



Visualización de datos

Cristóbal Honores

Resumen clase anterior

1. Regresión lineal.
2. Método de Stepwise.
3. PCR.
4. PLSR.



Regresión lineal

Ajusta los datos obtenidos a un modelo lineal.

Puede realizarse con uno o múltiples regresores.

Se utiliza el comando `lm()`.

```
lm(base$variable_x ~ base$variable_y)
```

Para obtener los valores de los coeficientes y resumen del modelo se utiliza el comando `summary()`.

A decorative network diagram at the bottom of the slide, consisting of several light blue circular nodes connected by thin, light blue lines, forming a web-like structure.

Método de Stepwise

Selecciona cuales son los mejores regresores para un modelo de regresión.

Trabaja atreves de criterios de selección de variables.

Utiliza el comando `step()`.

Puede realizarse mediante le método de forward, backward o both.

A decorative network diagram at the bottom of the slide, consisting of several light blue circular nodes connected by thin, light blue lines, forming a web-like structure.

PCR

Realiza un análisis de componentes principales con el fin de disminuir la dimensionalidad de la matriz de regresores.

Tiene el objetivo de minimizar el error cuadrático medio de predicción.

Se utiliza con el comando `pcr()` del paquete `pls`.



PLSR

Trabaja bajo principios similares a PCR, con la diferencia que utiliza una estrategia de reducción de dimensiones supervisada por el resultado.

Mientras PCR disminuye la varianza de X , PLSR explica también la covarianza entre X e Y .

Se utiliza el comando `pls()` del paquete `pls`.




Propósito de la clase

Lograr comunicar de manera efectiva los resultados obtenidos con R.



Visualización de datos R

1. Principios básicos de la visualización de datos.
 2. Personalizar gráficos básicos de R .
 3. Paquete ggplot2.
 4. Paquete Plotly.
 5. Tipos de gráficos.
- 
- A decorative network diagram at the bottom of the slide, featuring several light blue circular nodes connected by thin, light blue lines, creating a web-like structure.

Principios básicos de la visualización de datos

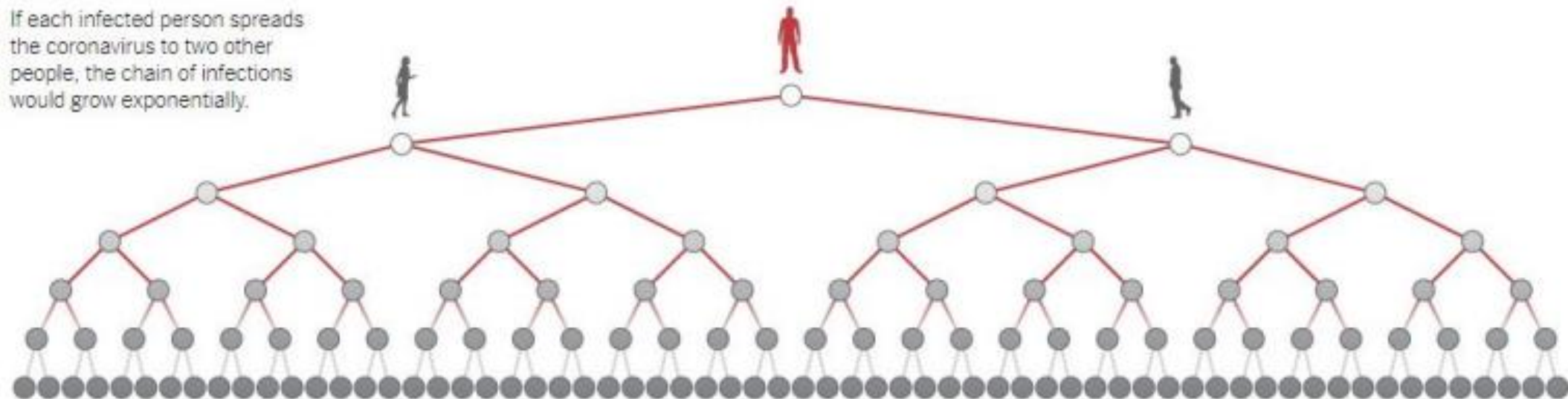
- La importancia de una visualización efectiva.
- Tres leyes de una visualización efectiva.
 - Tener un propósito claro
 - Mostrar los datos claramente
 - Hacer obvio el mensaje



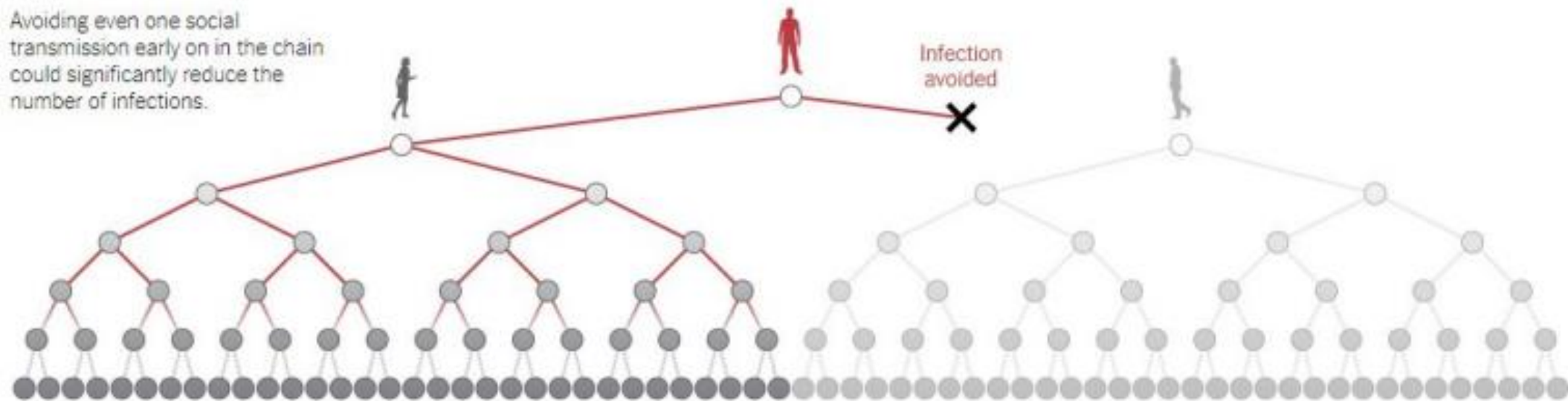
Cutting a Link in the Chain of Transmission

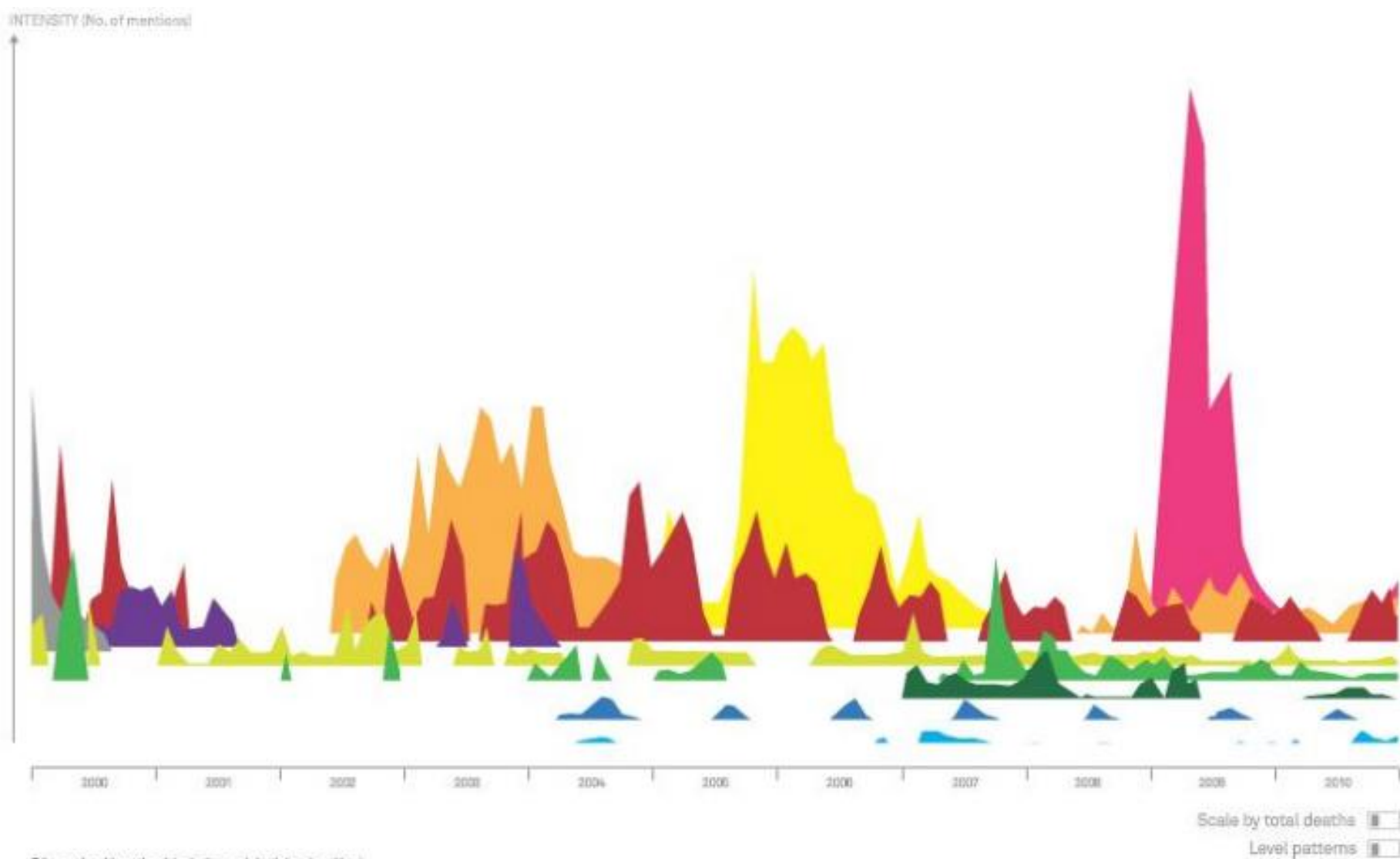
A simple tree diagram shows how limiting contacts early might prevent many infections.

If each infected person spreads the coronavirus to two other people, the chain of infections would grow exponentially.



Avoiding even one social transmission early on in the chain could significantly reduce the number of infections.





