(carpeta+ '/GO_MF.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(funcion):
        if len(funcion) == e+1:
            fq.write(i+'\t'+funcion[i])
        else:
            fq.write(i+'\t'+funcion[i]+'\n')

with open(carpeta+ '/GO_CC.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(componente):
        if len(componente) == e+1:
            fq.write(i+'\t'+componente[i])
        else:
```

CASCADA DE LAS ÓMICAS



Otras Tecnologías Ómicas

METAPROTEÓMICA
METABOLÓMICA
METAGENÓMICA
METATRANSCRIPTÓMICA
FOSFOPROTEÓMICA
FENÓMICA
GLICÓMICA

SECRETÓMICA
EPIGENÓMICA
LIPIDÓMICA
SUBPROTEÓMICA
FLUXÓMICA

NUTRIGENÓMICA
UBIQUITINÓMICA
ACETILÓMICA
INTERACTÓMICA
FARMACOGENÓMICA
IONÓMICA

ADN, ARN, PROTEÍNAS, etc.



Qué

Cuánto

Cómo

Para



Input

Cadenas de caracteres, archivos.

Formatos de inputs

(fasta, fastq, txt, ab1, json, entre otros).

.fastq

```
@P5_S371.1
GTGTCAGCGCCGGTAATACGTAGTTGGCAGCGTTGTCCGGATTACTGGG
+
<A@A@CECCF@:07@+,C,;,E,,<,6+++++8@+BCE,+,>C,<C,,,9+@
@P5_S371.2
GTGCCAGAGCGCGGTAAATACGTAGGTGGCAAGCGTTTCCGAATTATTGGG
+
CCCCCD<@FCF@C@F7@C,CCF9,C,8C,9,9;BC+8CC7+,+:C,EE,,,6+
@P5_S371.3
GTGCCAGAGCGCGGTAAATACGTAGGGGGCAAGCGTTATCCGGATTACTGGG
+
<CBCCD@<FEGD7FEF7,C,C<F9@;++6:,,C:BF,CCC:::@@FE,69<,,E
```

.fasta

```
>Sequence1
>Sequence1
NTGGTATAACGCCAGATCAACACAGGACGGCGCGGCC
ATGGTATAACGCCAGATCAACACAGGACGGCGCGGCC
CCTGACGGCCATGGTCGACGGCACCTCGGGCGGCACCG
ACAACTGACTTCCCTCTGAAGGTTAGATGCCCGCCG
ACAACAGCGTTTCAAAGGCCGAGGTACCCAGGAC
GTCCCCGGTCTCGGCATCGGCGGCCTCTCACTCGCCAC
CAACACTGACTTCCCTCTGAAGGTTAGATGCCCGCCG
GCATGACCTGCGAGGGCACCGTCGGCGCGTCAAGAAC
```

> 10,000 líneas

.txt

30956	72	4D2ZZ.faa.final_tree.fa -	148	9.8e-29	97.0	1.8	1	1	5.9e-32	1.1e-28	96.8	1.8	1	60	1	60	1	68	
30957	260	4D302.faa.final_tree.fa -	266	6.4e-136	448.9	1.6	1	1	7.6e-139	7e-136	448.7	1.6	9	266	3	266			
30958	86	4D302.faa.final_tree.fa -	266	0.83	5.9	6.2	1	1	0.001	0.94	5.7	6.2	46	83	37	73	21	84	0.72
30959	260	4D303.faa.final_tree.fa -	269	7e-77	255.3	6.9	1	1	2.7e-78	5e-75	249.3	6.9	26	269	3	260	1	260	
30960	206	4D304.faa.final_tree.fa -	215	8.6e-127	417.9	0.0	1	1	5.1e-130	9.4e-127	417.7	0.0	6	211	3				
30961	159	4D306.faa.final_tree.fa -	198	3.1e-05	20.4	5.0	1	1	2e-08	3.7e-05	20.1	4.1	91	150	68	127	54	158	
30962	138	4D307.faa.final_tree.fa -	141	3.2e-90	296.2	6.1	1	1	1.9e-93	3.5e-90	296.0	6.1	4	141	1	138	1	138	
30963	322	4D309.faa.final_tree.fa -	425	1.6e-75	251.7	105.0	1	2	7.3e-46	1.4e-42	143.3	13.1	1	94	1	90			
30964	322	4D309.faa.final_tree.fa -	425	1.6e-75	251.7	105.0	2	2	8.1e-37	1.5e-33	113.5	85.8		158	399	101	315		

Input

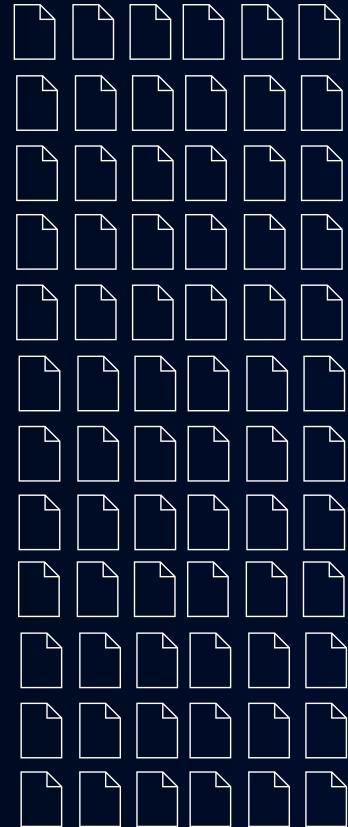
Cadenas de caracteres, archivos.

Formatos de inputs

(fasta, fastq, txt, ab1, json, entre otros).

T G T G G . . . G T G A C

T G T G G . . . A T G T G
C A T G T . . . G T G T C
G A C G C . . . G A T A G
T G G T C . . . A C A T G



bytes

kilobytes

megabytes

gigabytes

Básico

Intermedio

Avanzado

Conocimiento informático

Output

Visualización de Datos para Publicaciones Científicas

Reglas para crear visualizaciones personalizadas

Conocer los datos.

Definir la mejor opción de gráfico (barras, etc.)

Remover detalles innecesarios (sombras, bordes).

Definir colores adecuados.

Dimensiones adecuadas.

Variables categóricas asociadas a los datos.

