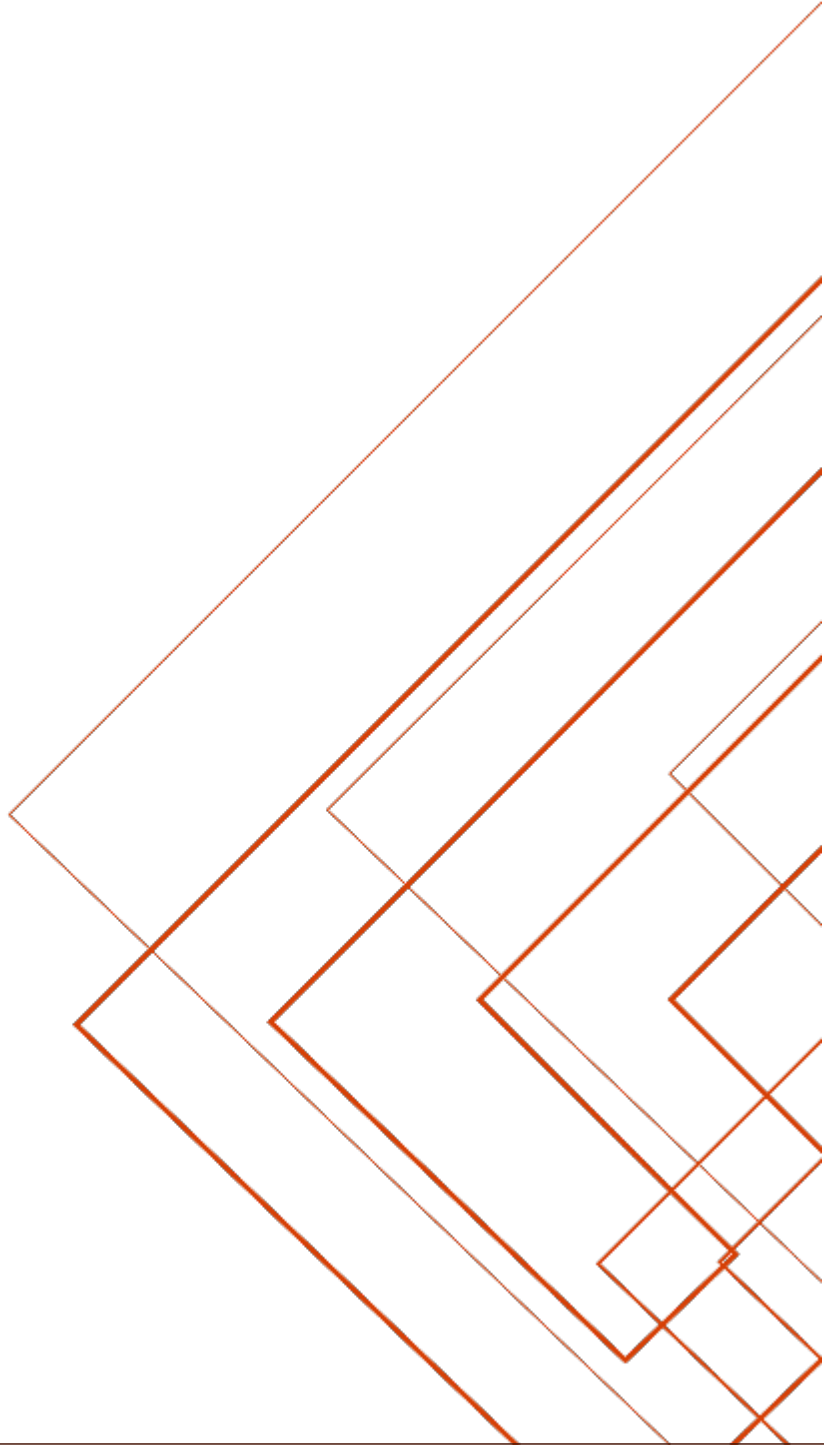
Visualización en R: ggplot2

Máster en Data Sience y Big Data

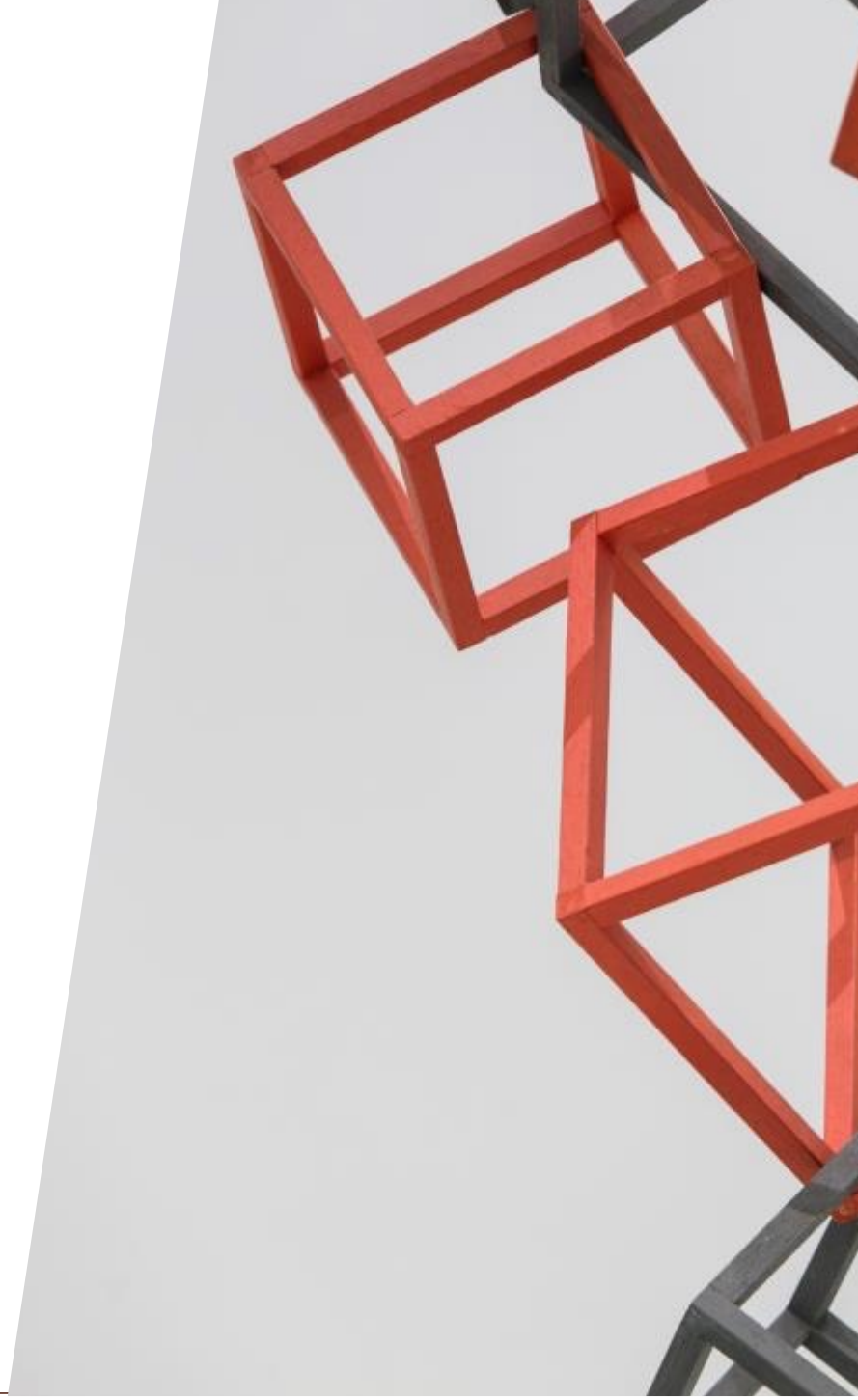
Miguel Ángel Corella
mcorella@geoblink.com

Noviembre 2021

Contenido



Introducción



¿Qué vamos a hacer?

- Vamos a usar un set de datos que contiene información sobre temperaturas diarias de Madrid desde 1995.