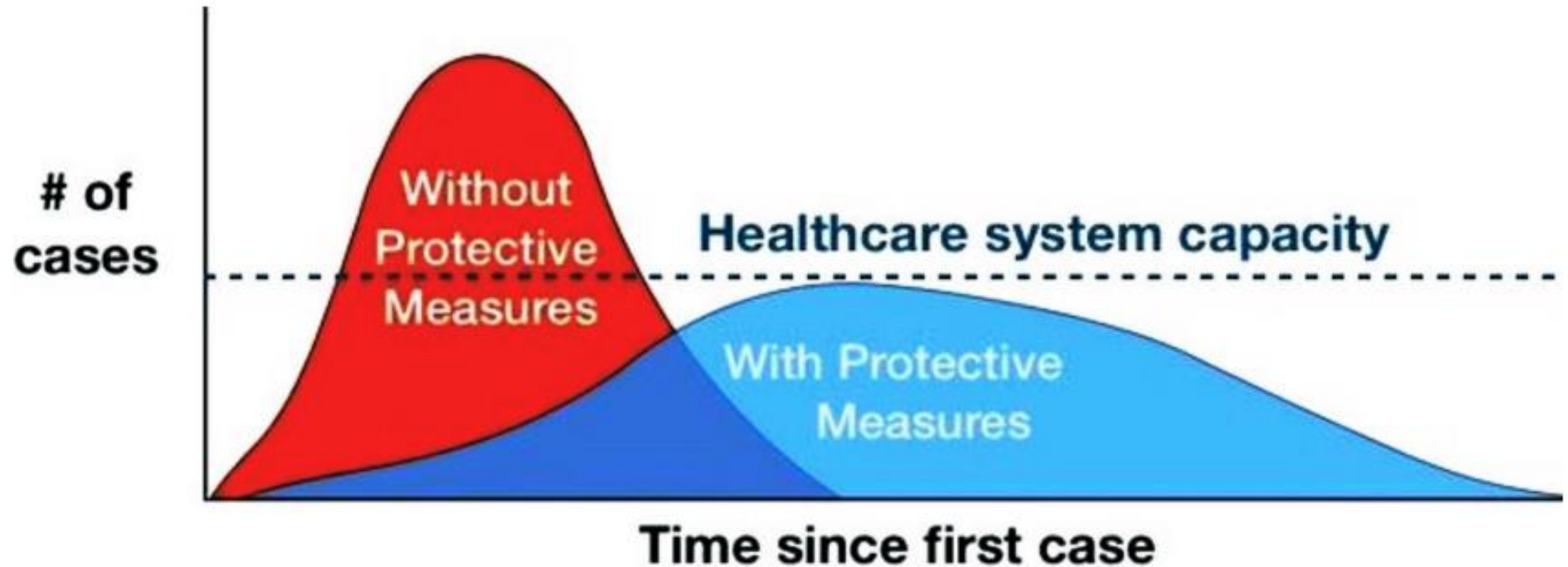
Story (estimated total worldwide deaths)

- | | | | |
|-------------------------------|-----------------------------|---------------------------|----------------------|
| ▲ Killer wasps (11,000) | ▲ Autism & vaccinations (0) | ▲ Mad Cow Disease (204) | ▲ Bird Flu (260) |
| ▲ Killer wifi (0) | ▲ Asteroid collisions (0) | ▲ Violent video games (0) | ▲ Swine Flu (18,000) |
| ▲ Mobile phones & tumours (0) | ▲ The Millennium Bug (0) | ▲ SARS (770) | |

David McCandless & Joshua Lee // informationisbeautiful.net // @infobeautiful // v1.0
 source: Google Insights & News Timeline // note: Seasonal flu has killed around 3.75m people since 2001

Mountains Out of Molehills
 A timeline of global media scare stories

La importancia de una visualización efectiva.



Adapted from CDC / The Economist

Un grafico efectivo...

- Es visualmente atractivo, intuitivo y legible.
- Usa el grafico y escala de ejes correctos.
- Usa la proximidad y alineación para facilitar la comparación.
- Usa etiquetas y anotaciones para agregar claridad al mensaje.

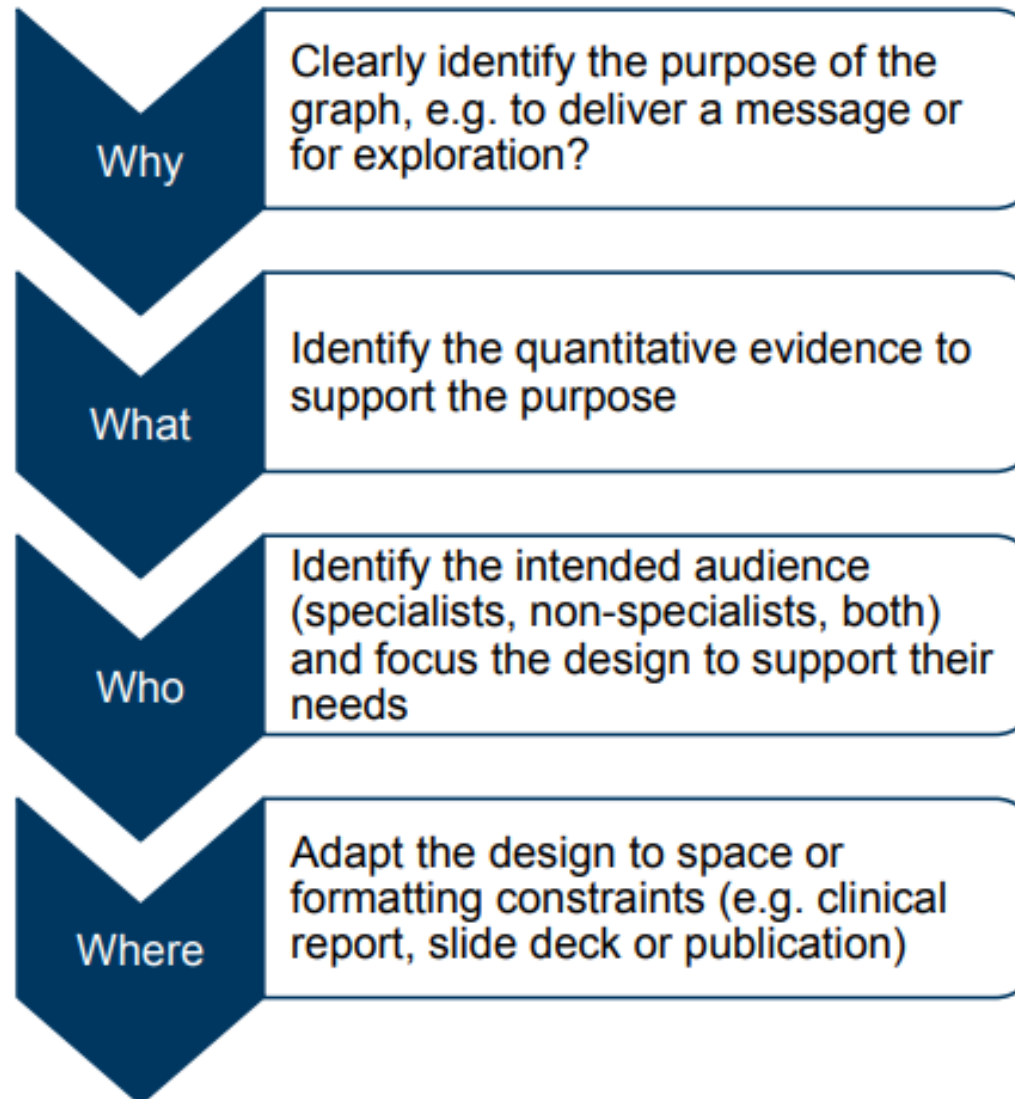
Una visualización efectiva...

- Permite una comunicación clara e impactante.
- Eleva nuestra influencia con nuestros grupos de interés.
- **Facilita la toma de decisiones!!!**

Las tres leyes de una visualización efectiva.

1. Tener el propósito claro.
2. Mostrar los datos claramente.
3. Hacer obvio el mensaje.

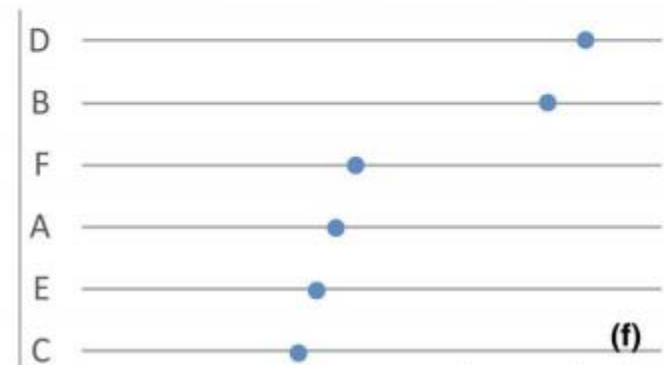
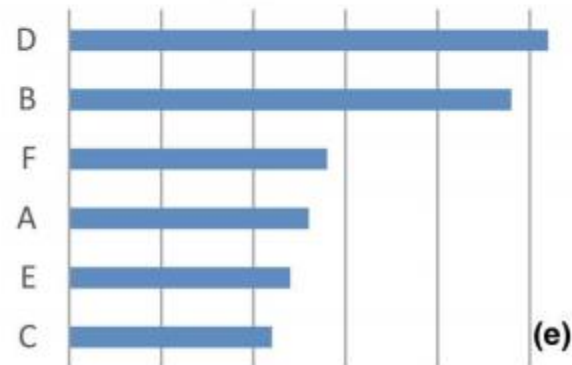
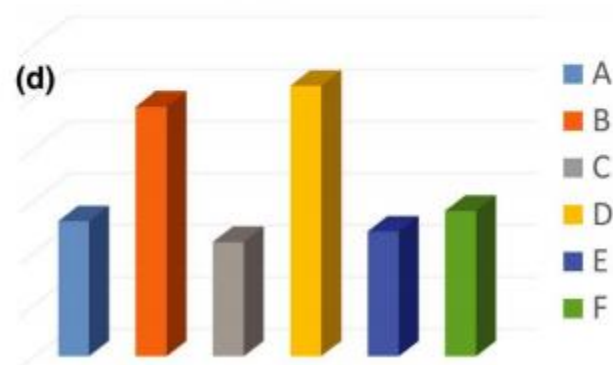
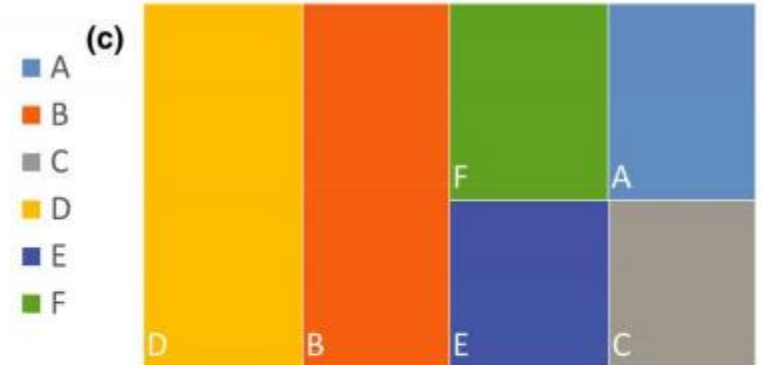
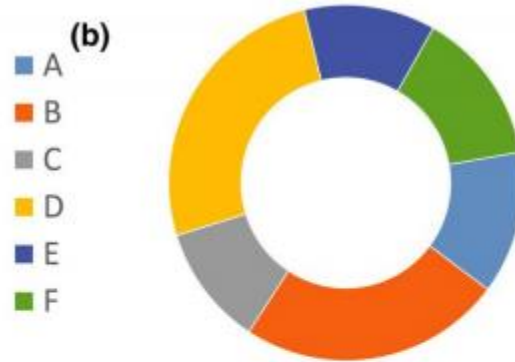
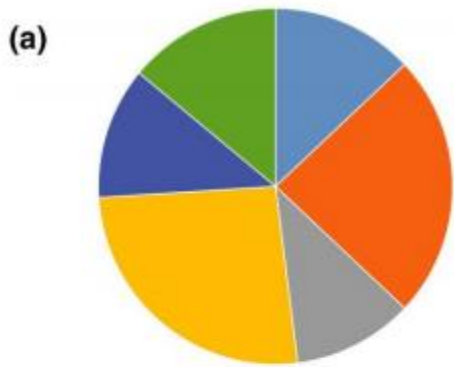
1. Tener el propósito claro.