Simplicidad > Orden > Calidad > Elegancia

EDITORIAL

Annals of Internal Medicine

Bringing Data to Life: Interactive Visualizations of Complex Data

When reflecting on his favorite part of data analysis, the esteemed statistician John Tukey (1915–2000), often called the father of exploratory data analysis, said, “Taking boring flat data and bringing it to

(2.81%) who were healthy at baseline and at 2-year follow-up but then experienced an ADL impairment 4 years after baseline. All individual participant trajectories that, when pooled, add up to the 22% of partici-



Contents lis

Environmental

journal homepage:

Short communication

Ten guidelines for effective data visualization in scientific publications

Christa Kelleher*, Thorsten Wagener

Department of Civil and Environmental Engineering, The Pennsylvania State University, 212 Sackett Building, University Park, PA 16802, USA

A

Arti

Rec

Patterns

Perspective

Principles of Effective Data Visualization

Stephen R. Midway^{1,*}

¹Department of Oceanography and Coastal Sciences, Louisiana State University, Baton Rouge, LA 70803, USA

*Correspondence: smidway@lsu.edu

<https://doi.org/10.1016/j.patter.2020.100141>



Psychological Science in the Public Interest
Volume 22, Issue 3, December 2021, Pages 110-161
© The Author(s) 2021, Article Reuse Guidelines
<https://doi.org/10.1177/15291006211051956>

SAGE
journals

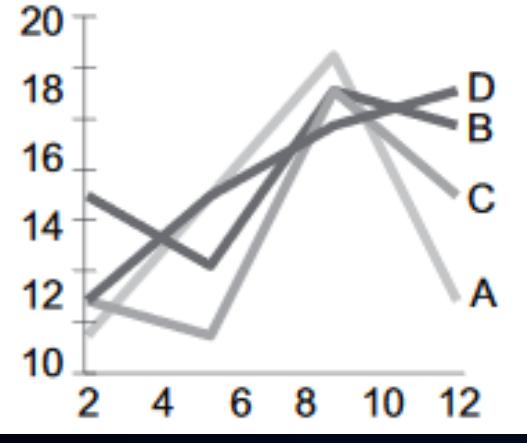
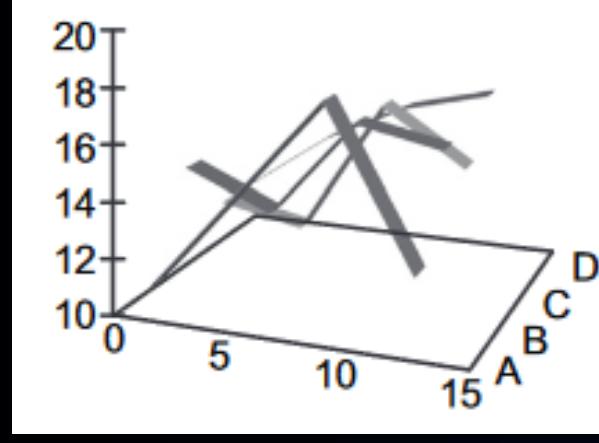
The Science of Visual Data Communication: What Works

Steven L. Franconeri¹, Lace M. Padilla², Priti Shah³, Jeffrey M. Zacks⁴, and Jessica Hullman⁵

Abstract

Effectively designed data visualizations allow viewers to use their powerful visual systems to understand patterns in data across science, education, health, and public policy. But ineffectively

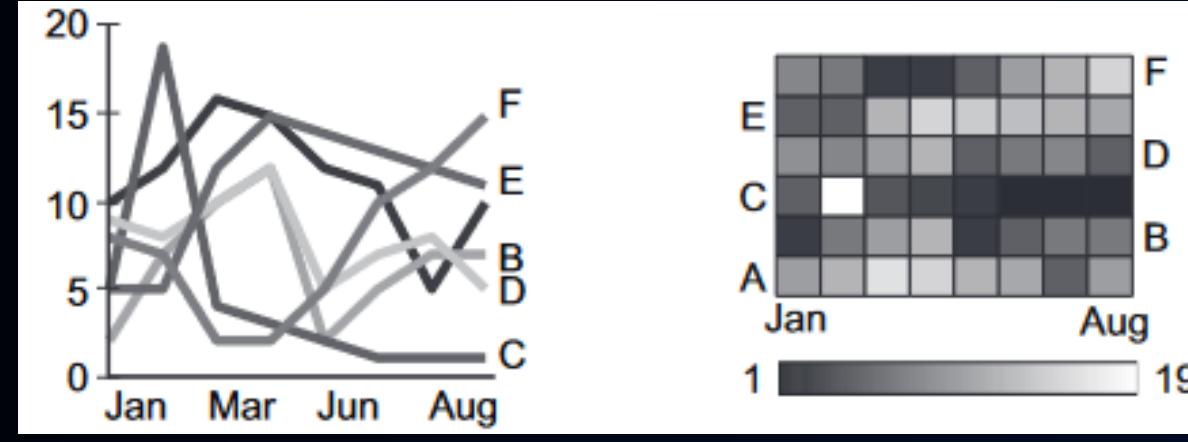
1) Crea el gráfico más simple que transmite la información que quieras transmitir.



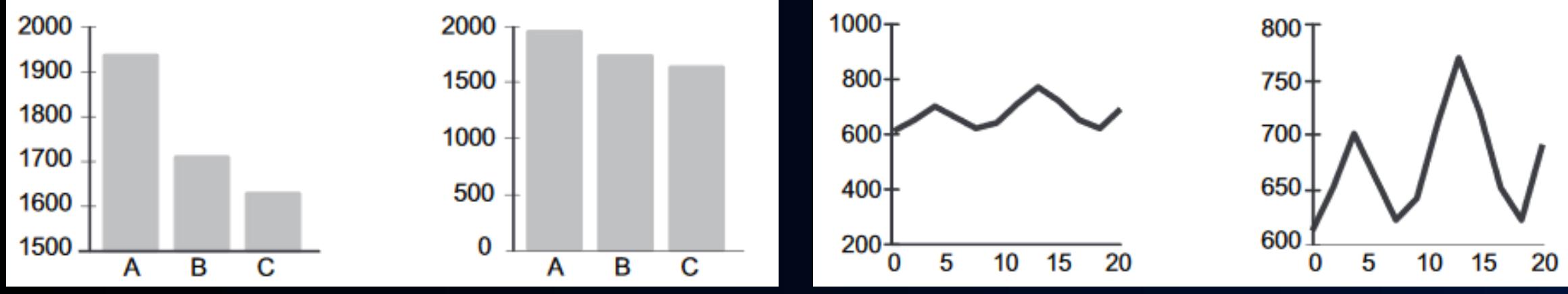
Value encoding attribute			
	Length	Width	Orientation
Form			\
	Size	Shape	Curvature
	●●●● ●●●● ●●●●	 ■■■)))))))))))))
Color	Enclosure	Blur	
	██████ ███████ ███████ █	●●●●● ●●●●● ●●●●●	
Spatial Position	Hue	Intensity	Transparency
	●●●● ●●●● ●●●●	●●●●● ●●●●● ●●●●●	●●●● ●●●● ●●●●
Spatial Position	2-D Position	Spatial Grouping	Density
Motion	Direction	Pathway	

2) Considerar el tipo de objeto de codificación y atributo utilizado para crear un gráfico (puntos, líneas, etc.)