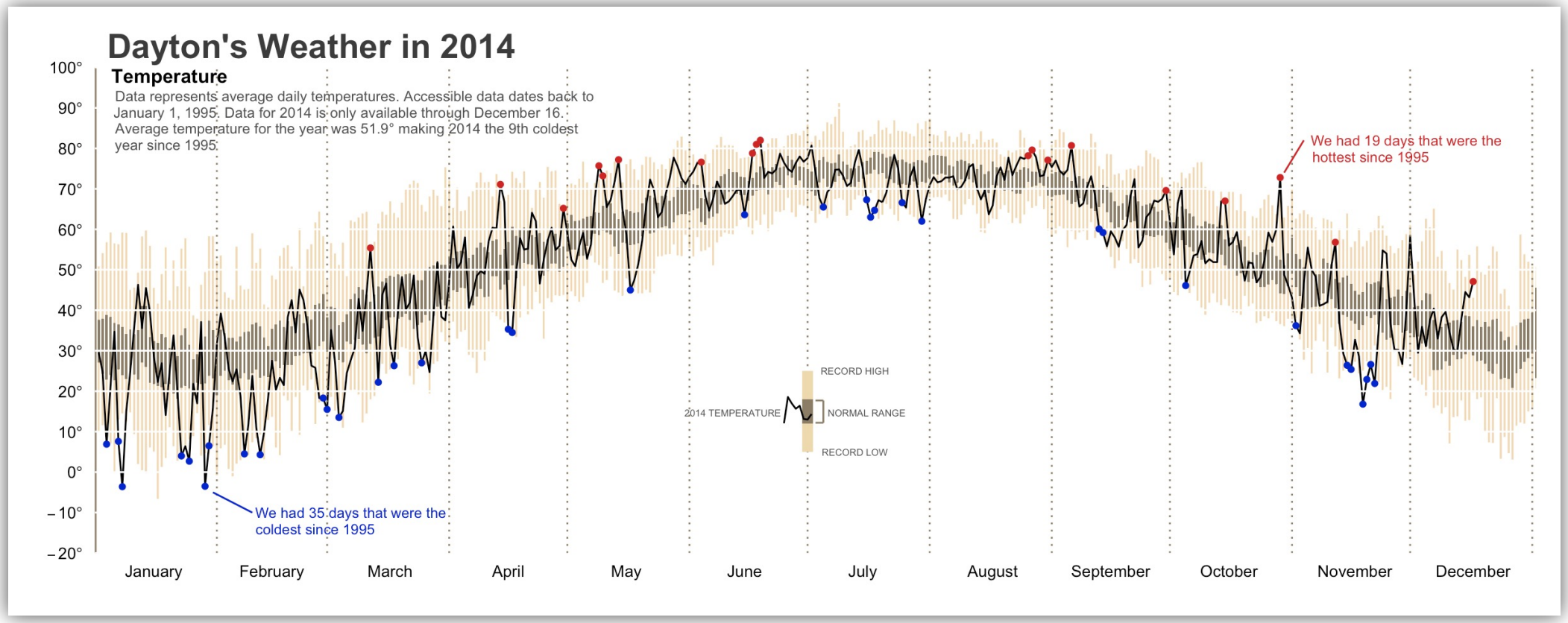
  http://academic.udayton.edu/kissock/http/Weather/

- Usando lo que vayamos aprendiendo de ggplot2 vamos a pasar de esto...

	V1	V2	V3	V4
1	1	1	1995	45.9
2	1	2	1995	40.0
3	1	3	1995	34.1
4	1	4	1995	37.0
5	1	5	1995	43.1
6	1	6	1995	46.7
7	1	7	1995	43.7
8	1	8	1995	40.5
9	1	9	1995	43.0
10	1	10	1995	39.1

¿Qué vamos a hacer?

- A esto...



¿Qué es ggplot2?

- Es un paquete de R que permite la creación de **visualizaciones 2D basadas en datos**.

- Al contrario que otros paquetes de gráficos (p.e. base, lattice...) **no incluye** funciones para la creación de **gráficos "predefinidos"**.

- Entiende los gráficos como **objetos formados por componentes** asociados a diferentes elementos visuales.

- Estos componentes y su interacción están descritos en una **"gramática de gráficos basada en capas"** que establece los elementos básicos de un gráfico y las "normas" que deben seguir.

¿Qué es la "gramática de gráficos"?

- En 1999 (rev. 2005), Leland Wilkinson, publica su trabajo "The grammar of graphics". En él, define una gramática que describe, de un modo formal, todos los elementos y características de los gráficos estadísticos o "basados en datos".

- En 2009, Hadley Wickam, publica "A layered grammar of graphics". Una revisión del trabajo de Wilkinson, que organiza los componentes de su gramática en capas, que se pueden superponer, lo que permite llevar a cabo la creación de cualquier gráfico estadístico de forma iterativa.

- Los conceptos definidos en la revisión realizada por Hadley Wickam son la base de la implementación de la gramática en R, es decir, del paquete ggplot2.

Componentes de la gramática

Componente	Descripción	Obligatorio	Ejemplos
Data	Set de datos en el que se basa el gráfico	●	iris, diamonds
Aesthetics	Características visuales asociadas a los datos	●	x, y, colour, fill, alpha
Geometries	Elementos visuales que representarán los datos	●	point, bar, line, jitter, boxplot
Statistics	Agregaciones/resúmenes de los datos	○	smooth, ablilne, errorbar
Facets	Desagregación de los datos en múltiples gráficos	○	grid, wrap
Coordinates	El espacio donde el gráfico va a ser presentado	○	cartesian, polar
Themes	Elementos visuales no relacionados con datos	○	axis.line, legend.background, plot.title

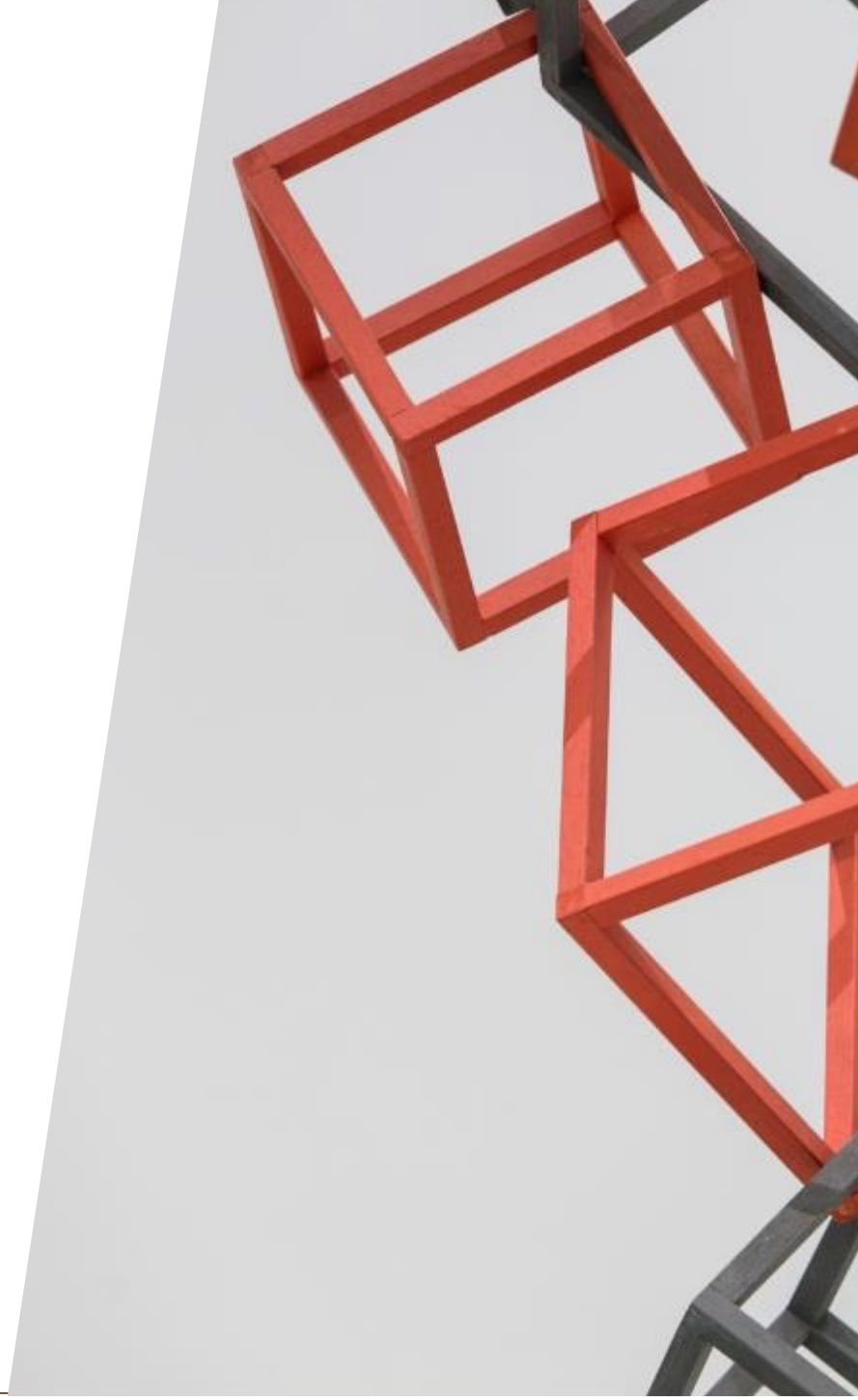
¿Qué no hace la gramática?

- **No define qué gráfico se debe usar** para la realización de un análisis concreto.
  - La gramática da las herramientas necesarias para crear "cualquier" tipo de gráfico.
  - La elección del tipo de gráfico correcto para el objetivo perseguido es responsabilidad del usuario.

- **No define qué estilo se debe seguir** en un gráfico.
  - Aunque ggplot2 ofrece algunos "temas" básicos, la gramática no define estilos "correctos" ni predefinidos.
  - Conseguir que un gráfico sea o no atractivo (en función de su objetivo) es responsabilidad del usuario.

- **No define ningún tipo de interacción** con los gráficos.
  - La gramática incluye definición únicamente de componentes estáticos y, por lo tanto, la interacción con un gráfico queda fuera del alcance de la misma.