2. Mostrar los datos claramente.

1. Elegir el gráfico.
2. Escala de adecuada para los ejes.
3. Agregar espacio entre los gráficos (en caso que sean mediciones por tiempo).

¿Cualquier gráfico sirve?



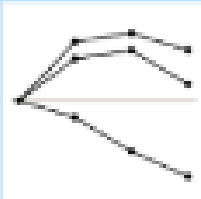
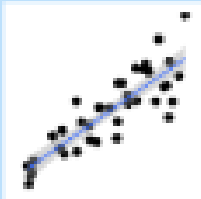
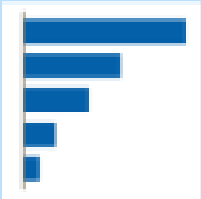

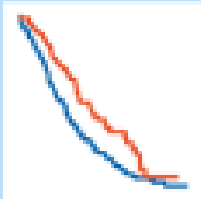
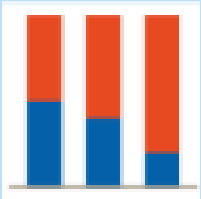
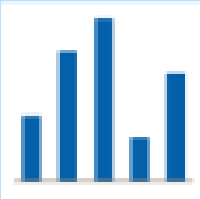
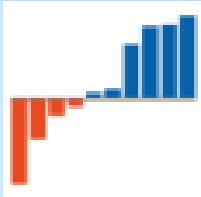
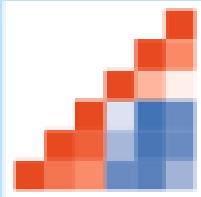
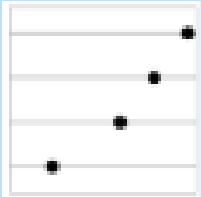
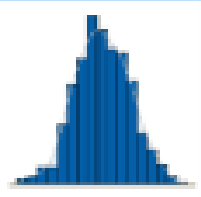
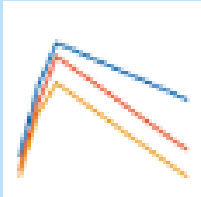
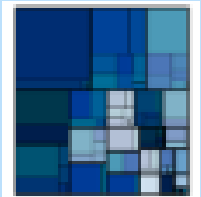
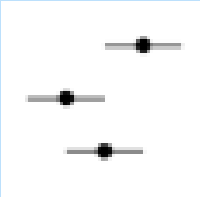
Elegir el tipo de gráfico

Selecting the right base graph

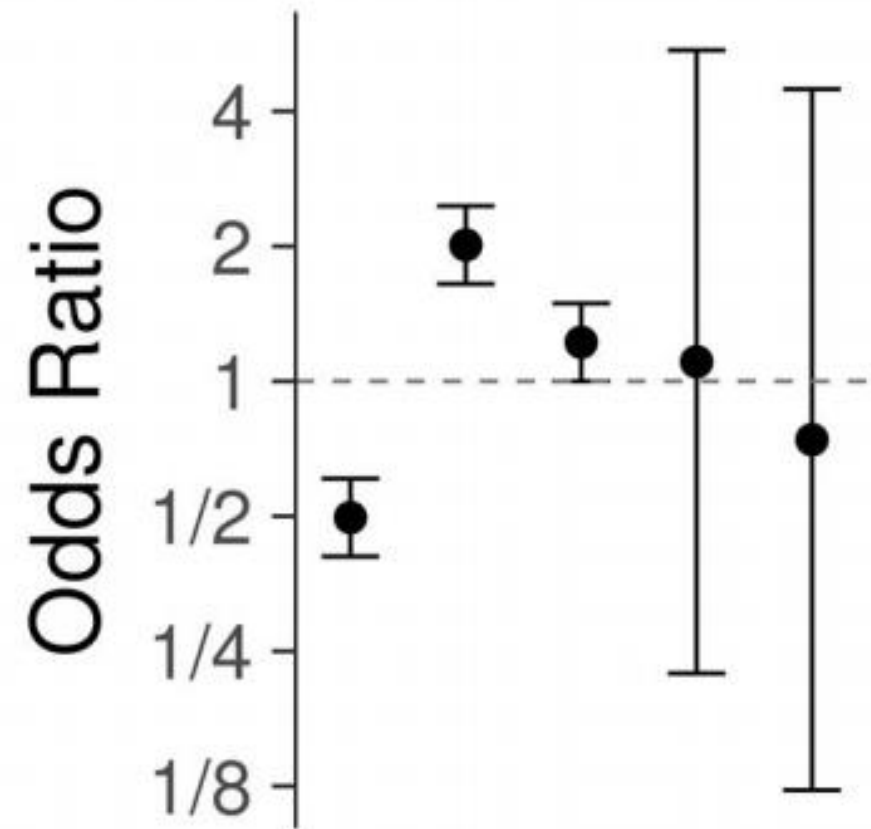
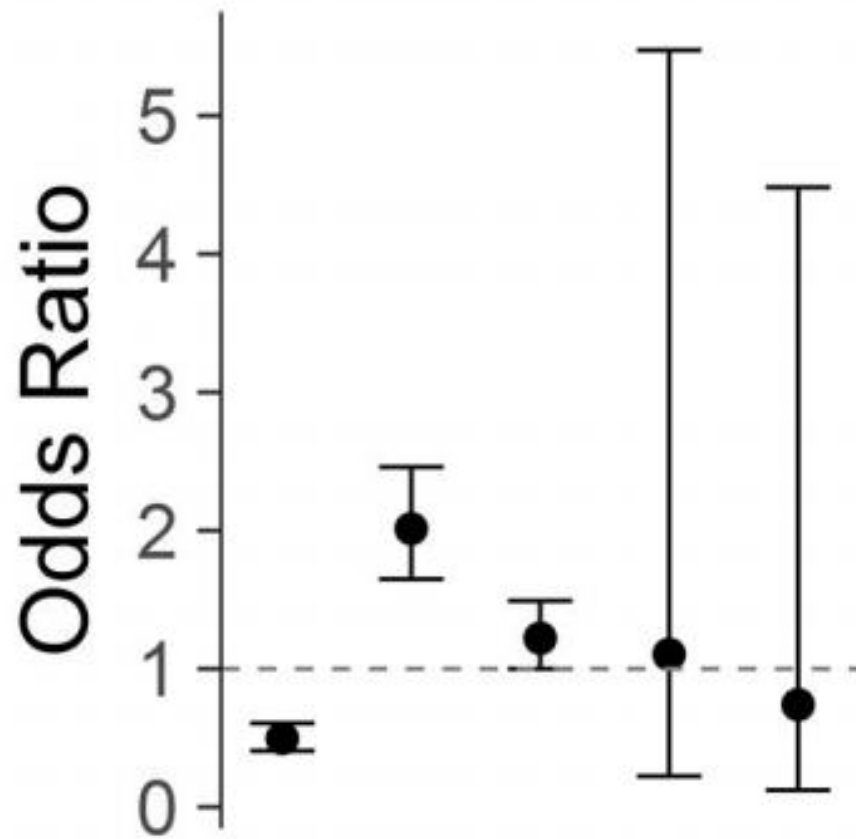
Consider if a standard graph can be used by identifying suitable designs based on the:

(i) **purpose** (i.e. message to be conveyed or question to answer) and (ii) **data** (i.e. variables to display).

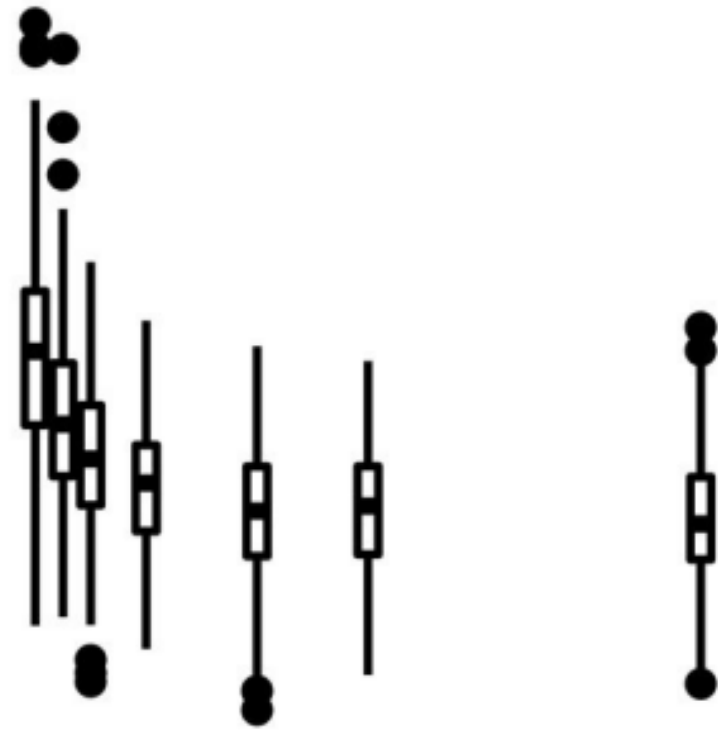
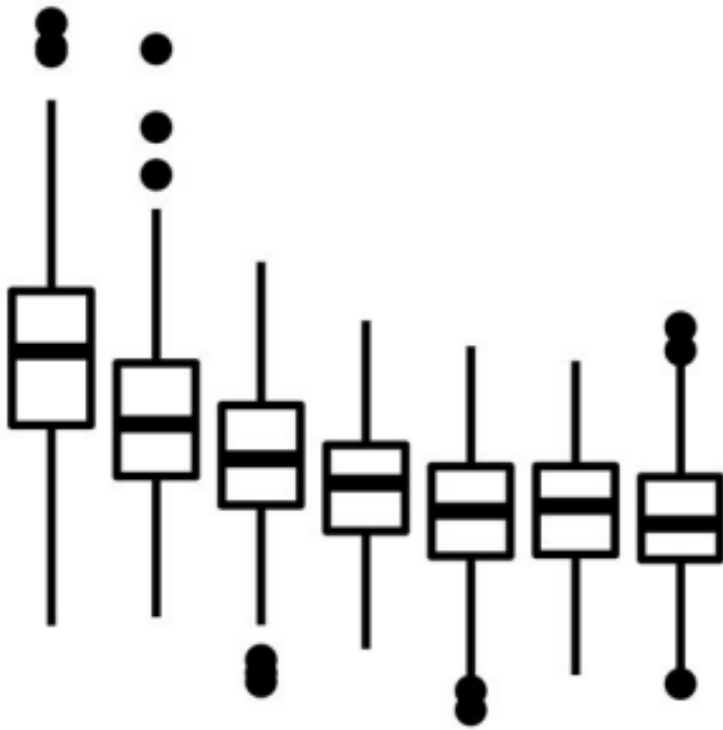
Example plots categorized by purpose

Deviation	Correlation	Ranking	Distribution	Evolution	Part-to-whole	Magnitude
Chg. from baseline	Scatter plot	Horizontal bar chart	Boxplot	Kaplan Meier	Stacked bar chart	Vertical bar chart
						
Waterfall	Heat map	Dotplot	Histogram	Line plot	Tree map	Forest plot
						

Escala de adecuada para los ejes.