3) Centrarse en visualizar patrones o en visualizar detalles, dependiendo del propósito de la trama.

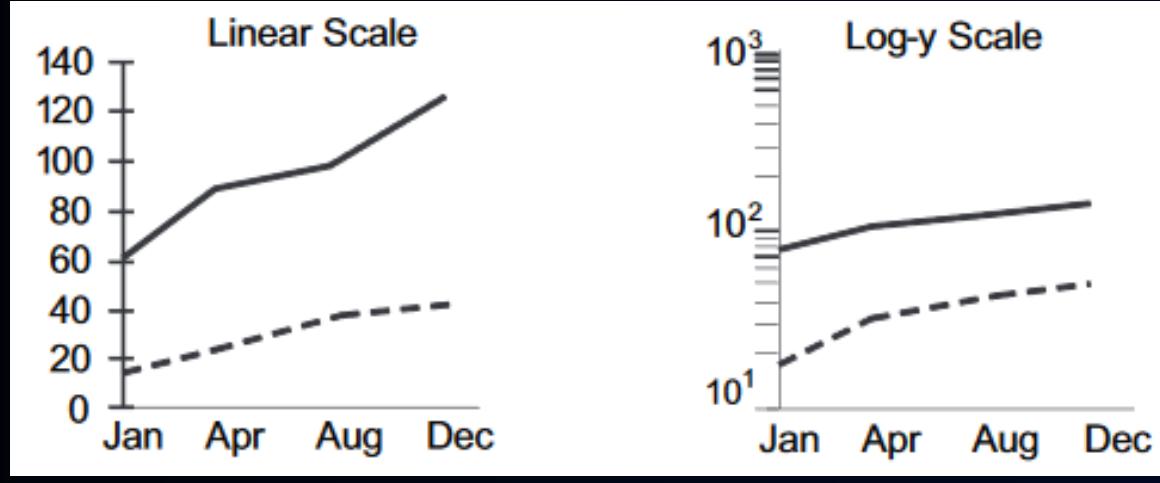


4) Seleccione rangos de ejes significativos.

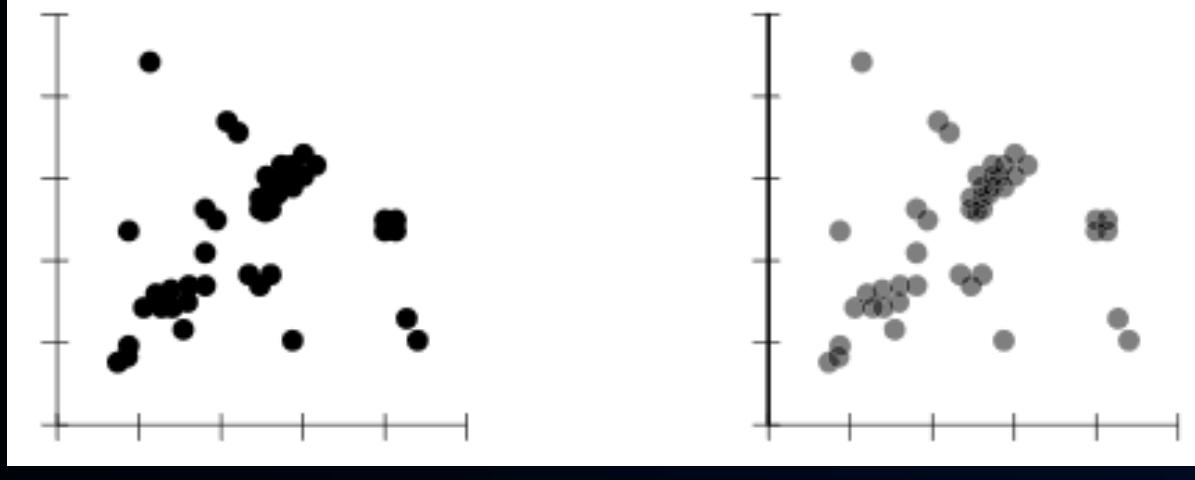


GAACACATAGCCGATCACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCCGTTTGAACGCGATTTCGAATT CAGCAGCACCGCAAGGAACATGCCAAGAAATT CGTGC

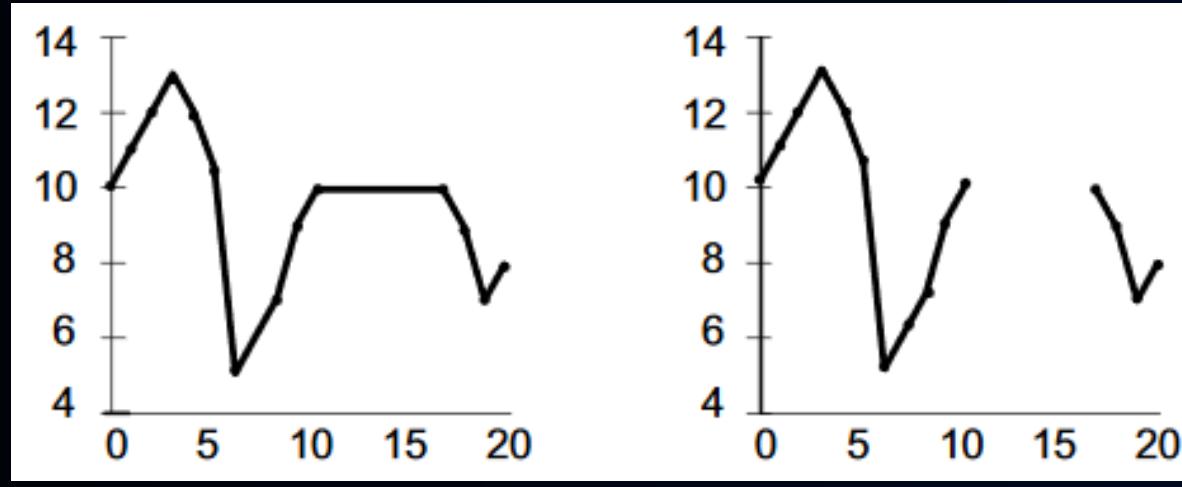
5) Elegir cuidadosamente la transformación de datos (normalización).