Base vs. ggplot2



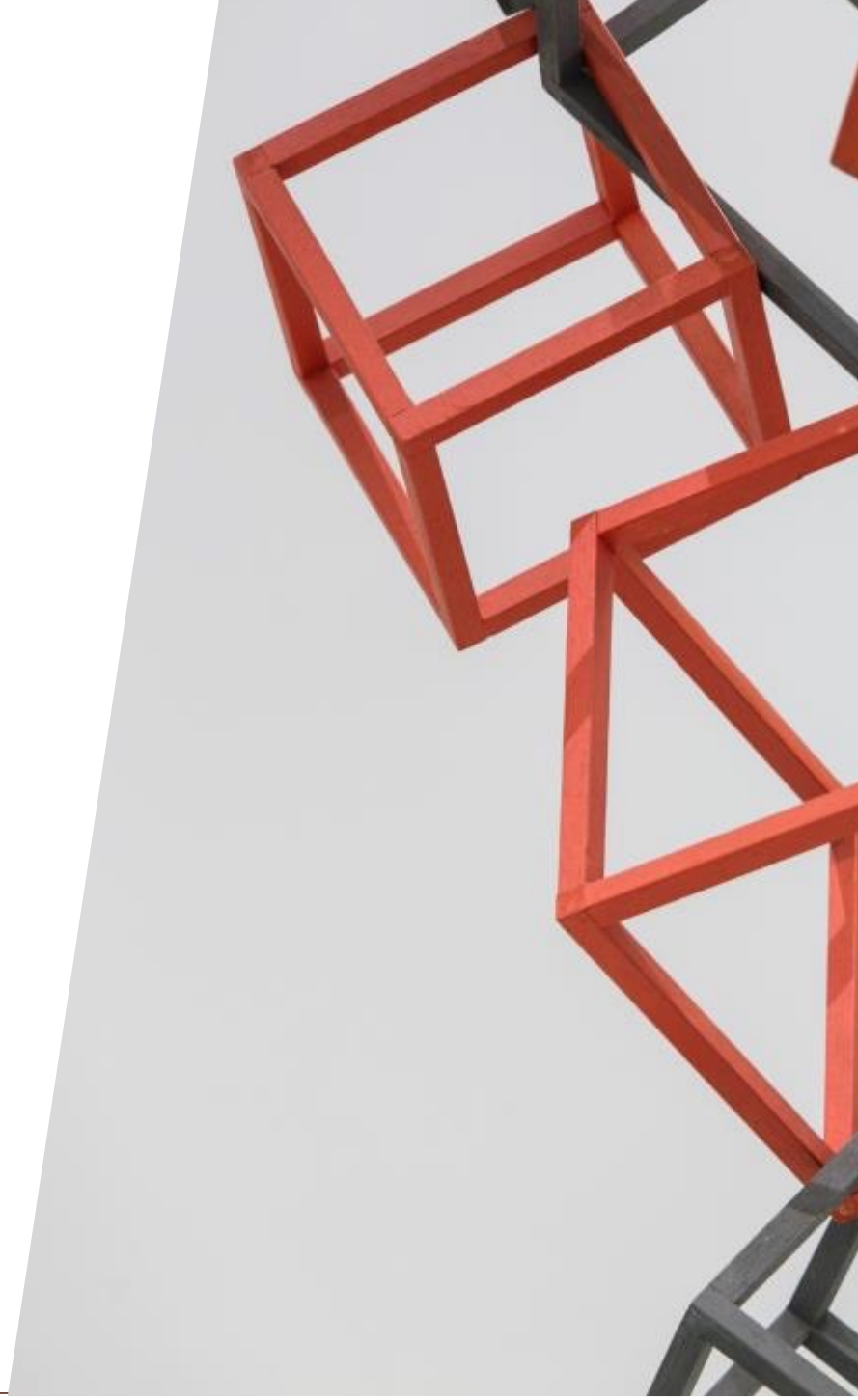
Comparativa de paquetes

Script 01-base-vs-ggplot2.R

Desventajas de base vs. ggplot2

- Tratamiento del gráfico como una pura imagen "estática" creada en la primera llamada.
- No realiza recálculos a la hora de añadir "anotaciones" o elementos adicionales al primer gráfico.
- No incluye automáticamente elementos adicionales basados en datos (p.e. leyendas) sino que hay que generarlos de forma manual.
- No define un framework común para la creación de gráficos, sino que da un conjunto de funciones "predefinidas".
- En resumen, dificulta la creación de gráficos de forma iterativa y no ofrece una base común para la generación de visualizaciones.

Funciones básicas de ggplot2



Funciones básicas de ggplot2

- **qplot** (quick plot):
  - Todos los conceptos de la gramática en una única llamada con múltiples parámetros.
  - Filosofía de plot de R base por la que se debe saber, a priori, el resultado final deseado.
  - Aporta sencillez de uso, pero sacrifica versatilidad.

```
qplot(x, y, ..., data, facets, margins, geom, xlim, ylim,
      log, main, xlab, ylab, asp, stat, position)
```

- **ggplot** (grammar of graphics plot):
  - Crea un objeto de gráficos base sobre el que ir añadiendo componentes.
  - La creación iterativa del gráfico facilita la obtención paso a paso del gráfico objetivo.
  - Su uso es algo más complejo pero da acceso a toda la potencia de la librería.
  - Su implementación refleja claramente todos los componentes de la gramática.

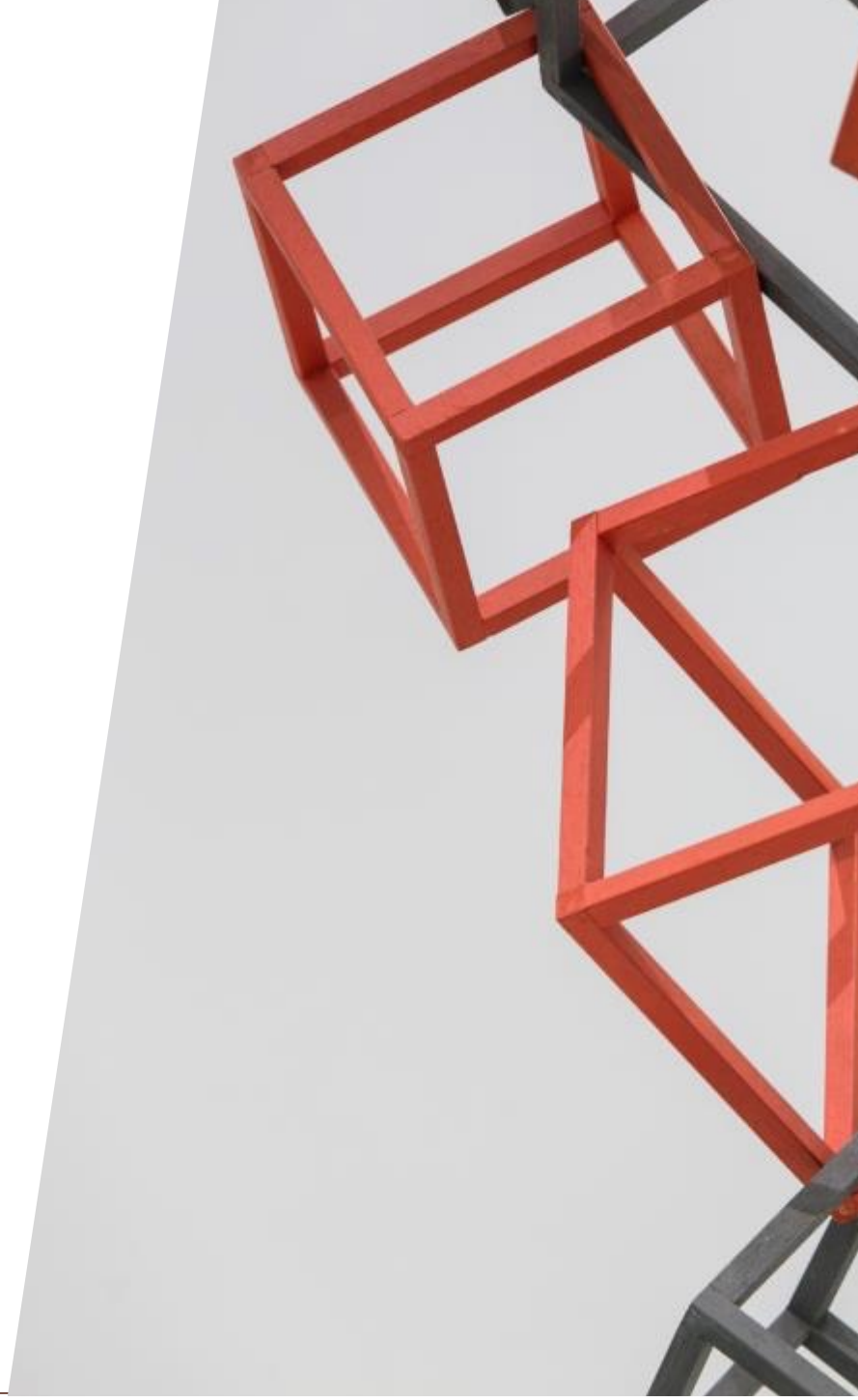
```
ggplot(data, mapping=aes())
```

Funciones básicas de ggplot2

Script 02-qplot-vs-ggplot.R

Componentes de ggplot2

Data



Componentes de ggplot2

Data: Datos subyacentes al gráfico

- El primer componente de cualquier gráfico en ggplot2 (y cualquier otro paquete) es, obviamente, el set de datos sobre el que se plantea la visualización.

- ggplot2 establece una única restricción sobre los datos: **TIENE que ser un data.frame**.