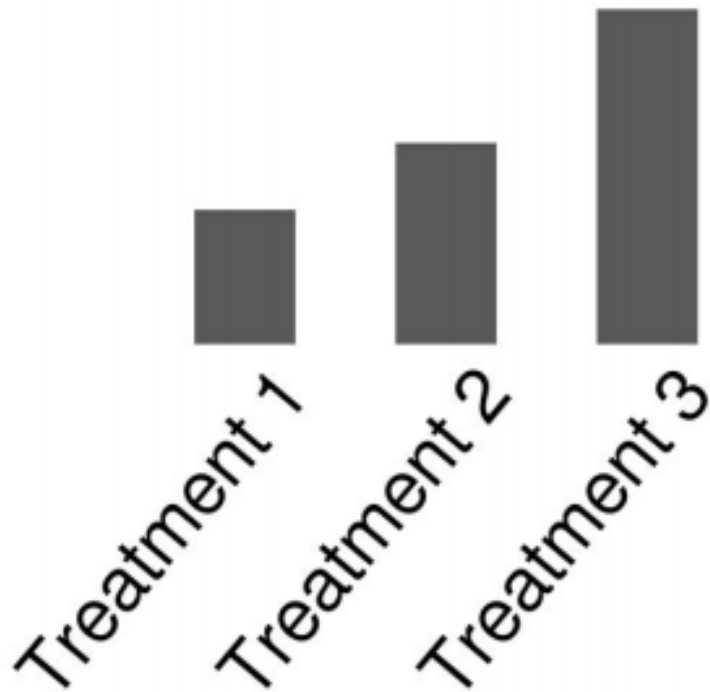
Espacio entre gráficos.



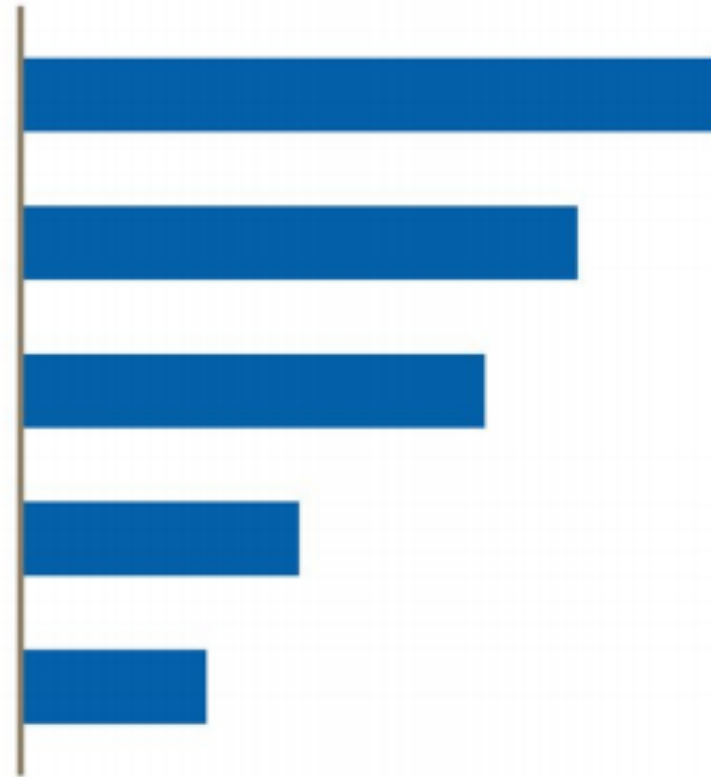
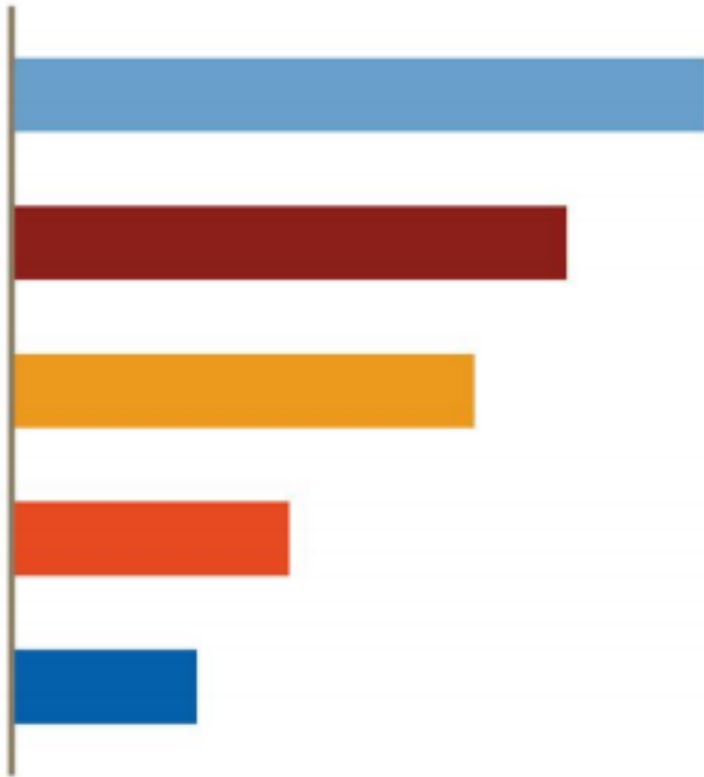
3. Hacer obvio el mensaje.

1. Evitar textos en ángulos.
2. No utilizar color innecesariamente.
3. Usar color para resaltar y comparar.
4. Agregar líneas y leyendas que aporten información.
5. Utilizar un color adecuado.

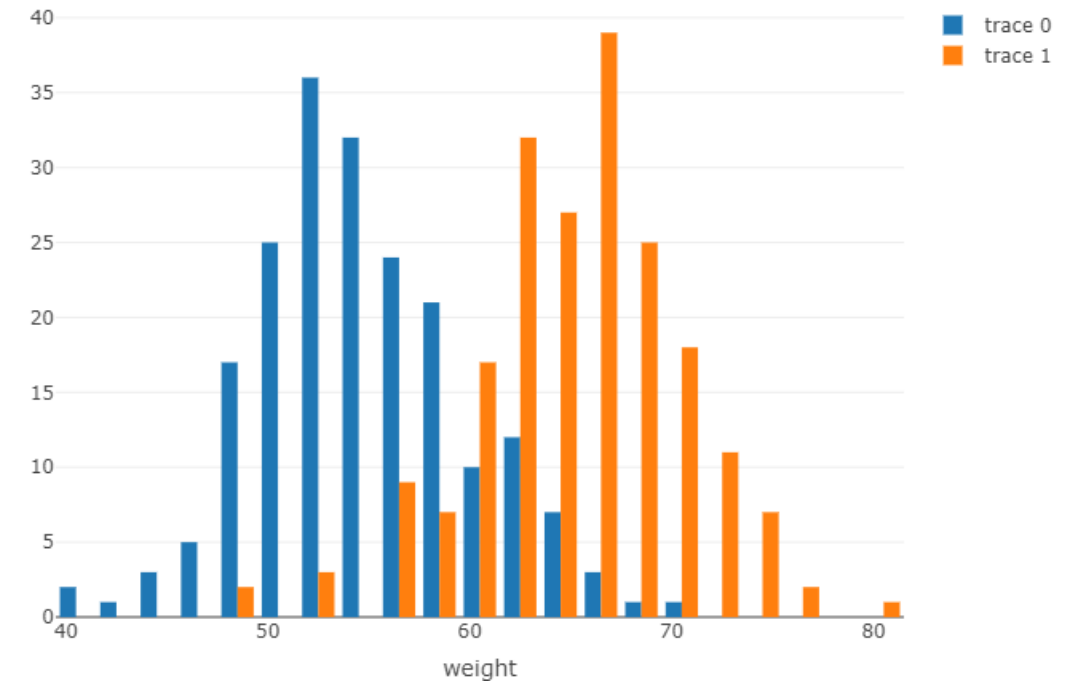
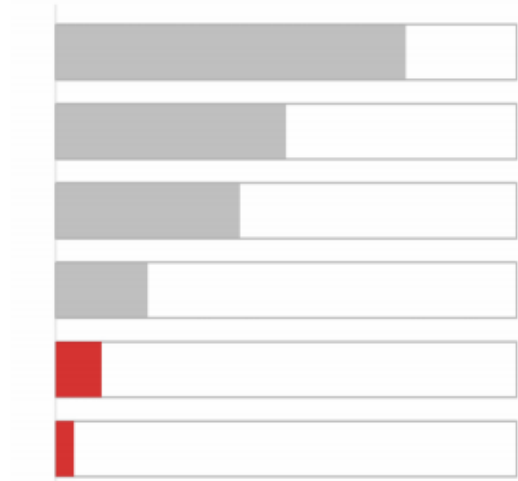
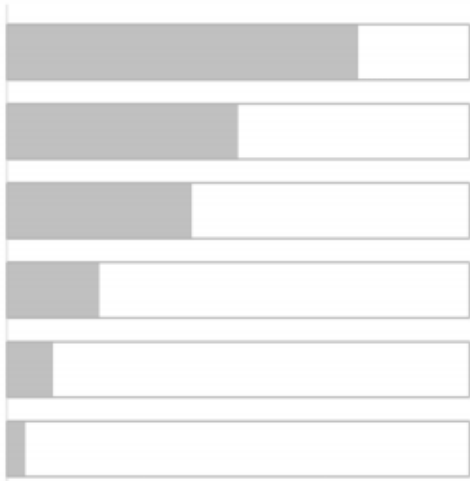
Evitar textos en ángulos