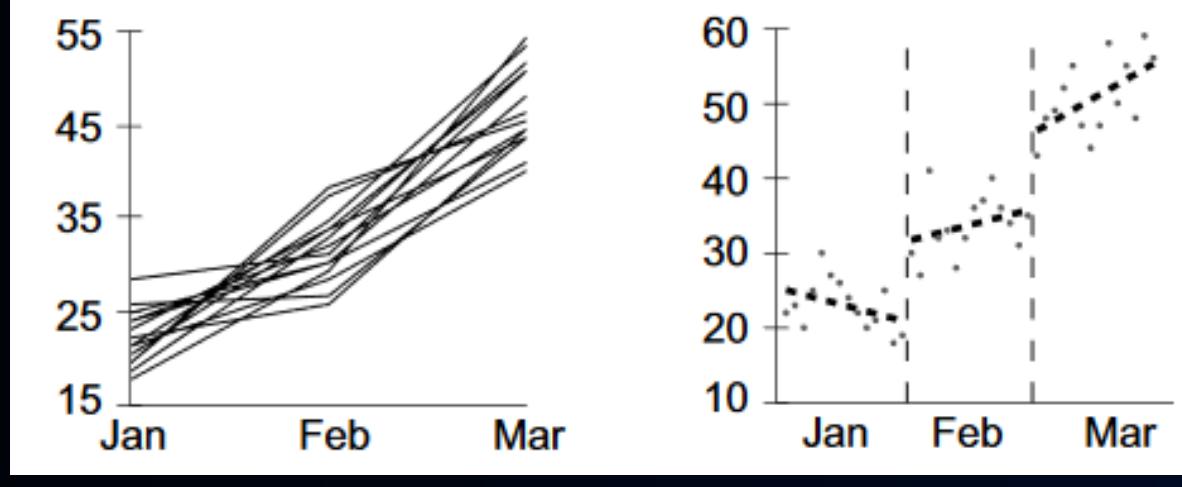
6) Graficar puntos superpuestos de manera que las diferencias de densidad se hagan evidentes en los diagramas de dispersión.



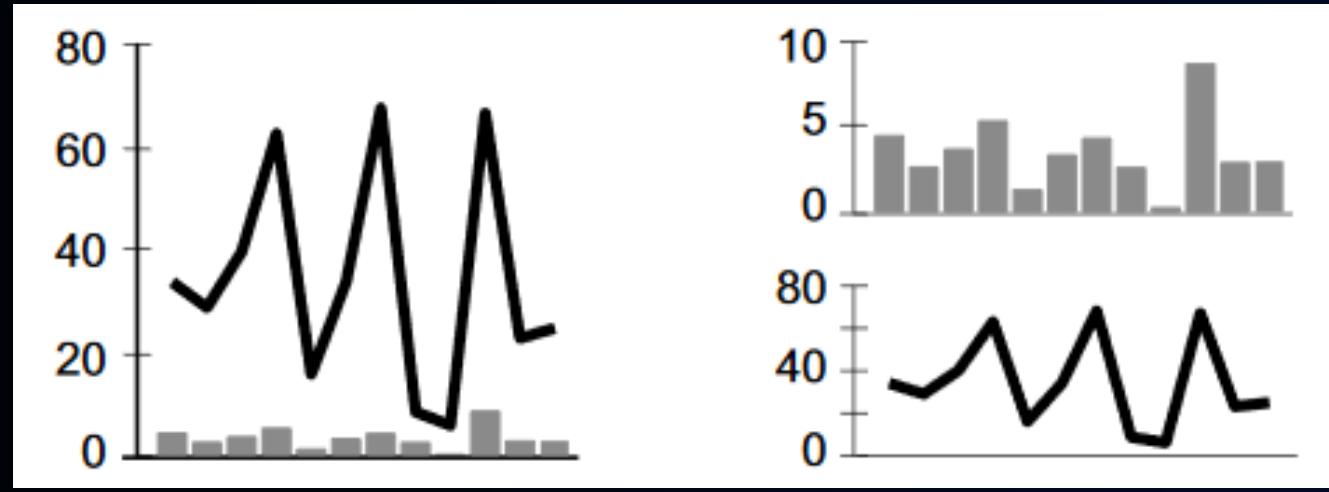
7) Utilice líneas al conectar datos secuenciales en gráficos de series temporales.



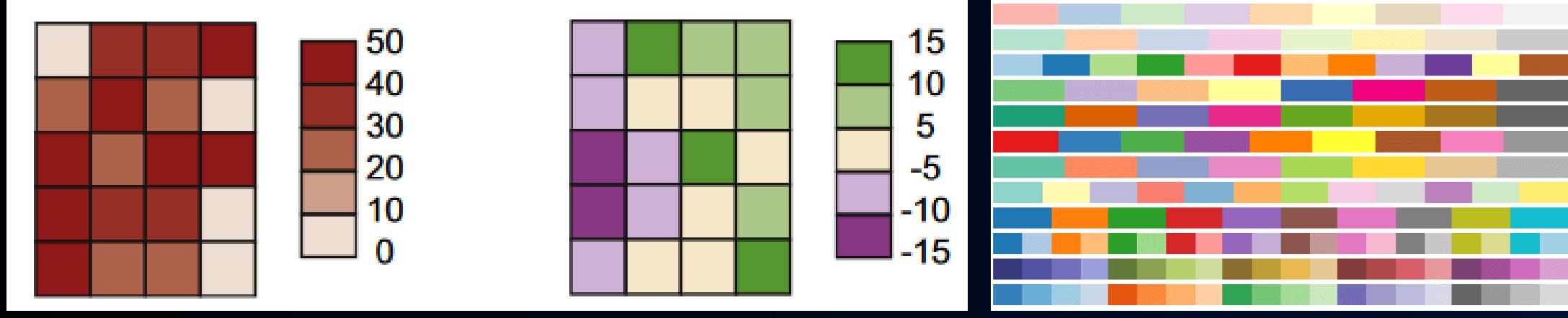
8) Agregar conjuntos de datos más grandes de maneras significativas.



9) Mantener los rangos de los ejes lo más similares posible para poder comparar variables.



10) Seleccione un esquema de color apropiado según el tipo de datos.



GAACACATAGCCGATCACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCCGTTTGAACGCGATTTCGAATT CAGCAGCACCGCAAGGAACATGCCAAGAAAATTCTGTGC

¿Qué necesito para crear
visualizaciones personalizadas?