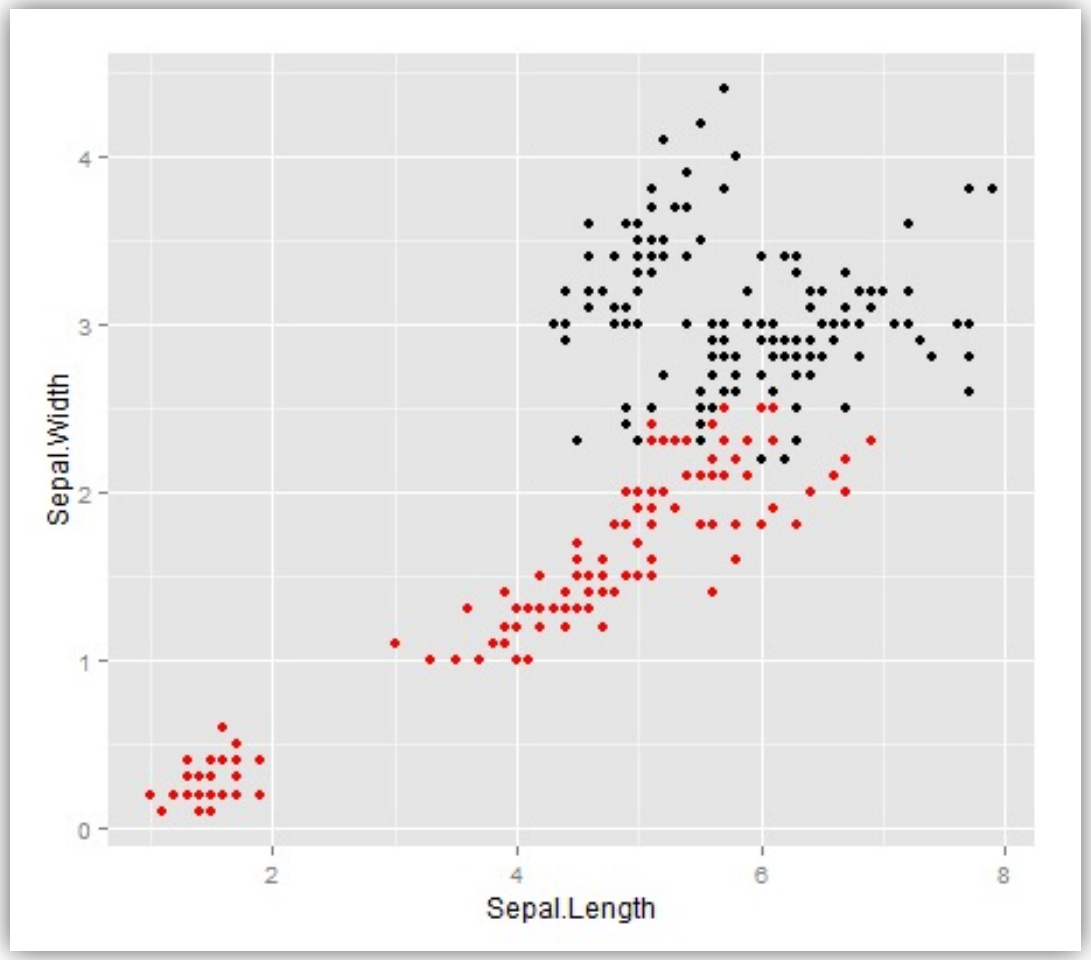
- Para definir el set de datos en nuestro gráfico:

```
ggplot(data = variable, ...)
```

- Aunque podemos usar cualquier data.frame como fuente de datos, conviene que éste esté correctamente formateado para facilitar su uso.

- Pero… ¿qué es un data.frame correctamente formateado?

Volvamos al ejemplo de antes...



¿Qué variables se están comparando en el gráfico?

¿Que significan los puntos de distintos colores?

¿Qué estábamos tratando de representar con el gráfico?

Datos para análisis vs. datos para visualización

- Un formato válido para determinadas tareas (p.e. análisis estadístico, algoritmos de machine learning, etc.) no tiene por qué ser el más adecuado para las tareas de visualización.

- Aún más, un set de datos formateado para dar soporte a una visualización, no tiene por qué ser el más adecuado para llevar a cabo una visualización distinta.

- **SIEMPRE CONVIENE ADECUAR EL SET DE DATOS PARA FACILITAR SU VISUALIZACIÓN**
  - **Contenido:** modificar la información contenida para disponer de resúmenes, agregaciones, agrupaciones...
  - **Estructura:** información incluida en fila, información incluida en columnas, formato y tipo de las variables incluidas...

Wide format vs. long/tidy format

Wide format

- Cada fila del set de datos es un sujeto.
- Cada columa es una variable observada y su valor.
- Puede ocurrir que una columna contenga "valores" en sus cabeceras (p.e. una columna por año en una serie temporal)
- Puede ocurrir que una columa contenga más de una variable/valor (p.e. una columna por trimestre en una serie temporal)

	Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	setosa	5.1	3.5	1.4	0.2
2	setosa	4.9	3.0	1.4	0.2
3	setosa	4.7	3.2	1.3	0.2
4	setosa	4.6	3.1	1.5	0.2
5	setosa	5.0	3.6	1.4	0.2
6	setosa	5.4	3.9	1.7	0.4

Wide format vs. long/tidy format

Long/Tidy format

- Cada fila del set de datos es una observación.
- Existen columnas distintas para describir el tipo de observación realizada y para almacenar su valor.
- La información asociada a un mismo sujeto estará distribuida entre múltiples filas.
- **Generalmente**, este formato facilita la creación de gráficos con ggplot2.

	Species	Part	Measure	Value
1	setosa	Sepal	Length	5.1
2	setosa	Sepal	Length	4.9
3	setosa	Sepal	Length	4.7
4	setosa	Sepal	Length	4.6
5	setosa	Sepal	Length	5.0
6	setosa	Sepal	Length	5.4

Manipulación de datos en R

- Manipulación de **contenido**:
  - lubridate.
  - stringr.
  - plyr.
  - dplyr.
  - data.table.

- Manipulación de **estructura**:
  - reshape2.
  - tidyr.

https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf

Juguemos con Data...

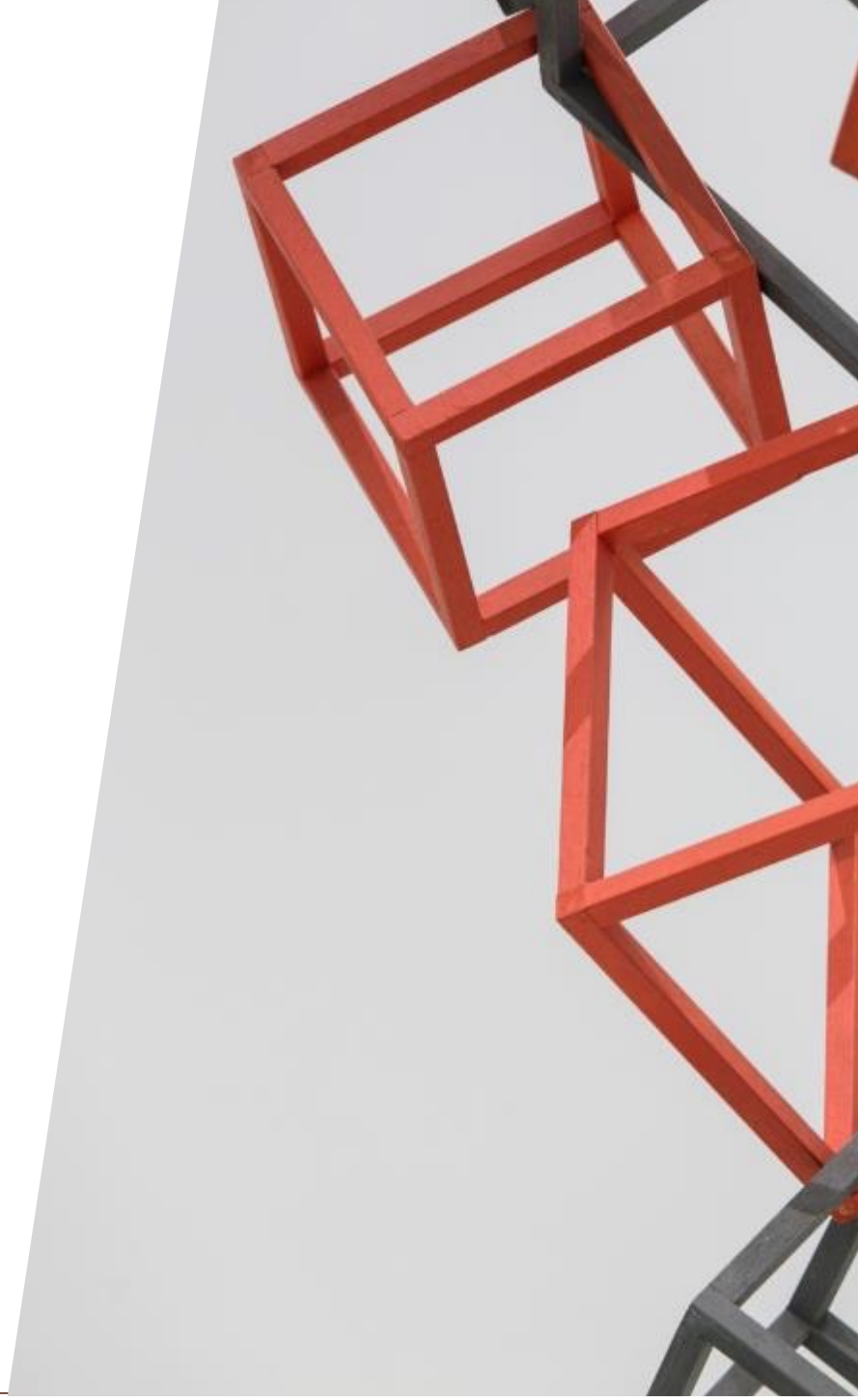
Script 03-data.R

Ejercicio 1: Data

Script E01-data.R

Componentes de ggplot2

Aesthetics



Componentes de ggplot2

Aesthetics: Características visuales

- A la hora de representar datos en un gráfico, los elementos utilizados para representarlos tienen asociadas una serie de características visuales o **aesthetics**.