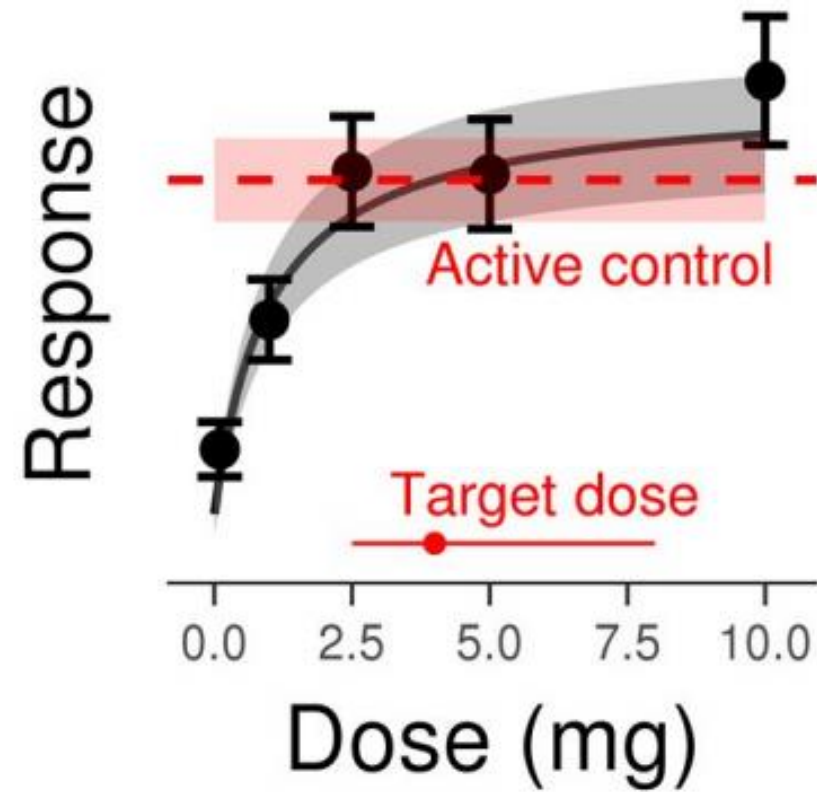
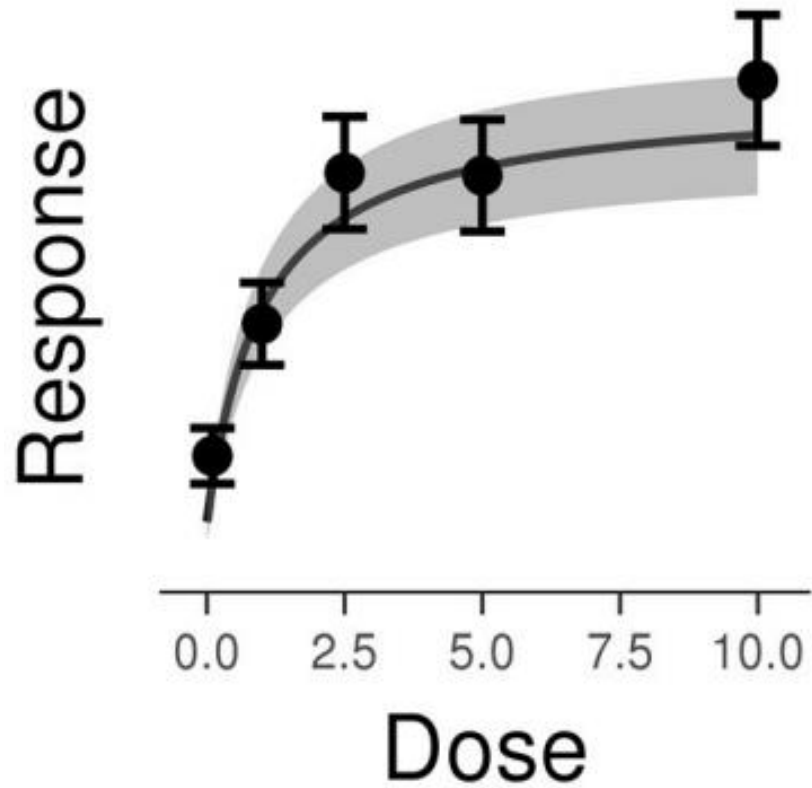
Mal uso del color



Buen uso del color



Agregar líneas y leyendas



Colores adecuados

- Utilizar colores institucionales.
 - Permite utilizar los gráficos obtenidos mas fácilmente.
- De no tener colores institucionales:
 - <https://htmlcolorcodes.com/es/selecto-de-color/>

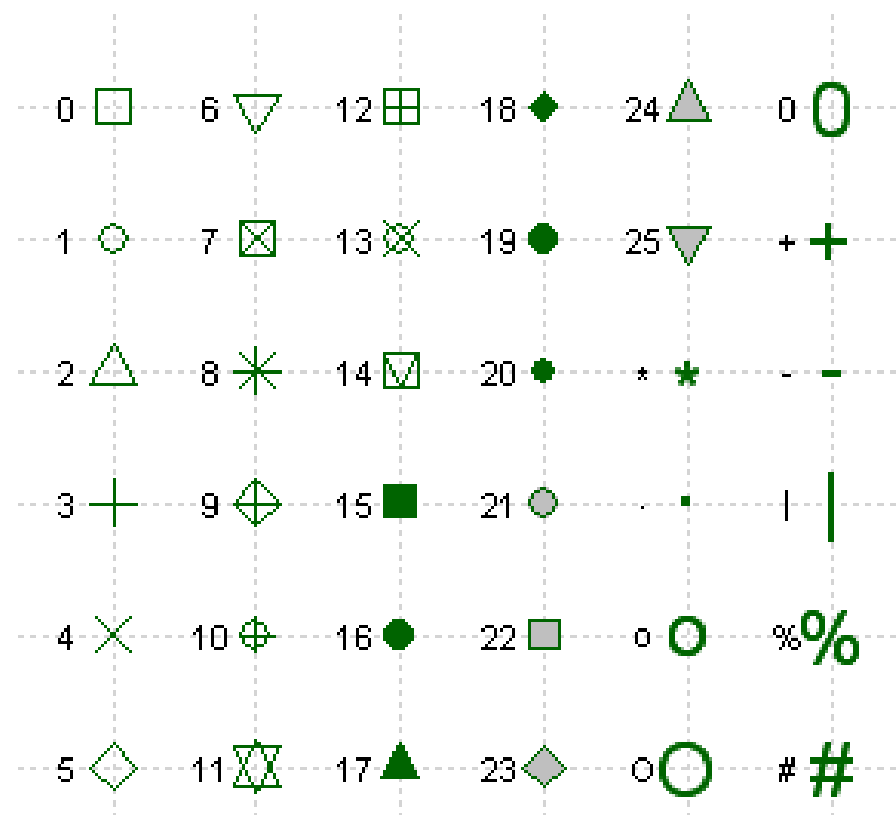
Parámetros gráficos en R

1. Símbolos.
2. Tipos de líneas
3. Color
4. Fuente
5. Margen y tamaño del grafico
6. Titulo
7. Líneas de referencia

Símbolos en R

- Con el argumento `pch`: se puede elegir el símbolo con el cual se quiere graficar.

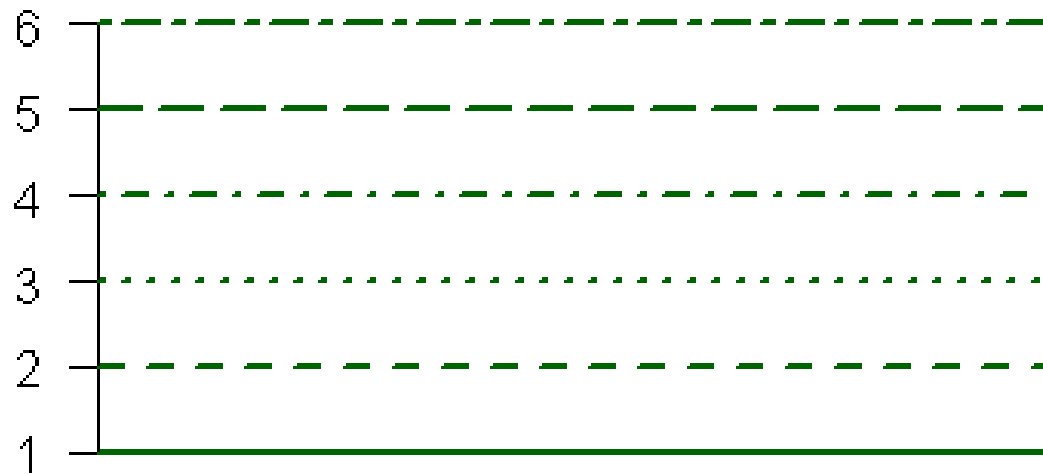
plot symbols : `pch =`



Tipos de líneas

- Con el argumento `lwd`: se puede seleccionar el ancho de la línea (1 por default) donde 2 es el doble de ancho de 1.
- Con el comando `lty`: se puede seleccionar el tipo de línea a graficar

Line Types: `lty=`



Colores

- `col`: permite agregar color al grafico, puede utilizarse con números del 1 al 657, con el nombre entre comillas, con código RGB o hexadecimal con `#`.
- `col.axis`, `col.lab`, `col.main`, `col.sub`: permite agregarle color a las anotaciones, ejes, titulo y subtítulo.
- `Bg`: permite agregarle color al fondo del grafico.

Fuentes

- font: Especifica el tipo de fuente, 1 = plano, 2 = **Negrita**, 3 = *cursiva*, 4 = *negrita cursiva*, 5 = símbolos.
- font.axis, font.lab, font.main, font.sub: **fuentes para anotaciones, ejes, título y subtítulo.**
- family: selecciona la familia de fuente a usar: “sans”, “mono”, “serif”,

Margen y tamaño del grafico

- `mai`: vector numérico que indica el tamaño del margen (anajo, izquierda, arriba, derecha) en pulgadas
 - Ejemplo: `c = (1,1,1,1)`
- `pin`: vector para el tamaño del grafico en pulgadas (ancho, largo).

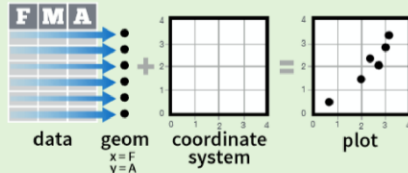
Titulo y líneas de referencia

- `title(main="Titulo", sub="Sub titulo", xlab="Eje X", ylab="Eje Y")`
- `abline(h=Valores en Y, v=Valores en X)`

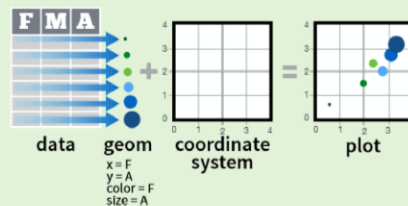
Visualización con ggplot2

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, aes(x = cty, y = hwy))

Begins a plot that you finish by adding layers to. No defaults, but provides more control than **qplot()**.

**ggplot(mpg, aes(hwy, cty)) +
geom_point(aes(color = cyl)) +
geom_smooth(method = "lm") +
coord_cartesian() +
scale_color_gradient() +
theme_bw()**

add layers,
elements with +

layer = geom +
default stat +
layer specific
mappings

additional
elements

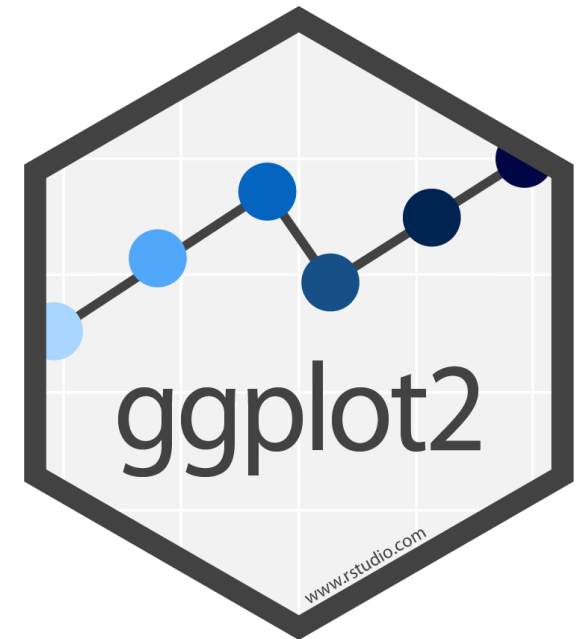
Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()

Returns the last plot

ggsave("plot.png", width = 5, height = 5)

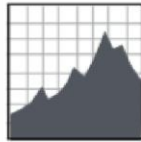
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.



One Variable

Continuous

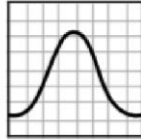
```
a <- ggplot(mpg, aes(hwy))
```



a + geom_area(stat = "bin")

x, y, alpha, color, fill, linetype, size

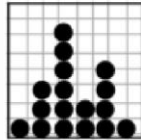
b + geom_area(aes(y = ..density..), stat = "bin")



a + geom_density(kernel = "gaussian")

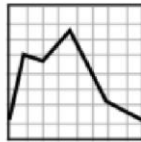
x, y, alpha, color, fill, linetype, size, weight

b + geom_density(aes(y = ..density..))



a + geom_dotplot()

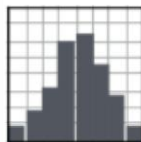
x, y, alpha, color, fill



a + geom_freqpoly()

x, y, alpha, color, linetype, size

b + geom_freqpoly(aes(y = ..density..))