> programar |

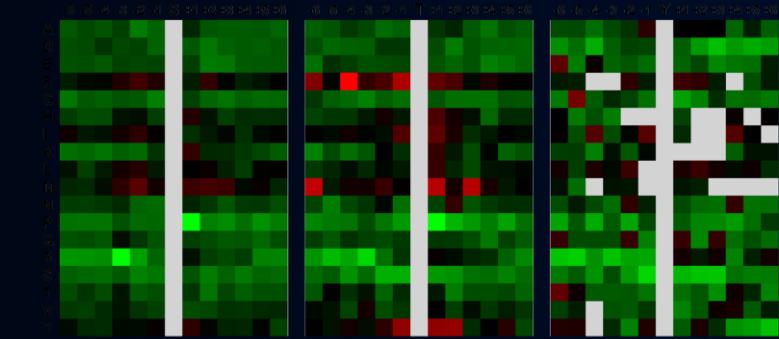
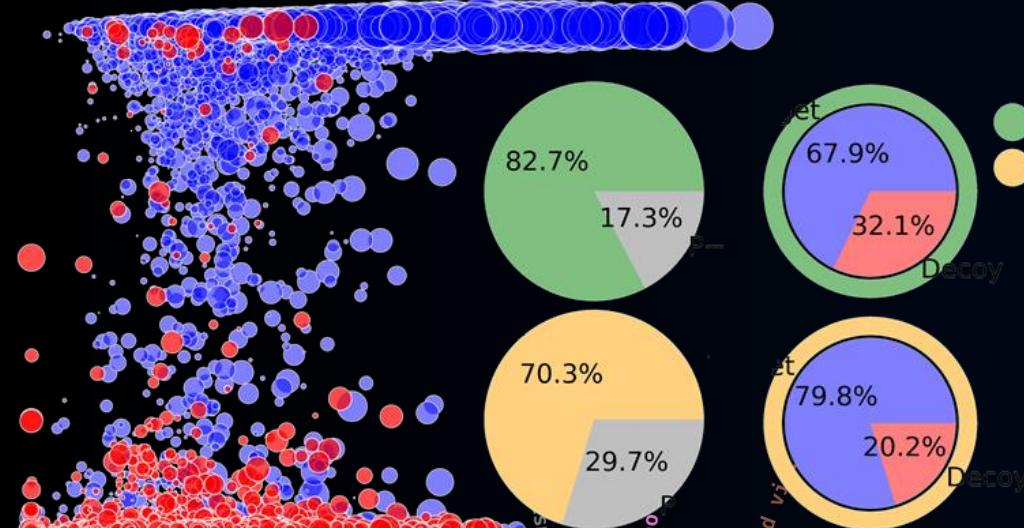
GAACACATAGCCGATACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCCGTTTGAACGCGATTTGAATTTCAGCAGCACCGCAAGGAACATGCCAAGAAAATTCTGTC

Análisis de datos y visualización

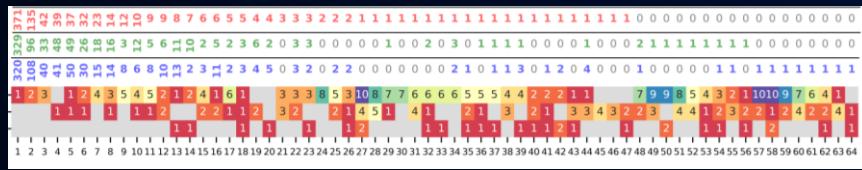
Patrones

Interacciones

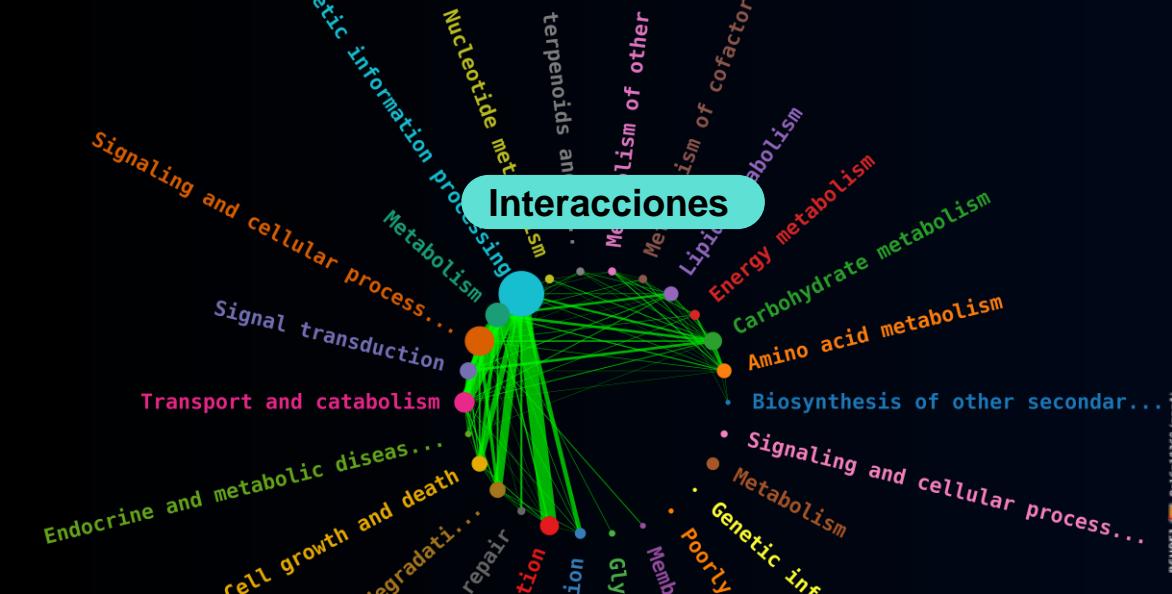
Distribuciones



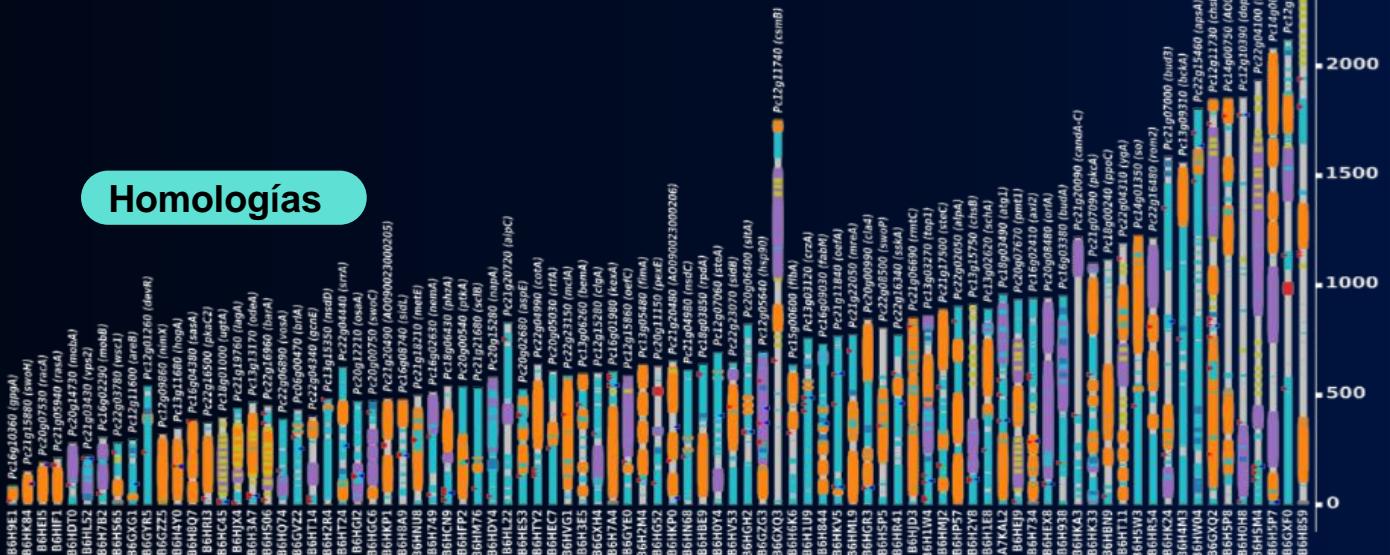
Patrones



Interacciones



Homologías



Análisis de datos y visualización

1

Order Article Reprints 

Open Access Article

NeVOmics: An Enrichment Tool for Gene Ontology and Functional Network Analysis and Visualization of Data from OMICs Technologies

by Eduardo Zúñiga-León , Ulises Carrasco-Navarro  and Francisco Fierro *  

Departamento de Biotecnología, Universidad Autónoma Metropolitana-Unidad Iztapalapa, Ciudad de Mexico 09340, Mexico

* Author to whom correspondence should be addressed.