- Algunos ejemplos de aesthetics (aunque no se listan todos) son:
  - x, y: Posición en el eje X y posición en el eje Y del elemento.
  - col: Color del elemento (en puntos y líneas) o color del borde (en polígonos).
  - fill: Color de relleno del elemento.
  - alpha: Transparencia del elemento.
  - shape: Forma del marcador utilizado.
  - size: Tamaño del elemento.
  - ...

- Una de las principales potencias de ggplot2 es la flexibilidad que da al usuario para definir la asignación de valores a estas características visuales.

Definición de Aesthetics / Attributes

- ggplot2 permite llevar a cabo la asignación de valores a características visuales relacionadas con datos de dos formas muy diferenciadas:
  - Mediante **asignación directa** de un valor estático: **Attribute**
  - Mediante **mapeo a una variable** contenida en el set de datos de gráfico: **Aesthetic**

- **ggplot** diferencia entre aesthetic y attribute en función de dónde se incluya la asignación.

```
ggplot(..., aes(alpha=variable)) / geom_XXX(..., alpha=0.4)
```

Aesthetics disponibles en ggplot2

- El conjunto de **aesthetics disponibles depende**, siempre, **del tipo de elemento gráfico** a utilizar para la representación de los datos, es decir, del componente Geometries.
  - En un gráfico de puntos (scatter) no tiene sentido definir el tipo de linea.
  - En un gráfico de barras no tiene sentido definir la forma del punto.
  - ...

- En la ayuda de las funciones específicas del componente Geometries, se puede encontrar el conjunto de aesthetics disponibles para cada una de ellas, así como la indicación de cuáles son obligatorias.

- O mucho más sencillo y mucho más útil...

https://github.com/rstudio/cheatsheets/blob/main/data-visualization-2.1.pdf

Juguemos con Aesthetics...

Script 04-aesthetics.R

Componentes de ggplot2

Scales: Mapeo de datos a aesthetics

- La asignación de valores específicos desde una variable del set de datos a una aesthetic se realiza a través de un elemento denominado **scale.**

- Un scale **define el rango de valores posibles de una aesthetic** y **establece la equivalencia con los valores de la variable del set de datos** asignada.

- Todas las aesthetics de ggplot2 definen una escala por defecto.

- Generalmente, existen dos tipos de escala, en función del tipo de variable asignada a la aesthetic:
  - Discretas: Para variables de tipo character y/o factores.
  - Continuas: Para variables numéricas.

- Pero ggplot2 nos da control absoluto sobre cómo realizar este mapeo de variables a scales.

Definición “manual” de Scales

- Para modificar el scale aplicado a un aesthetic, basta con añadir (+) a la función ggplot una llamada del tipo:

```
ggplot(...) + scale_XXX_YYY(...)
```

  - Donde **XXX** será el nombre del aesthetic a modificar.
  - Donde **YYY** será el tipo de escala que se desea aplicar.
  - Donde **(…)** serán los parámetros específicos requeridos por cada función scale.

- Se pueden agrupar las funciones scale en función del aesthetic al que se apliquen:
  - De propósito general: Para cualquier aesthetic.
  - De posición: Para los aesthetics “x” e “y”.
  - De color y relleno: Para los aesthetics “col” y “fill”.
  - De forma: Para el aesthetic “shape”.
  - De tamaño: Para el aesthetic “size”.

Juguemos con Scales...

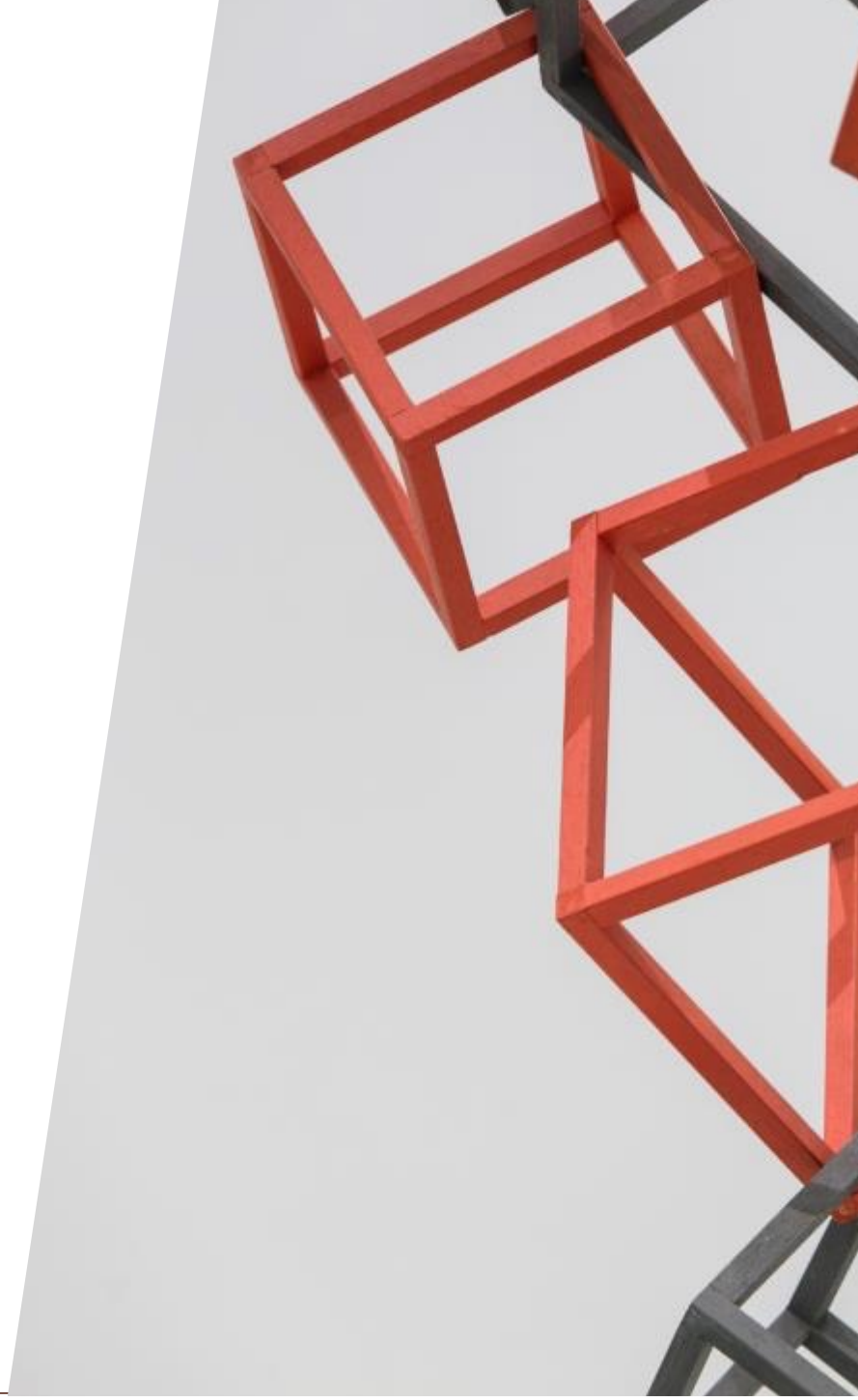
Script 05-scales.R

Ejercicio 2: Aesthetics y Scales

Script E02-aesthetics-scales.R

Componentes de ggplot2

Geometries



Componentes de ggplot2

Geometries: Elementos visuales

- Los **geometries** (o geoms) definen la forma o figura geométrica que se va a utilizar para representar cada observación de nuestros datos en el gráfico.

- Representan, por tanto, el tipo de gráfico que se desea utilizar:
  - Puntos (o scatter plot).
  - Barras.
  - Línea.
  - Texto.
  - …

- Los geoms únicamente establecen la **forma** que representarán las observaciones de nuestros datos, pero **no tiene una asociación directa con los datos.**

- Esta independencia de los datos permite modificar el tipo de gráfico de una forma rápida, directa, lo que facilita enormemente la creación iterativa de gráficos.

Definición de Geometries

- Como se ha visto anteriormente, es obligatorio definir al menos un geom para nuestro gráfico (aunque podemos definir tantos como queramos).

```
ggplot(...) + geom_XXX(...) + geom_YYY(...)
```

  - Donde **XXX** e **YYY** serán el tipo de geom a usar.
  - Donde **(…)** serán los parámetros específicos de cada geom a utilizar.

- El sistema de herencia establecido por ggplot2 hace que **las funciones geom utilicen**, por defecto, el **data y los aesthetics definidos en la llamada a la función ggplot**.

- Sin embargo, se pueden **redefinir estos dos parámetros** en cada geom (p.e. sobreescritura de aesthetics, múltiples set de datos en un gráfico…)

Geoms (y aesthetics) disponibles en ggplot2

- Como hemos comentado anteriormente, aunque aesthetics y geoms se definen por separado, hay que tener en cuenta que **no todos los aesthetics están disponibles (ni tienen sentido) para todos los geoms**.
  - En un gráfico de puntos (scatter) no tiene sentido definir el tipo de linea.
  - En un gráfico de barras no tiene sentido definir la posición en y.
  - En un gráfico de líneas no tiene sentido definir el color de relleno.
  - …

- De nuevo, se puede consultar el conjunto de aesthetics disponibles para un geom determinado (y cuales de ellos son obligatorios) en la ayuda de cada función geom.

- O, de nuevo, mucho más fácil y sencillo…

http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Juguemos con Geometries...

Script 06-geoms.R

Componentes de ggplot2

Positions: Posicionamiento de geoms

- La posición de los geoms viene dada, generalmente, por los propios datos subyacentes y su mapeo a los aesthetics x e y.