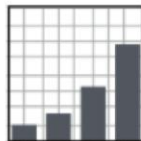
a + geom_histogram(binwidth = 5)

x, y, alpha, color, fill, linetype, size, weight

b + geom_histogram(aes(y = ..density..))

Discrete

```
b <- ggplot(mpg, aes(fl))
```



b + geom_bar()

x, alpha, color, fill, linetype, size, weight

Una Variable

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
```



f + geom_blank()



f + geom_jitter()

x, y, alpha, color, fill, shape, size



f + geom_point()

x, y, alpha, color, fill, shape, size



f + geom_quantile()

x, y, alpha, color, linetype, size, weight



f + geom_rug(sides = "bl")

alpha, color, linetype, size



f + geom_smooth(model = lm)

x, y, alpha, color, fill, linetype, size, weight



f + geom_text(aes(label = cty))

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
g <- ggplot(mpg, aes(class, hwy))
```



g + geom_bar(stat = "identity")

x, y, alpha, color, fill, linetype, size, weight



g + geom_boxplot()

lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight



g + geom_dotplot(binaxis = "y",

stackdir = "center")
x, y, alpha, color, fill



g + geom_violin(scale = "area")

x, y, alpha, color, fill, linetype, size, weight

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
```



j + geom_area()

x, y, alpha, color, fill, linetype, size



j + geom_line()

x, y, alpha, color, linetype, size



j + geom_step(direction = "hv")

x, y, alpha, color, linetype, size

Visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)  
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```



k + geom_crossbar(fatten = 2)

x, y, ymax, ymin, alpha, color, fill, linetype, size



k + geom_errorbar()

x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)



k + geom_linerange()

x, ymin, ymax, alpha, color, linetype, size



k + geom_pointrange()

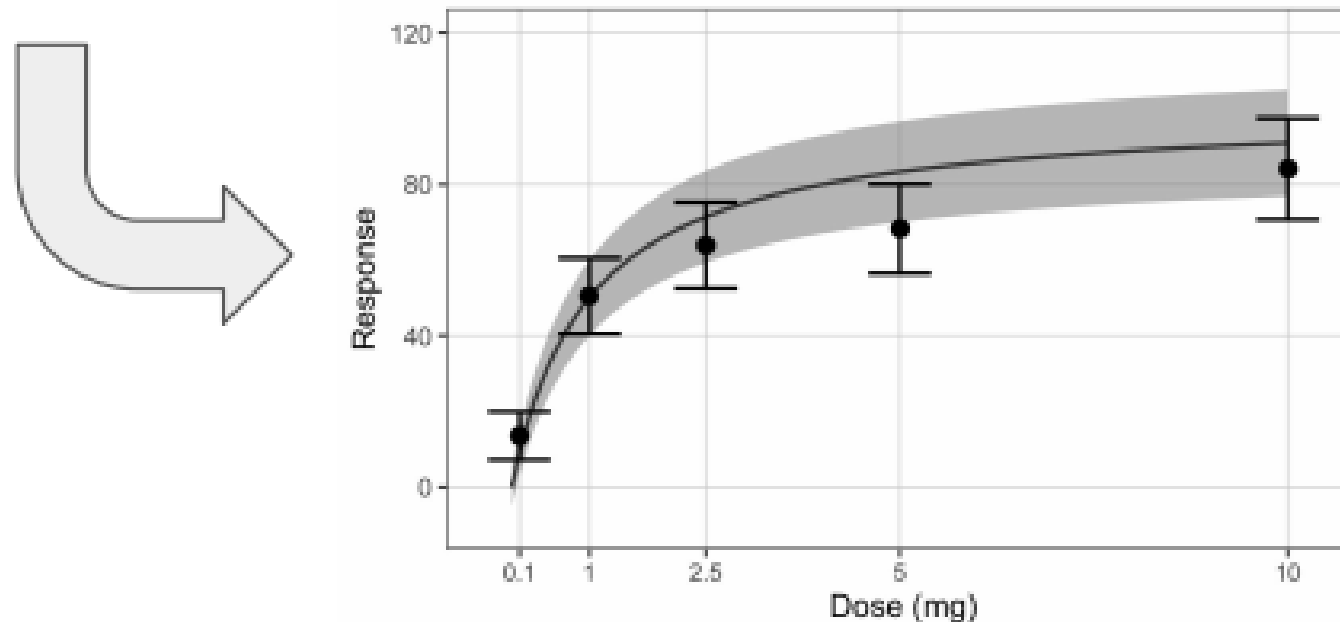
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Dos Variable

```

ggplot(data = my.data, aes(x = Dose, y = Response)) +
  geom_line(size = 1) +
  geom_ribbon(aes(ymin = ymin, ymax = ymax), fill = rgb(0.5,0.5,0.5), alpha = 0.5) +
  geom_point(data = my.data[Dose %in% c(0.1,1,2.5,5,10),],
            aes(x=Dose, y=obs), size = 4) +
  geom_errorbar(data = my.data[Dose %in% c(0.1,1,2.5,5,10),],
              aes(x = Dose, ymin = obs-5-0.1*obs, ymax = obs+5+0.1*obs), size = 1) +
  scale_x_continuous(breaks = c(0.1,1,2.5,5,10), labels = c(0.1,1,2.5,5,10)) +
  xlab("Dose (mg)") +
  ylab("Response") +
  coord_cartesian(ylim = c(-10,120)) +
  theme_bw(base_size = 16) +
  theme(panel.grid.minor=element_blank(),
        panel.grid.major=element_line(color = "lightgrey", size = 0.4),
        legend.position="none",
        axis.text.x=element_text(size = 12)
  )

```



Visualización con plotly

LAYOUT

☰ Legends

```
set.seed( 123 )  
x = 1 : 100  
y1 = 2*x + rnorm( 100 )  
y2 = -2*x + rnorm( 100 )  
  
plot_ly (   
  x = x ,  
  y = y1 ,  
  type = 'scatter' ) %>%  
  
  add_trace(   
    x = x ,  
    y = y2 ) %>%  
  
  layout(   
    legend =   
      list( x = 0.5 ,  
            y = 1 ,  
            bgcolor = '#F3F3F3' ) )
```

✂ Axes

```
set.seed( 123 )  
x = 1 : 100  
y1 = 2*x + rnorm( 100 )  
y2 = -2*x + rnorm( 100 )  
  
axis_template <- list(   
  showgrid = F ,  
  zeroline = F ,  
  nticks = 20 ,  
  showline = T ,  
  title = 'AXIS' ,  
  mirror = 'all' )  
  
plot_ly (   
  x = x ,  
  y = y1 ,  
  type = 'scatter' ) %>%  
  
  layout(   
    xaxis = axis_template ,  
    yaxis = axis_template )
```



BASIC CHARTS

Line Plots

```
plot_ly (  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter',  
  mode = 'lines' )
```

Bubble Charts

```
plot_ly (  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter',  
  mode = 'markers',  
  size = c( 1, 5, 10 ),  
  marker = list(  
    color = c( 'red', 'blue',  
              'green' )))
```

Scatter Plots

```
plot_ly (  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter',  
  mode = 'markers' )
```

Heatmaps

```
plot_ly (  
  z = volcano ,  
  type = 'heatmap' )
```

Bar Charts

```
plot_ly (  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'bar',  
  mode = 'markers' )
```

Area Plots

```
plot_ly (  
  x = c( 1, 2, 3 ),  
  y = c( 5, 6, 7 ),  
  type = 'scatter',  
  mode = 'lines',  
  fill = 'tozeroy' )
```

Gráficos básicos

STATISTICAL CHARTS

Histograms

```
x <- rchisq ( 100, 5, 0 )  
plot_ly (  
  x = x ,  
  type = 'histogram' )
```

Box Plots

```
plot_ly (  
  y = rnorm( 50 ) ,  
  type = 'box' ) %>%  
  
add_trace(y = rnorm( 50, 1 ))
```

2D Histogram

```
plot_ly (  
  x = rnorm( 1000, sd = 10 ) ,  
  y = rnorm( 1000, sd = 5 ) ,  
  type = 'histogram2d' )
```

3D CHARTS

3D Surface Plots

```
# Using a dataframe:  
plot_ly (  
  type = 'surface' ,  
  z = -volcano )
```

3D Line Plots

```
plot_ly (  
  type = 'scatter3d' ,  
  x = c( 9, 8, 5, 1 ) ,  
  y = c( 1, 2, 4, 8 ) ,  
  z = c( 11, 8, 15, 3 ) ,  
  mode = 'lines' )
```

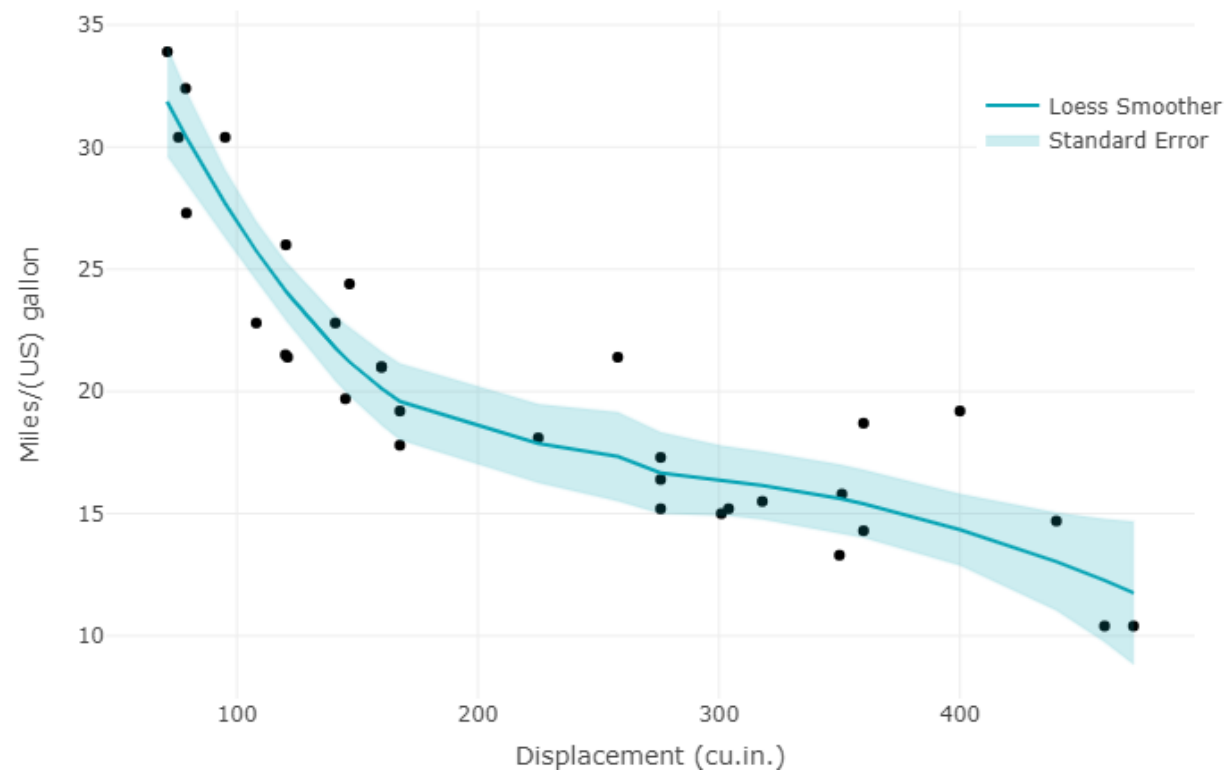
3D Scatter Plots