Genes 2018, 9(12), 569; <https://doi.org/10.3390/genes9120569>

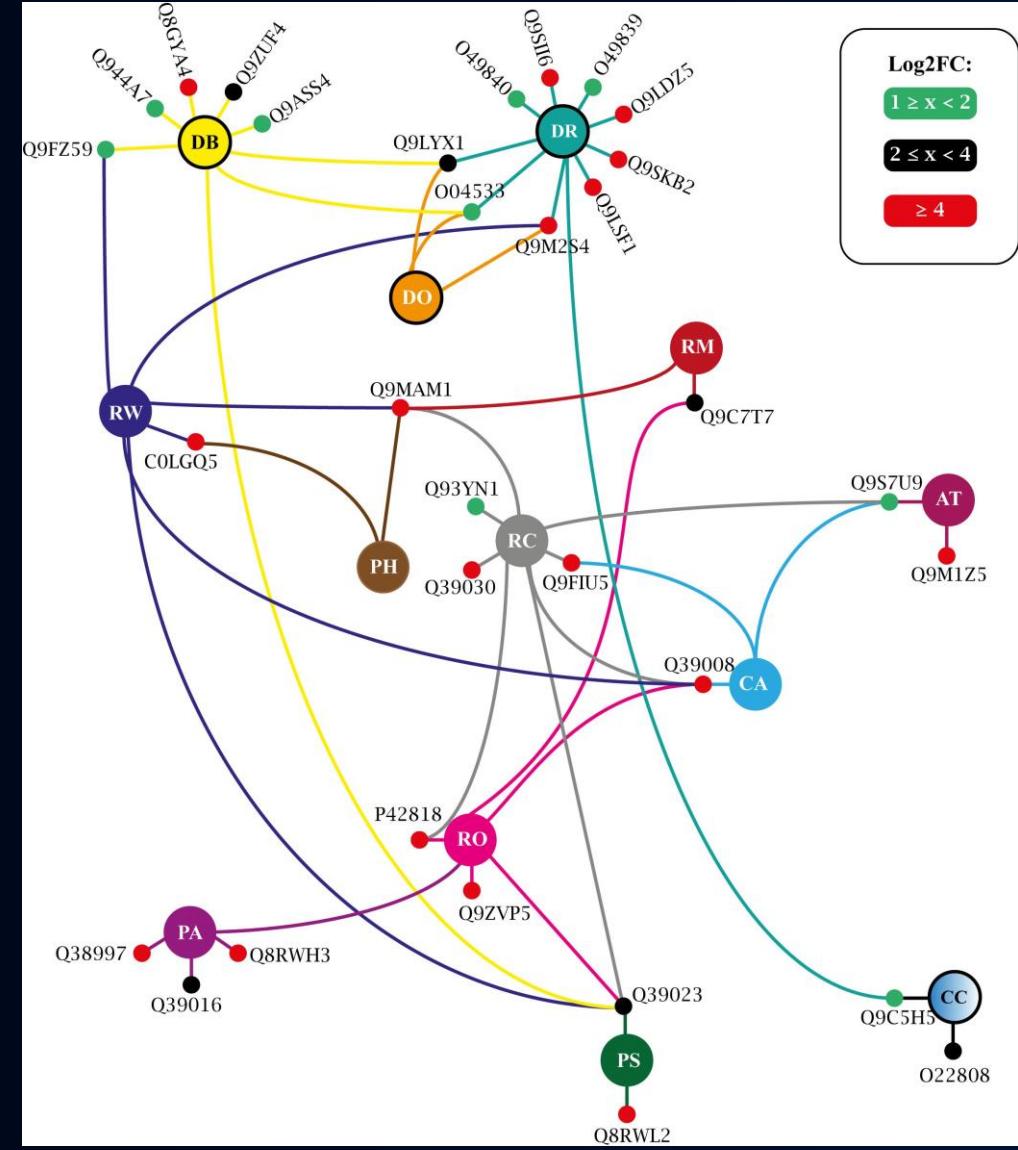
ORIGINAL RESEARCH article

Front. Plant Sci., 13 June 2021
Sec. Plant Systems and Synthetic Biology
Volume 12 - 2021 | <https://doi.org/10.3389/fpls.2021.667013>

This article is part of the Research Topic
A Systems View of Plant Cellular Communication
[View all 8 articles >](#)

The Cowpea Kinome: Genomic and Transcriptomic Analysis Under Biotic and Abiotic Stresses

José Ribamar Costa Ferreira-Neto¹ Artemisa Nazaré da Costa Borges² Manassés Daniel da Silva¹
David Anderson de Lima Moraes³ João Pacífico Bezerra-Neto² Guillaume Bourque⁴ Ederson Akio Kido¹
Ana Maria Benko-Iseppon^{2*}



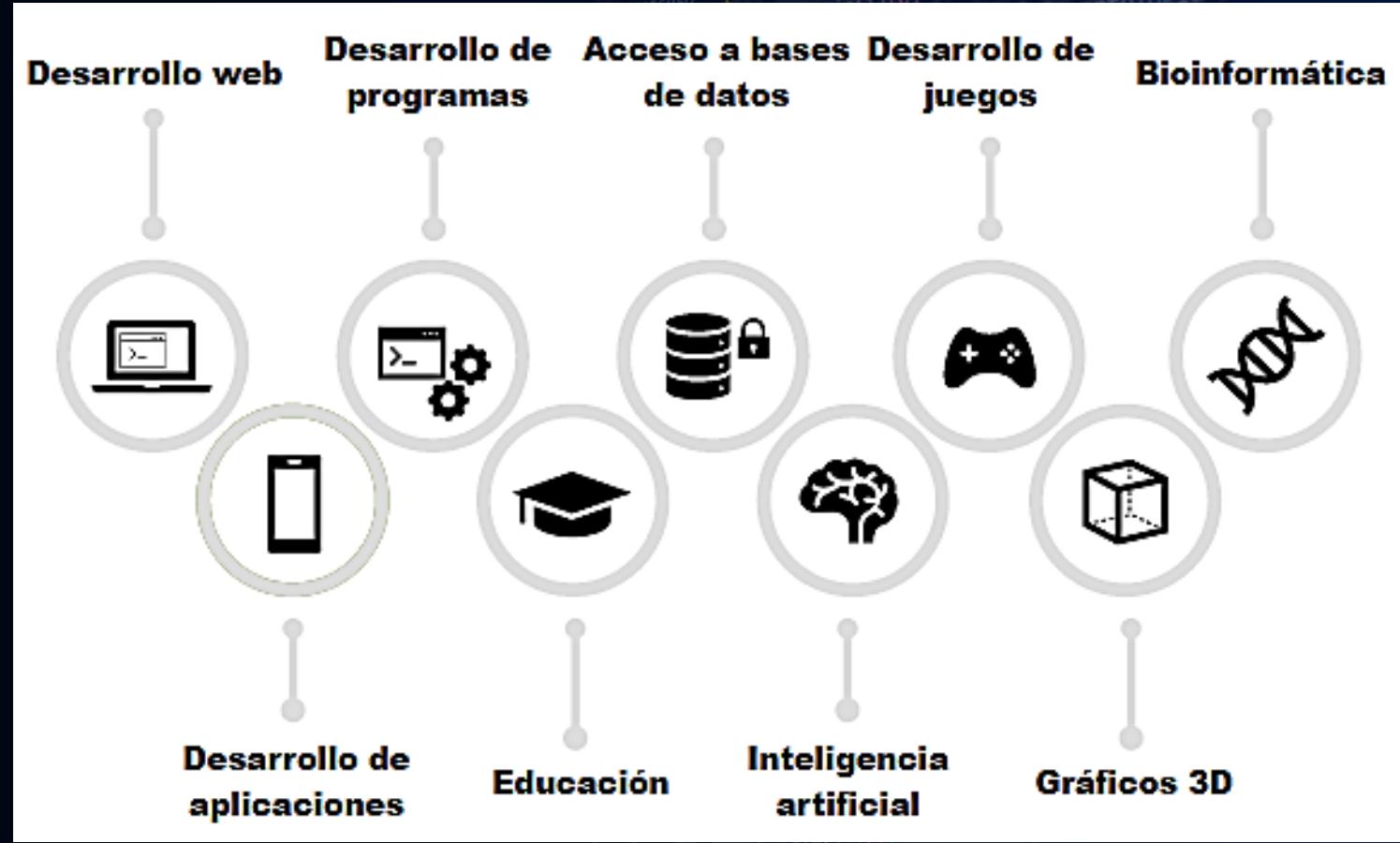
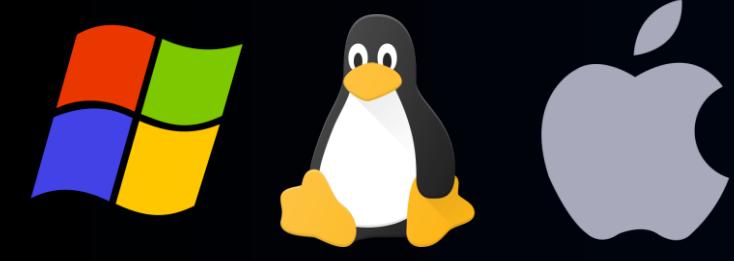
Lenguajes



GAACACATAGCCGATCACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCCGTTTGAACGCGATTTCGAATTCAAGCAGCACCGCAAGGAACATGCCAAGAAAATCGTGC

Python

- Propuesta general.
- Lenguaje de alto nivel.
- Tipado dinámico.
- Sintaxis sencilla.
- Altamente interactivo.
- Gran cantidad de bibliotecas.
- Código abierto.
- Aplicaciones científicas.
- Tiene una gran comunidad.



GAACACATAGCCGATCACCAACACCGACAAGGTACGGAATGAAGAAGATCGCACGGAACCGTTTGAACGCGATTTCGAATT CAGCAGCACCGCAAGGAACATGCCAAGAAAATCGTGC

Esencia de la programación

Estructura de datos (objetos)

Números (*int* o *float*) : 1, 2, 3, 4, 5 ó 0.1, 3.5, 11.2

Cadenas de caracteres (*str*) : "ACTGACAGTCTCA"

Listas (*list*) : ["ACG", "CCT", "ATT", "GCT"]

Tuplas (*tuple*) : ("ACG", "CCT", "ATT", "GCT")

Diccionarios (*dict*) : {"A":123, "C":"GGG", "G":"--", "T":5}

Arreglos (array)

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

DataFrames

Conjuntos (set)

`set([1, 1, 2, 3])`



{1, 2, 3}

Diagram illustrating the structure of a DataFrame:

- Column names:** Labels for the columns, such as Name, Team, Number, Position, Age, Height, Weight, College, and Salary.