- Sin embargo, en ocasiones esto puede hacer que múltiples observaciones queden representadas gráficamente en la misma ubicación, dificultando así la interpretación de los resultados.
  - Gráficos de barras "agrupados": ¿Cómo posicionar los diferentes grupos?
  - Gráficos de puntos: ¿Qué hacer si múltiples puntos coinciden en x e y?
  - …

- Esta situación problemática se conoce como **overplotting.**

- ggplot2 nos permite solucionar este problema mediante la definición de **positions** que modifican la ubicación "por defecto" de los geoms.

Definición “manual” de Positions

- Para modificar la ubicación de un geom, basta con utilizar el parámetro **position** de la función geom que estemos utilizando:

```
geom_XXX(..., position="YYY")
```

  - Donde **YYY** será el nombre del tipo de posición a aplicar.

- Existen cinco tipos de posicionamiento:
  - **identity:** Es el valor por defecto.
  - **jitter:** Introduce un pequeño ruido en x e y para evitar el overplotting.
  - **dodge:** Ordena los elementos uno al lado del otro.
  - **stack:** Ordena los elementos uno encima del otro.
  - **fill:** Ordena los elementos uno encima del otro y normaliza sus valores.

Otra definición “manual” de Positions

- Además de definir un position de forma manual a través de su nombre, ggplot2 nos permite también modificar ligeramente el funcionamiento de algunos tipos:

```
geom_XXX(..., position=position_YYY(...))
```

  - Donde **YYY** será el tipo de posición a aplicar.
  - Donde **(…)** contendrá los parámetros específicos a utilizar en cada posición.

- Solo los tipos **dodge** y **jitter** ofrecen parámetros que modificar (aunque existen las funciones para el resto, los parámetros son ignorados):
  - **jitter:** width y height para establecer el porcentaje de ruido a introducir.
  - **dodge:** width para establecer la separación entre elementos.

Juguemos con Positions...

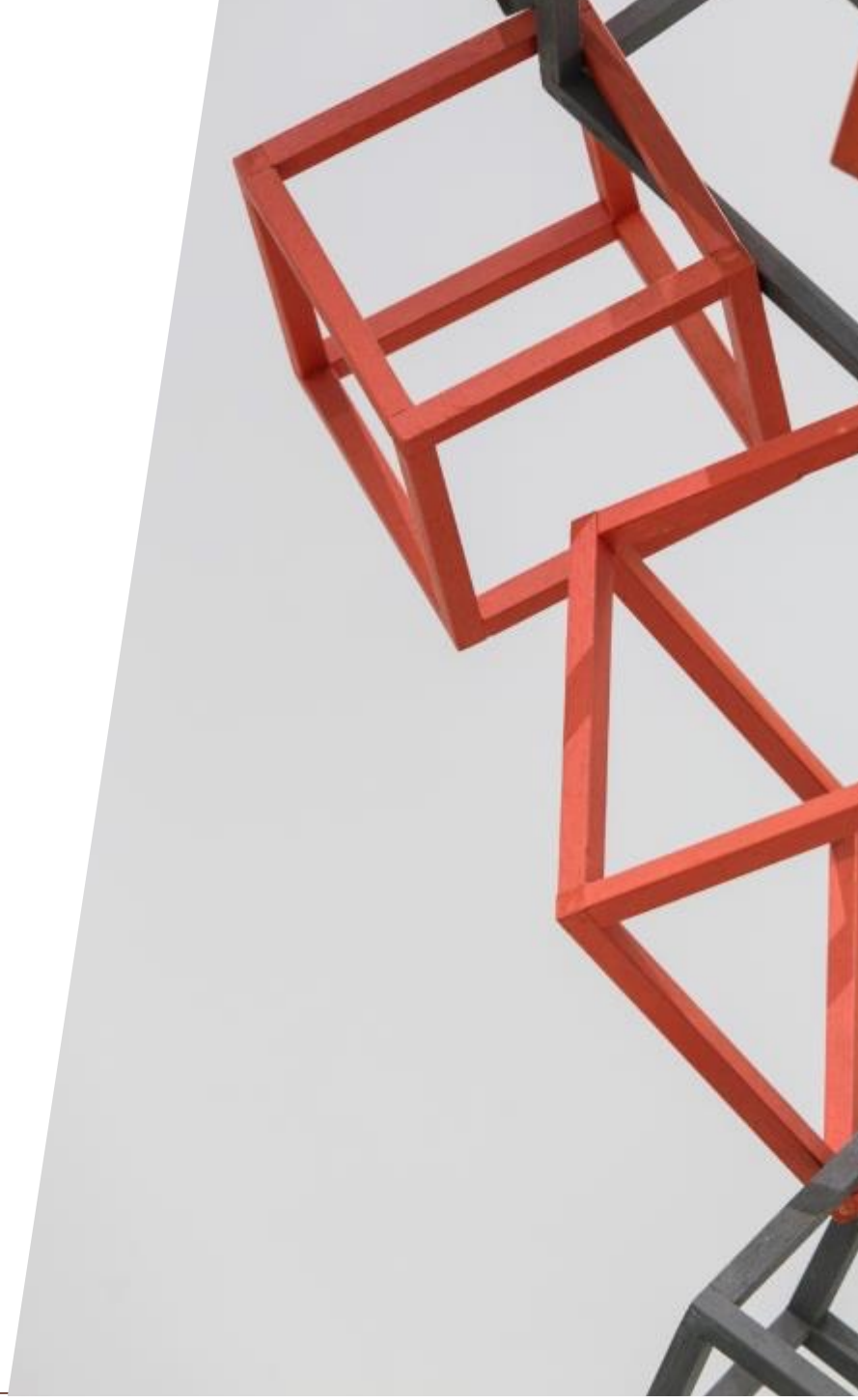
Script 07-positions.R

Ejercicio 3: Geometries y Positions

Script E03-geoms-positions.R

Componentes de ggplot2

Statistics



Componentes de ggplot2

Statistics: Agrupación/Resumen de datos

- Los **statistics** (o stats) definen diferentes funciones de agregación / resúmen que se pueden aplicar sobre las variables del set de datos utilizado en un gráfico.

- Cada stat realiza una serie de cálculos sobre los datos utilizados en el gráfico y **genera nuevas variables** que pone a disposición (generalmente de forma automática) para ser mapeadas a aesthetics.

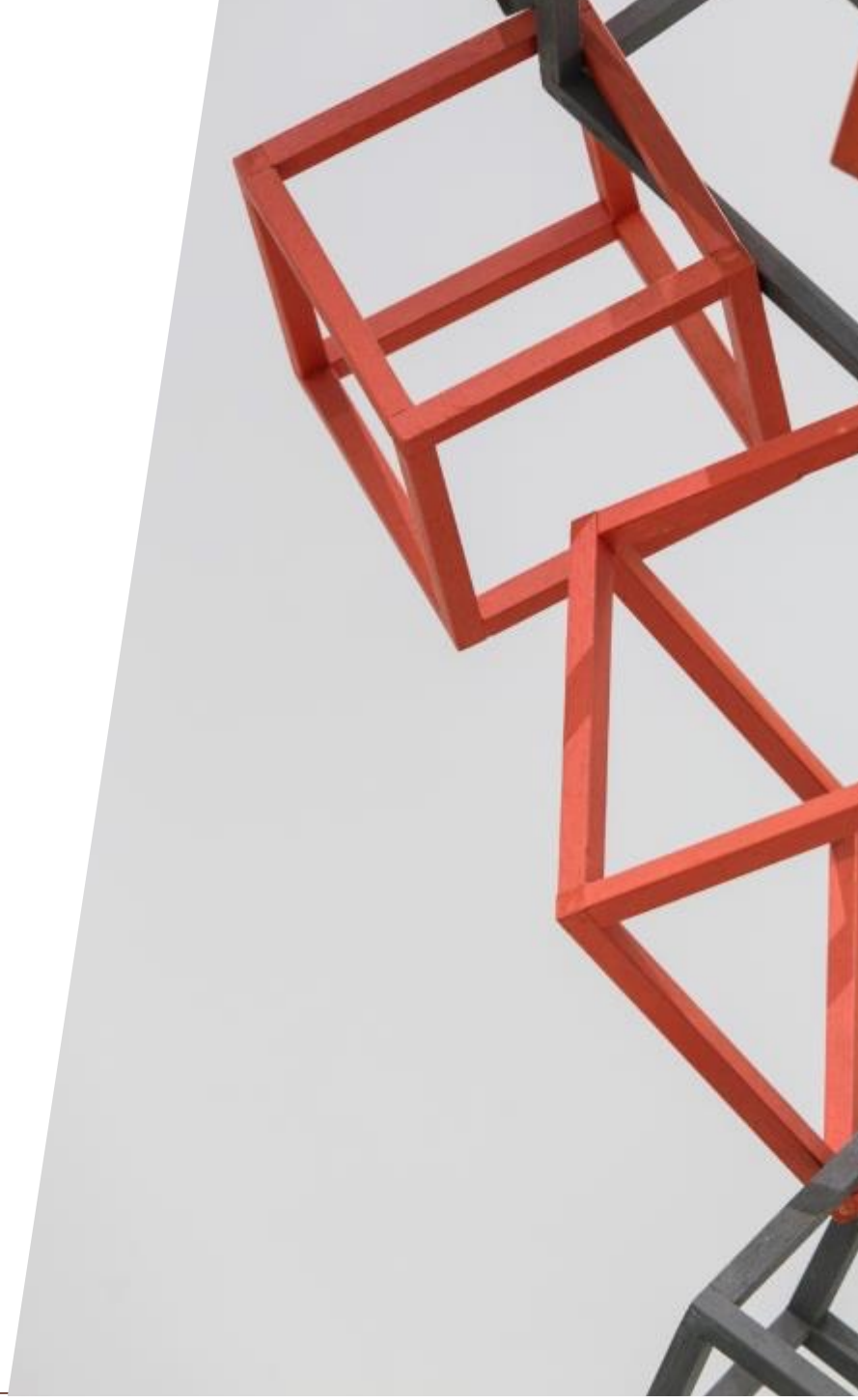
- Algunos ejemplos de stats disponibles en ggplot2 podrían ser:
  - Suavizados.
  - Regresiones.
  - Cuantiles.
  - Discretización.
  - Distribuciones.
  - ...