```
plot_ly (  
  type = 'scatter3d' ,  
  x = c( 9, 8, 5, 1 ) ,  
  y = c( 1, 2, 4, 8 ) ,  
  z = c( 11, 8, 15, 3 ) ,  
  mode = 'markers' )
```

Gráficos Estadísticos

Gráficos 3D

fig



Histograma

- Sirven para obtener una "primera vista" general, o panorama, de la distribución de la población, o de la muestra, respecto a una característica, cuantitativa y continua.

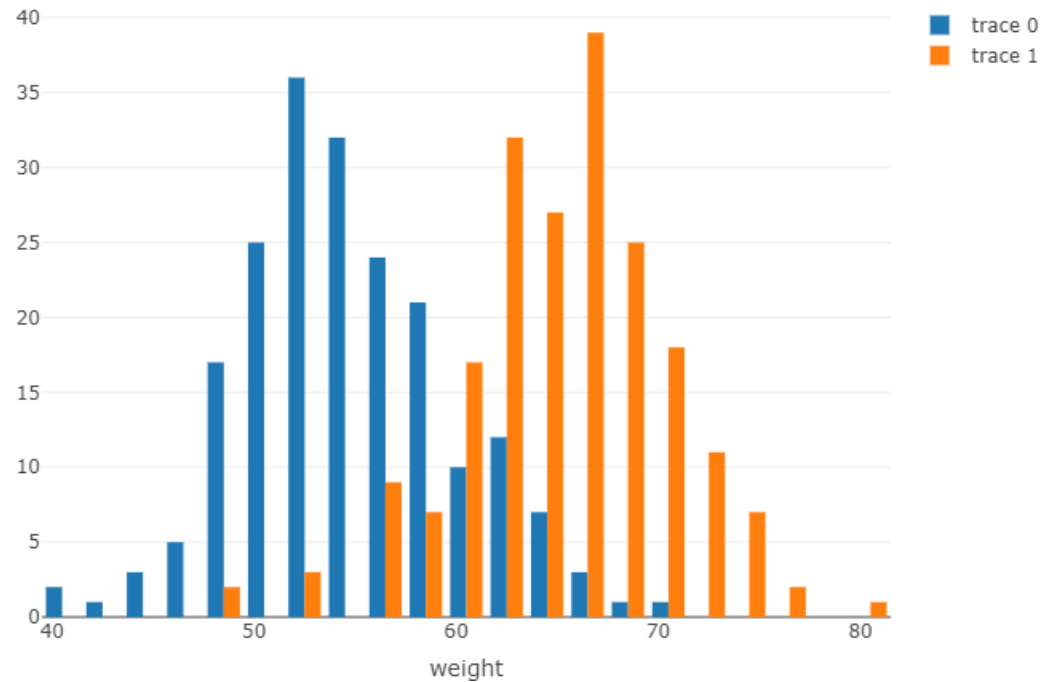


Gráfico de Barras

- Los gráficos de barras son usados para comparar cantidades de valores en diferentes momentos, o también podría decirse productos.

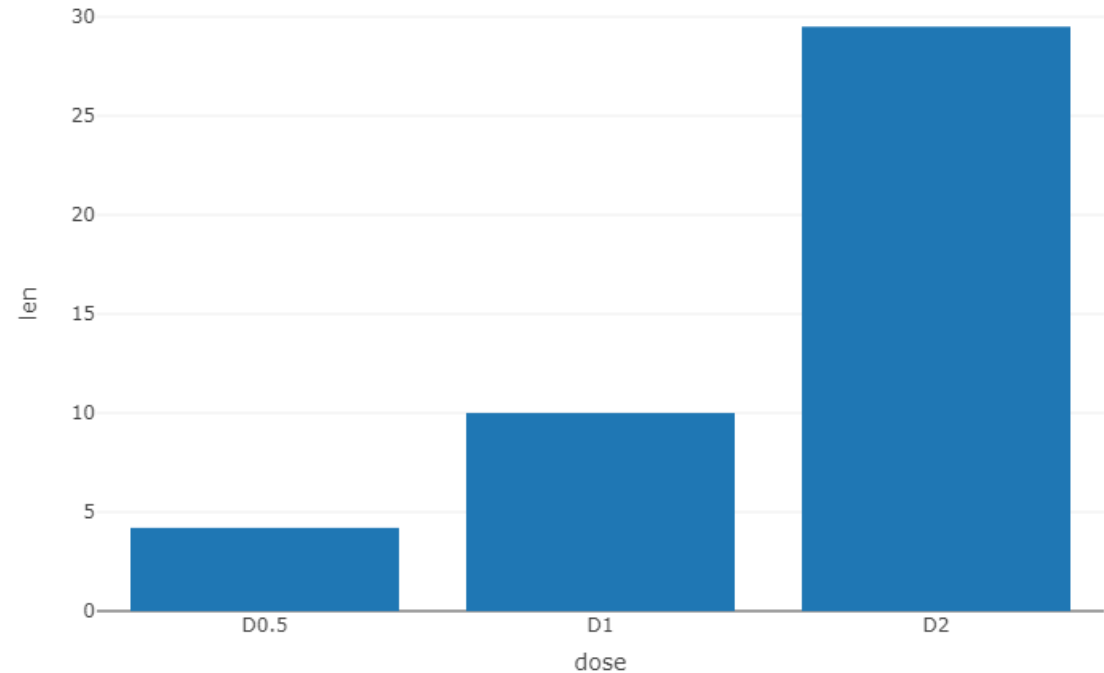


Gráfico de circular

- Se utilizan en aquellos casos donde interesa mostrar el número de veces que se dan una característica o atributo de manera gráfica, de tal manera que se pueda visualizar mejor la proporción en que aparece esa característica respecto del total.

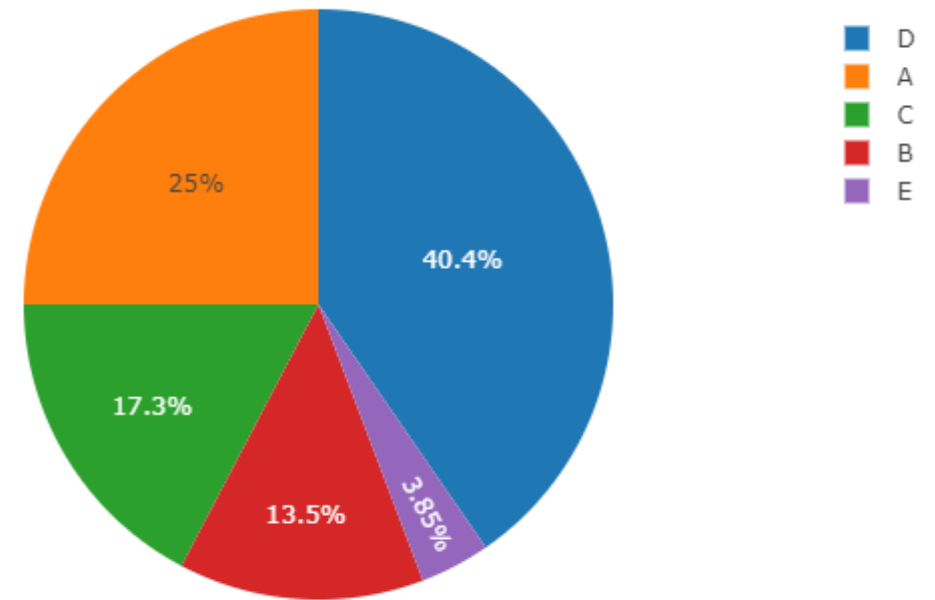


Gráfico de líneas

- Los gráficos de líneas muestran tendencias o cambios a lo largo del tiempo mostrando una serie de puntos de datos conectados por líneas rectas.

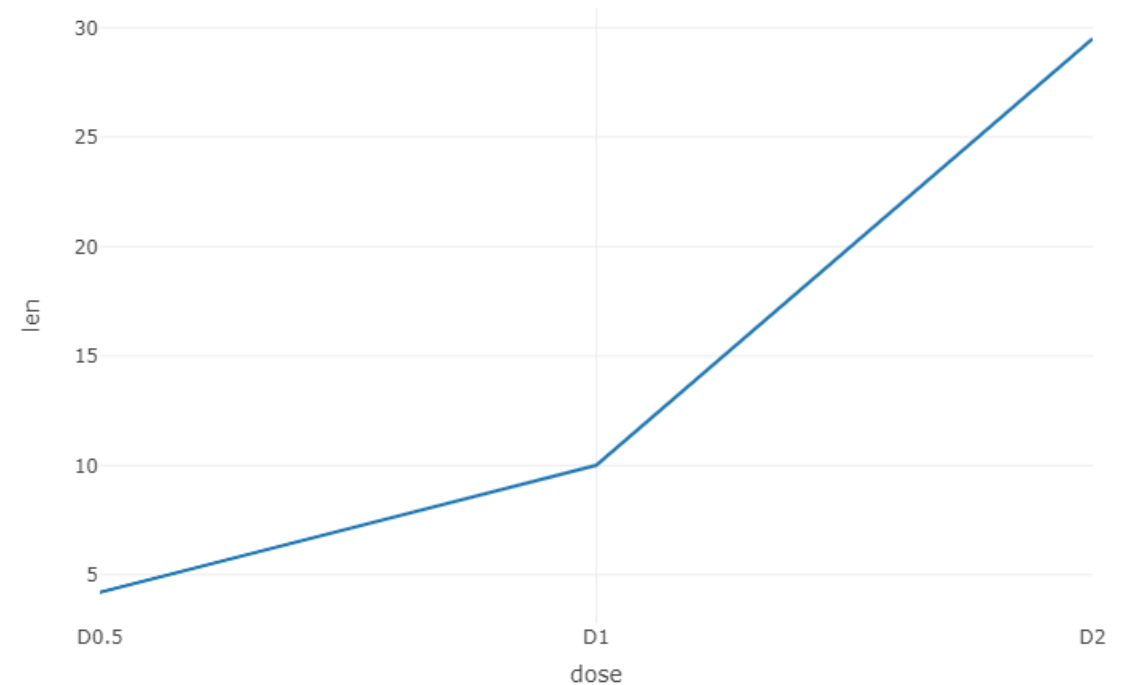
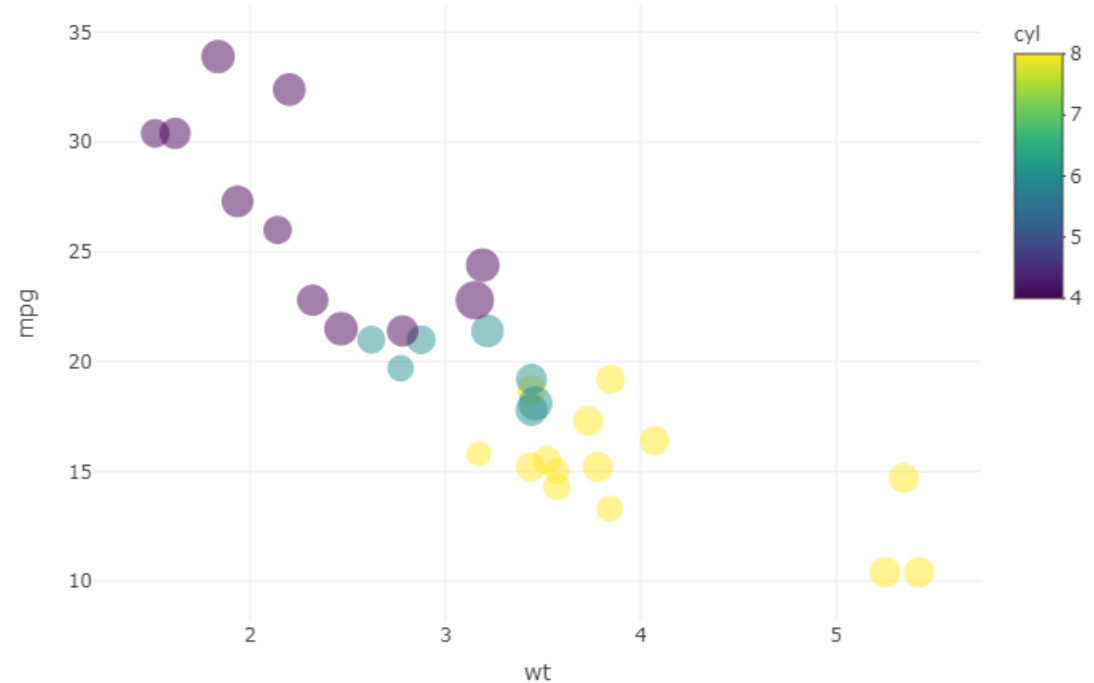


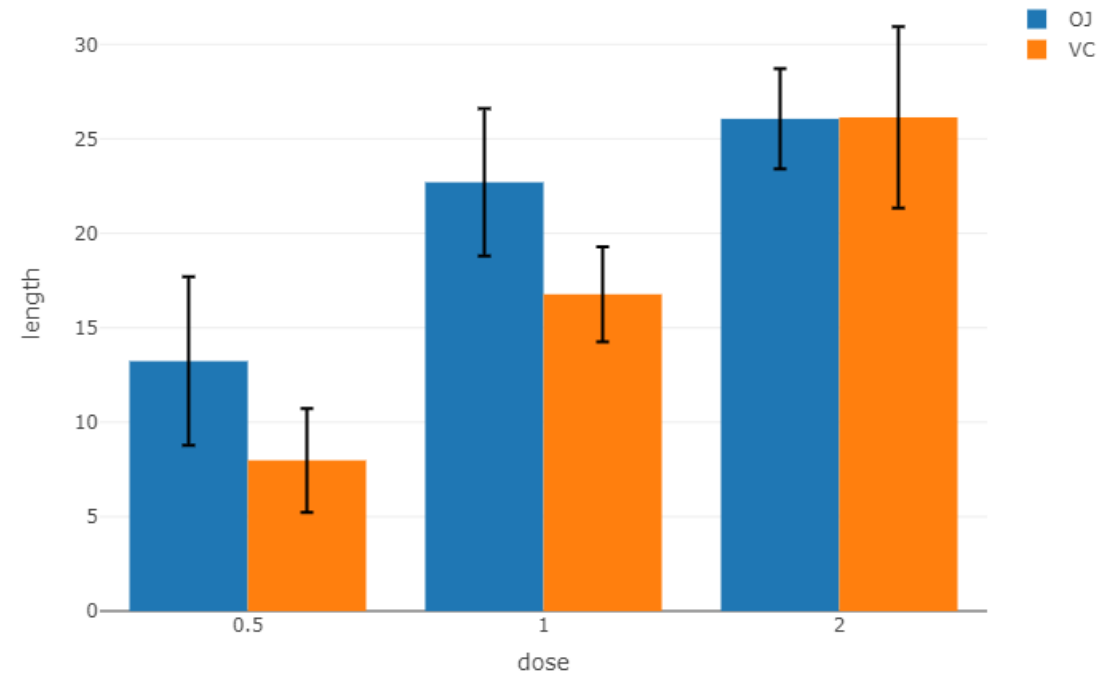
Gráfico de burbujas

- Un gráfico de burbujas es una variación de un gráfico de dispersión en el que los puntos de datos se reemplazan por burbujas y se representa una dimensión adicional de los datos en el tamaño de las burbujas.



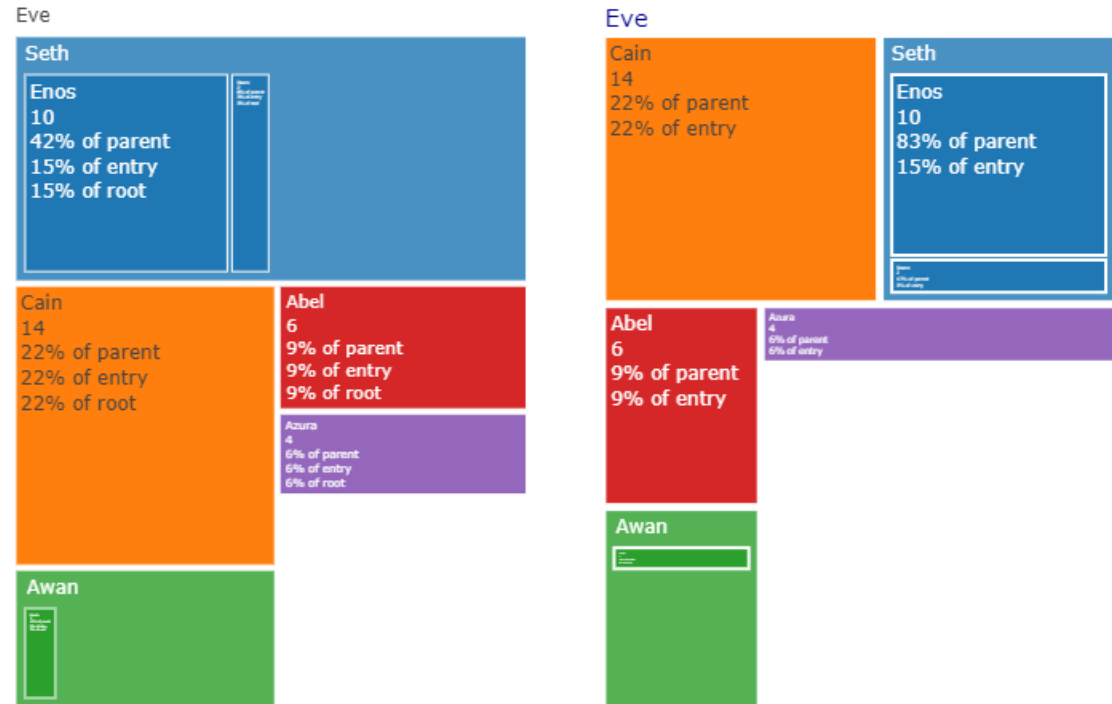
Gráficos de error