- Index label:** Label for the index axis, pointing to the row index (0 to 6).
- Columns axis=1:** Labels for the columns within a row, such as Name, Team, Number, Position, Age, Height, Weight, College, and Salary.
- Index axis=0:** Label for the index axis, pointing to the row index (0 to 6).
- Data:** The actual numerical or textual values stored in the DataFrame cells.
- Missing value:** A cell containing "NaN" highlighted in red, indicating a missing value.

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0	6-8	235.0	LSU	1170960.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0	6-2	190.0	Louisville	1824360.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN	6-9	260.0	Ohio State	2569260.0
6	Evan Turner	Boston Celtics	11.0	SG	27.0	6-7	220.0	Ohio State	3425510.0

Jupyter Notebook



```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != None}

with open(carpeta+'GO_BP.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(proceso):
        if len(proceso) == e+1:
            fq.write(i+'\t'+proceso[i])
        else:
            fq.write(i+'\t'+proceso[i]+'\n')

with open(carpeta+'GO_MF.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(funcion):
        if len(funcion) == e+1:
            fq.write(i+'\t'+funcion[i])
        else:
            fq.write(i+'\t'+funcion[i]+'\n')

with open(carpeta+'GO_CC.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(componente):
        if len(componente) == e+1:
            fq.write(i+'\t'+componente[i])
        else:
```

Jupyter Notebook

Educación y ciencia.

Aplicado en programas de enseñanza.

Permite el análisis de grandes volúmenes de datos, apoya la reproducibilidad y permite visualizaciones interactivas.

Los documentos ejecutables ofrecen una manera transparente y reproducible de comunicar los resultados de una investigación.