Juguemos con Statistics...

Script 08-stats.R

Componentes de ggplot2

Facets



Componentes de ggplot2

Facets: Múltiples gráficos

- Los **facets** permiten desagregar un gráfico en múltiples subgráficos en función de una variable (o varias) del set de datos utilizado.

- Los subgráficos generados **comparten aesthetics (y escalas)**, **geoms (y positions) y stats**, de forma que se asegura que sean comparables entre sí, facilitando por tanto el análisis de set de datos amplios.

- ggplot2 ofrece dos formas bien diferenciadas de hacer esta desagregación:
  - Desagregación en un eje sobre una variable: **wrap**.
  - Desagregación en dos ejes sobre una variable (o varias): **grid**.

Definición de Facets

- Como en el resto de componentes, ggplot2 nos ofrece un nuevo conjunto de funciones para la definición de facets a utilizar en nuestros gráficos:

```
ggplot(...) + geom_ZZZ(...) + facet_XXX(formula, ...)
```

  - Donde **XXX** será el tipo de facet a aplicar.
  - Donde **formula** definirá las variables usadas en cada eje para la desagregación.
  - Donde **...** serán los parámetros específicos de cada tipo de facet.

- Las fórmulas se definen con la sintáxis **<variables en y> ~ <variables en x>**.
  - En qplot, la existencia de <variables en y> marcará la diferencia entre wrap y grid.
  - En ggplot, la existencia de <variables en y> dependerá de si XXX es wrap o grid (y dará un error en caso de que la fórmula no coincida con el tipo de facet).

Juguemos con Facets...

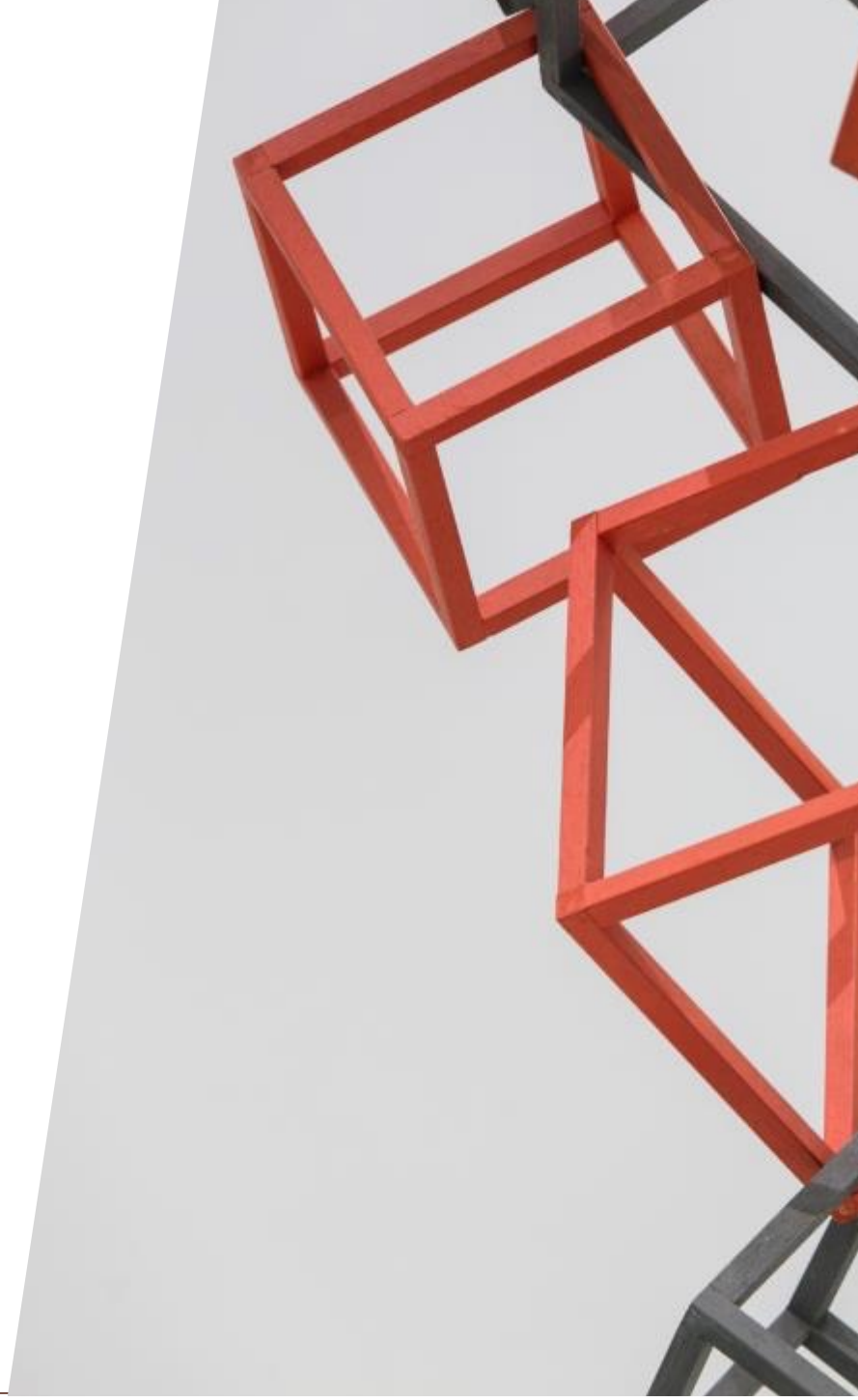
Script 09-facets.R

Ejercicio 4: Facets

Script E04-facets.R

Componentes de ggplot2

Coordinates



Componentes de ggplot2

Coordinates: Espacio del gráfico

- El componente **coordinates** (o coords) permite controlar todas las características relacionadas con el espacio donde el gráfico quedará representado.

- Estas características incluyen:
  - Tipo de sistema de coordenadas:
  - Límites de los ejes (de un modo "más coherente" que con scales).
  - Relación de aspecto: cuántas unidades de y representan una unidad de x.
  - Orientación de los ejes.
  - …

Definición de Coordinates

- Como hasta ahora, ggplot2 pone a nuestra disposición la definición de coords a través de otro conjunto de funciones.

```
ggplot(...) + geom_ZZZ(...) + coord_XXX(...)
```

  - Donde **XXX** será el tipo de coord a aplicar al gráfico.
  - Donde **(...)** serán los parámetros específicos de cada tipo de coord.

Juguemos con Coordinates...