- Las barras de error funcionan como una mejora gráfica que visualiza la variabilidad de los datos trazados en un gráfico cartesiano.



Treemap

- Los treemap son perfectos para mostrar gran cantidad de datos de estructura jerárquica (estructura de árbol).



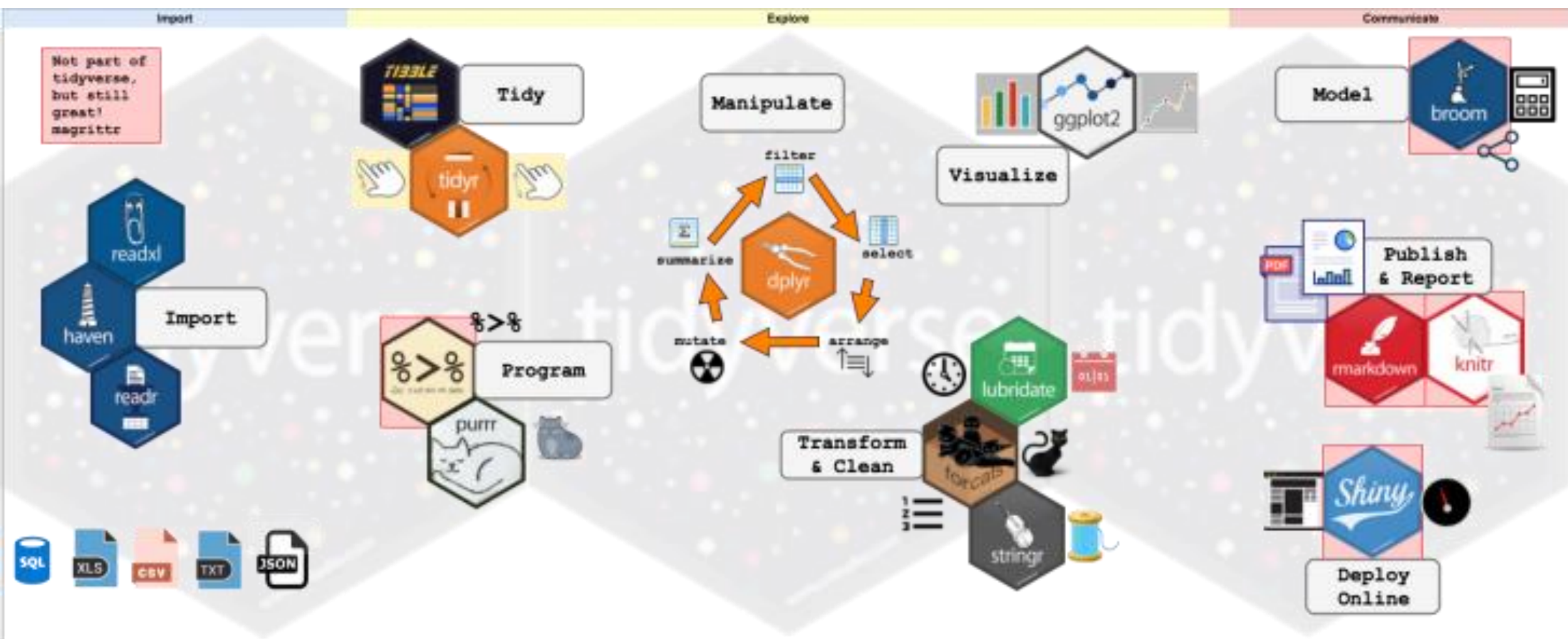
Tips para seguir mejorando en R

1. Tidyverse.
2. STHDA.
3. Stack Overflow.
4. Rpubs.
5. GitHub.
6. Rmarkdown.

Tidyverse

Tidyverse es un conjunto de paquetes implementados por Rstudio, el cual tiene funciones para visualización, manejo de datos, creaciones de tablas, modelamiento, importación de datos, etc.





STHDA

Es un sitio especializado para R donde se puede encontrar, tutoriales, ejemplos, explicaciones y respuestas a problemas que se puedan tener en los Script de R.



Stack Overflow

Es un sitio de preguntas y respuestas para programadores profesionales y aficionados.

Aquí se pueden encontrar soluciones a cualquier tipo de problema que se tenga en R.



stack overflow

R Pubs

Es un sitio, implementado por Rstudio en donde se publican y comparten resultados y proyectos de R.

Aquí se encuentran ejemplos, clases, Scripts, etc.



GitHub

GitHub es un sitio que sirve como repositorio para proyectos de R y otros programas.

Aquí se encuentran proyectos, Scripts, paquetes, etc.



Rmarkdown

Es un paquete para la creación de informes, presentaciones y dashboards.

Se puede automatizar y publicar en R Pubs fácilmente.





Visualización de datos

Cristóbal Honores