communications physics

Explore content ▾ Journal information ▾ Publish with us ▾

nature > communications physics > comment > article

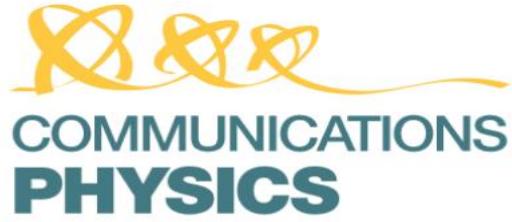
Comment | Open Access | Published: 19 August 2020

Creating an executable paper is a journey through Open Science

Jana Lasser 

Communications Physics 3, Article number: 143 (2020) | Cite this article

5286 Accesses | 3 Citations | 70 .



COMMENT

<https://doi.org/10.1038/s42005-020-00403-4>

OPEN

Creating an executable paper is a journey through Open Science

PLOS

COMPUTATIONAL BIOLOGY



Jana Lasser^{1,2}✉

PMCID: PMC7643937
PMID: 33151926

PLoS Comput Biol. 2020 Nov; 16(11): e1008326.

Published online 2020 Nov 5. doi: [10.1371/journal.pcbi.1008326](https://doi.org/10.1371/journal.pcbi.1008326)

Using interactive digital notebooks for bioscience and informatics education

Alan Davies,* Frances Hooley, Peter Causey-Freeman, Iliada Eleftheriou, and Georgina Moulton

Francis Ouellette, Editor

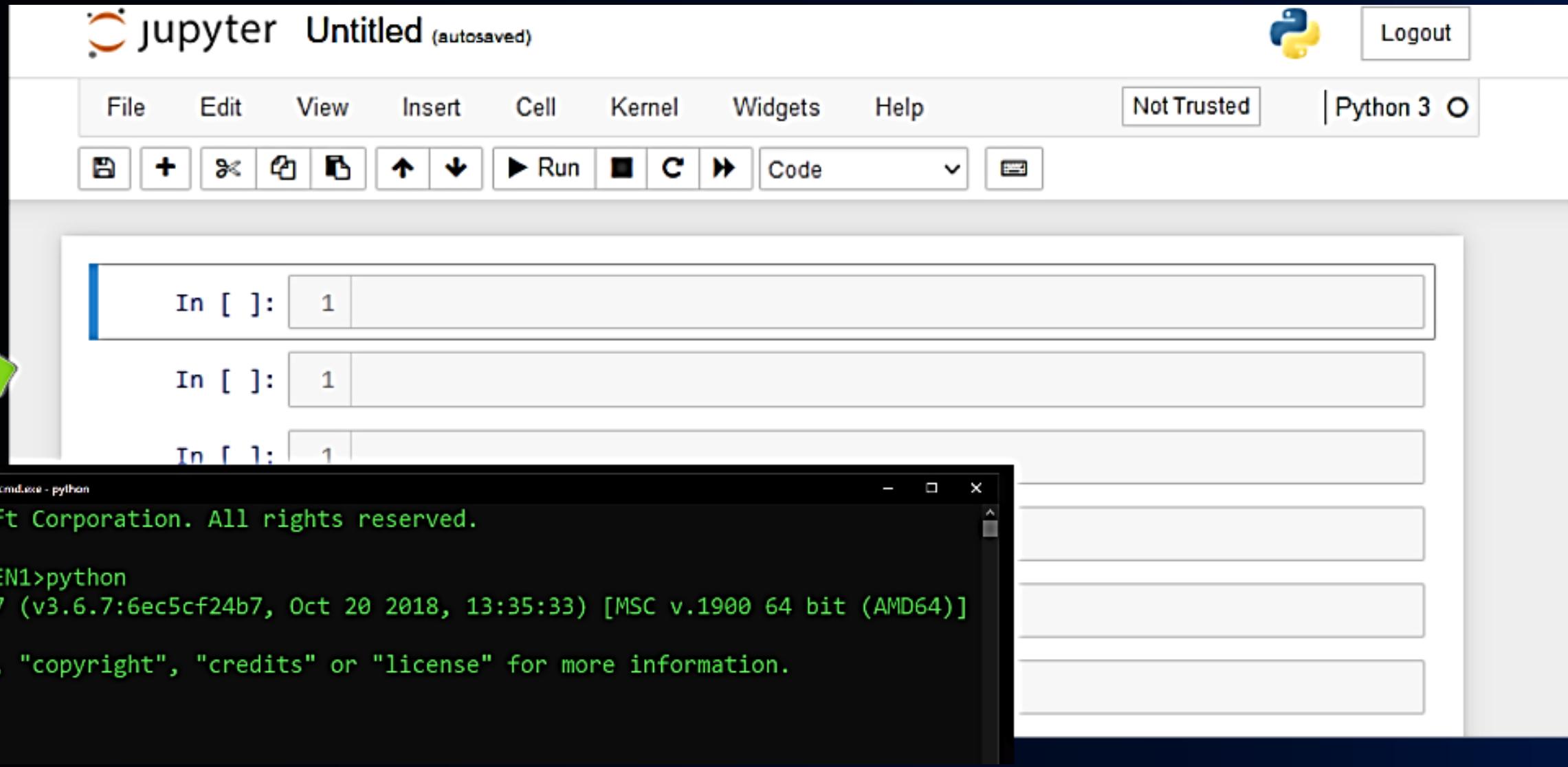
► Author information ► Copyright and License information ► Disclaimer

Abstract

Interactive digital notebooks provide an opportunity for researchers and educators to carry out data analysis and report the results in a single digital format. Further to just being digital, the format allows for rich content to be created in order to interact with the code and data contained in such a notebook to form an educational narrative. This primer introduces some of the fundamental aspects involved in using Jupyter notebooks in an

Go to: 

Entorno de programación



Entorno de trabajo

Documentos

nov2024

- notebooks
- archivos

Sesión 1

Python.
Jupyter Notebook.
Variables.
Operadores booleanos.
Listas y tuplas (list, tuple).
Diccionarios (dict).
DataFrame (Pandas).
Condicionales if, elif y else.
Bucle for.

```
proceso = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
funcion = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
componente = {i: ontologia[i][0] for i in ontologia if ontologia[i][0] != ""}
with open(carpeta+ '/GO_BP.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(proceso):
        if len(proceso) == e+1:
            fq.write(i+'\t'+proceso[i])
        else:
            fq.write(i+'\t'+proceso[i]+'\n')
with open(carpeta+ '/GO_MF.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(funcion):
        if len(funcion) == e+1:
            fq.write(i+'\t'+funcion[i])
        else:
            fq.write(i+'\t'+funcion[i]+'\n')
with open(carpeta+ '/GO_CC.txt', 'w') as fq:
    fq.write('GO\tTerm\n')
    for e, i in enumerate(componente):
        if len(componente) == e+1:
            fq.write(i+'\t'+componente[i])
        else:
```