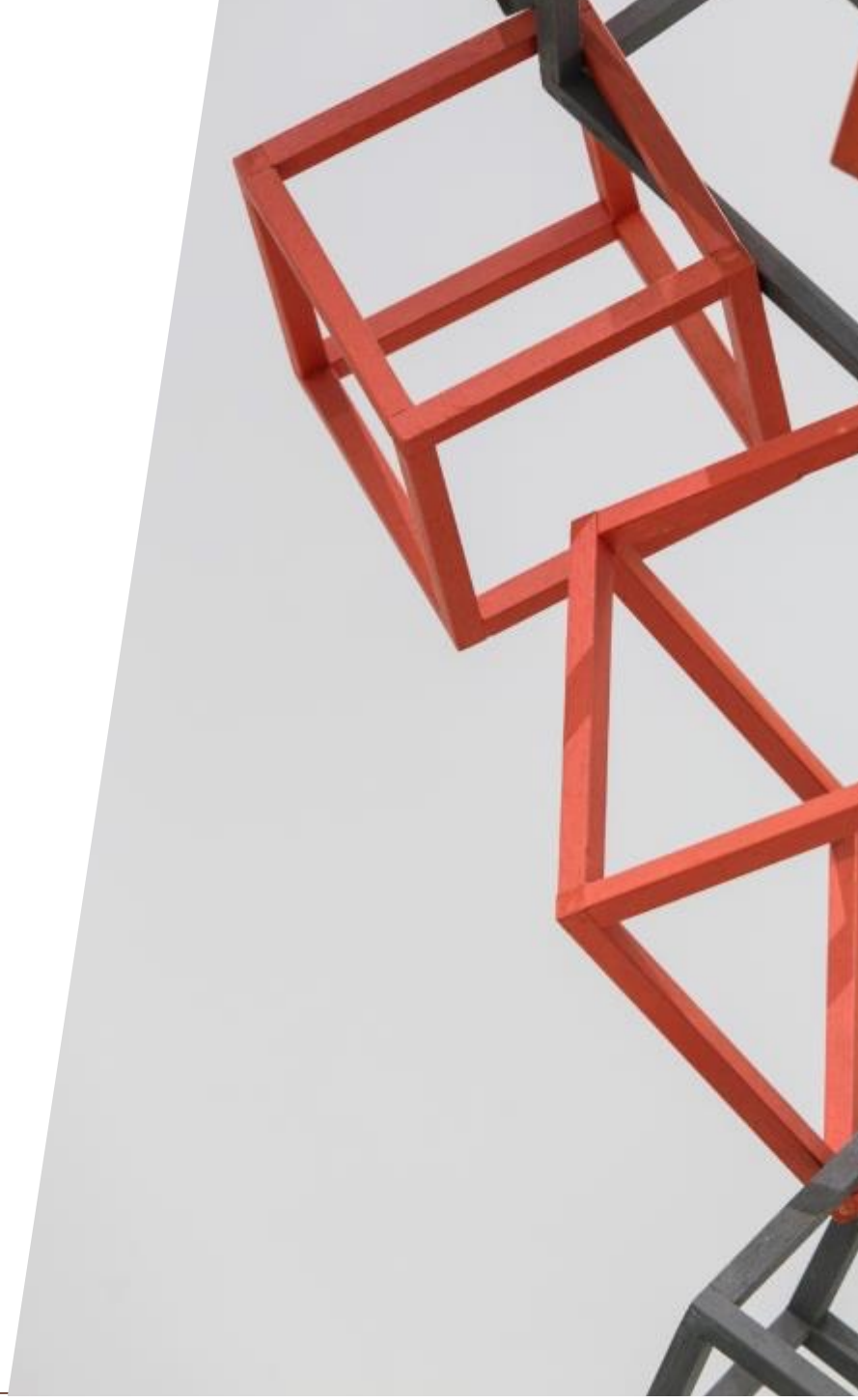
Script 10-coords.R

Ejercicio 5: Coordinates

Script E05-coords.R

Componentes de ggplot2

Themes



Componentes de ggplot2

Themes: Estilos de los gráficos

- El último componente de la gramática, themes, permite definir el estilo de todos los elementos del gráfico que no están directamente relacionados con los datos.

- Entre estos elementos podemos encontrar, por ejemplo:
  - Títulos de gráfico, ejes y leyenda.
  - Lineas de los ejes.
  - Rejilla.
  - Colores de fondos de gráfico, área de gráfico y leyenda.
  - ...

- Modificar y personalizar estos elementos es lo que nos permitirá generar gráficos con estilos distintos a los ofrecidos por defecto en ggplot2.

- Podemos, por tanto, **generar gráficos en ggplot2** que "no parezcan" **hechos con ggplot2**.

Definición de Themes

- Al contrario que en el resto de componentes, ggplot2 no permite la definición de themes a través de los parámetros de qplot:

```
ggplot(...) + geom_ZZZ(...) + theme(...)
```

  - Donde **(...)** incluirá el conjunto de elementos visuales modificados.

- El conjunto completo de elementos visuales se puede dividir en tres grupos:
  - Elementos de línea.
  - Elementos de texto
  - Elementos rectangulares / contenedores.

- El conjunto completo de elementos parametrizables se puede encontrar en:

http://ggplot2.tidyverse.org/reference/theme.html

Asignación de estilos a elementos visuales

- El establecimiento de un valor a un elemento visual de theme se debe realizar asignando el resultado de las funciones de la familia **element**_ en función del tipo de elemento.

  - Elementos de línea:

```
... + theme(xxx = element_line(color, size, linetype, lineend))
```

  - Elementos de texto:

```
... + theme(xxx = element_text(family, face, color, size, hjust,
                               vjust, angle, lineheight))
```

  - Elementos rectangulares / contenedores:

```
... + theme(xxx = element_rect(fill, color, size, linetype))
```

  - Eliminar elemento:

```
... + theme(xxx = element_blank())
```

Themes “predefinidos” y “por defecto”

- ggplot2 ofrece un conjunto de funciones que encapsulan temas “predefinidos” que se pueden usar directamente o utilizar como base de otros temas personalizados por el usuario:
  - theme_bw, theme_grey, theme_light, theme_linedraw...
- Además, se puede usar el paquete **ggthemes** para tener acceso a un listado amplio de temas predefinidos adicionales.
- ggplot2 nos ofrece, además funciones que permiten modificar el theme “por defecto” aplicado a los gráficos de forma que no tengamos que incluir la función theme en todos ellos:
  - theme_set: establece un theme como “por defecto”.
  - theme_get: obtiene un objeto con el theme actualmente “por defecto”.
  - theme_update: modifica (fusiona) el theme “por defecto” con el suministrado.

Juguemos con Themes...

Script 11-themes.R

Componentes de ggplot2

Annotations: Elementos no asociados a datos

- Aunque podría considerarse un componente independiente, los annotations permiten definir e incluir elementos no asociados directamente a los datos.

- Entre estos elementos podemos encontrar, por ejemplo:
  - Textos.
  - Líneas.
  - Rectángulos.
  - ...

- Estos elementos pueden utilizarse para incluir "aclaraciones" dentro del gráfico de forma que se pueda facilitar su comprensión por parte de los lectores.

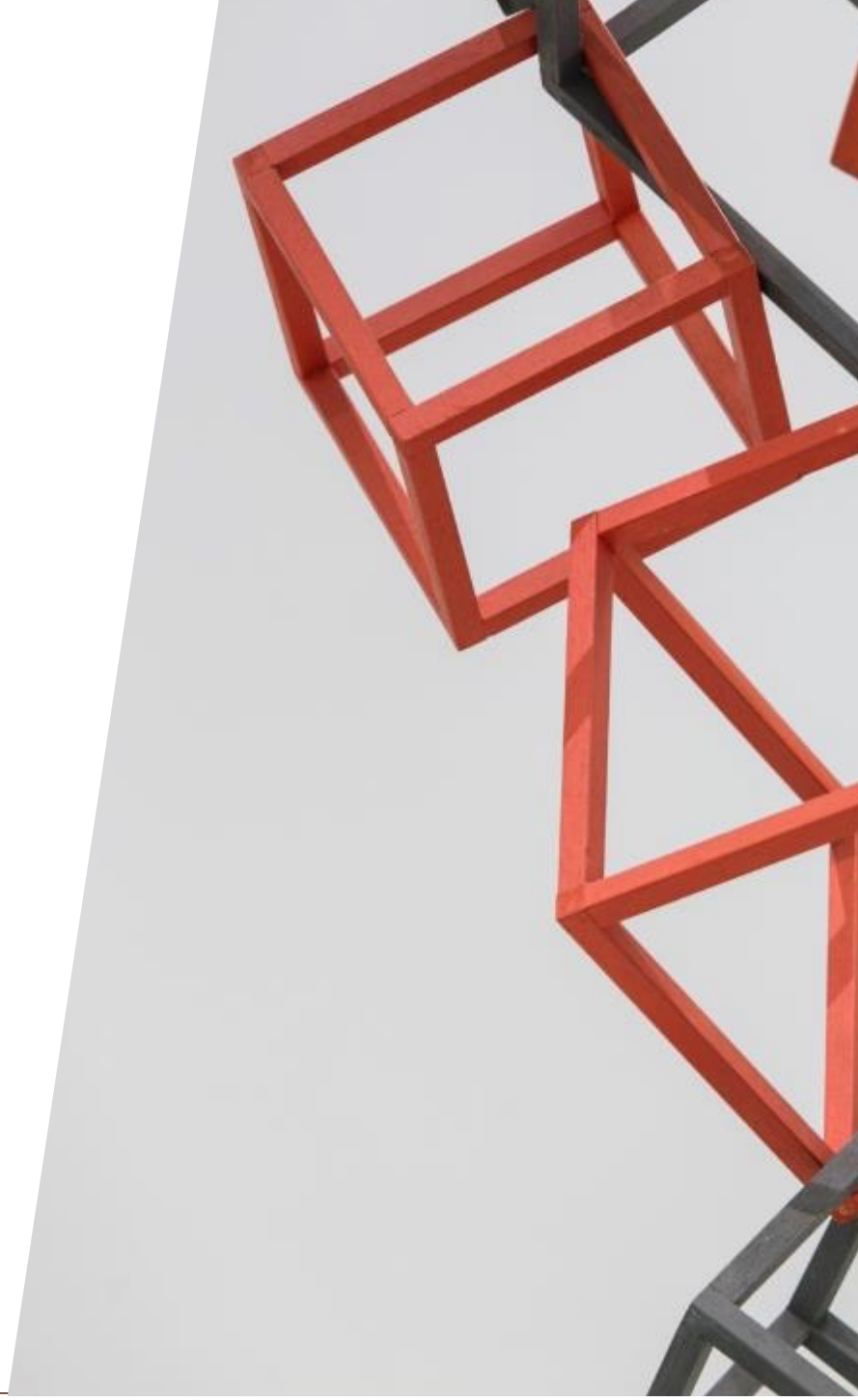
Juguemos con Annotations...

Script 12-annotations.R

Ejercicio 6: Themes y Annotations

Script E06-themes-annotations.R

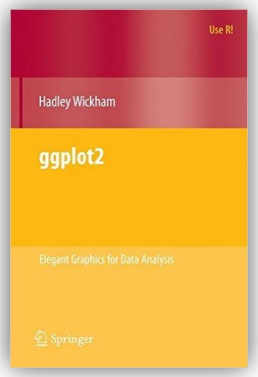
Referencias

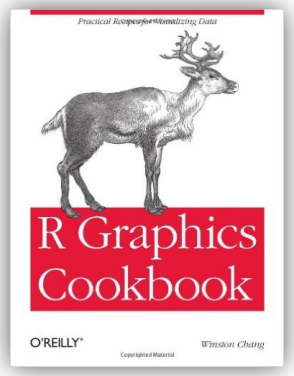


Referencias

- Online:
  - http://ggplot2.tidyverse.org/index.html
  - https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf
  - http://sape.inf.usi.ch/quick-reference/ggplot2

- Libros:
  - "ggplot2: Elegant Graphics for Data Analysis (Use R!)" – Hadley Wickam
  - "R Graphics Cookbook" – Winston Chang





Afi Escuela