

**Para ver el documento en html**

---

format: html

editor: visual

---

Crear el documento en un Jupiter notboock

---

jupyter:

kernelspec:

display\_name: R

language: R

name: ir

---

Paste lo que devuelve es una variable de texto

Print muestra los elementos

En los vectores c[] esto me genera un vector

**Variables en R**

Objetivo

Estudiar el concepto de variable en R

Vamos a

1. Realizar operaciones aritméticas en consola
2. Asignar variables a un script en RStudio
3. Identificar distintos tipos de datos

* R es una calculadora

Podemos sumar, restar, multiplicar o dividir

* Las operaciones aritméticas

ayudan a

* Crear nuevas variables para el análisis
* Realizar cálculos complejos
* Y resolver un... acertijo de redes sociales

1+1-1\*(1+1-1)/1+1\*(-1)

Una variable es un espacio en la computadora donde guardamos un objeto.

* Son útiles cuando vamos a:
* Utilizar los mismos datos varias veces
* Repetir operaciones aritméticas
* Estructurar análisis
* Puedes usar "--- o "="
* Aquí se utilizará “<-”
* Las guías de estilo son normas para escribir código bonito.
* Utiliza nombres que resuman el contenido de los datos
* Evita nombres genéricos
* X, y, z, var1, var2...
* R diferencia entre minúsculas y mayúsculas
* Datos, DATOS, datos son diferentes variables
  + mango <- 10
  + manzana <- 5
  + aguacate <- 15
  + resultado <- 2 \* aguacate + 4 \* manzana + 1.5 \* mango
  + print(resultado)

Tipos de datos

* Hablamos de números, textos o binarios.
* Para saber de qué tipo es una variable, utilizamos la función class()
* class(resultado)

Vectores en R:

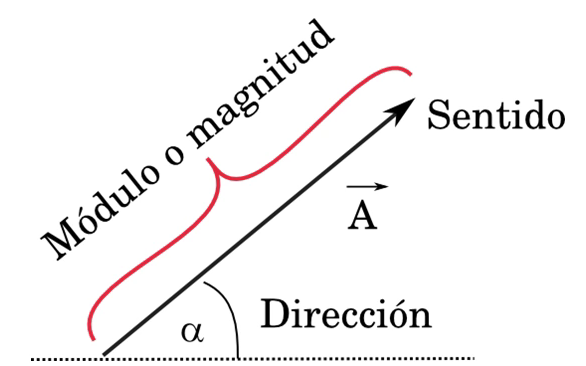
Objetivo

Estudiar el concepto de vectores en R

Vamos a:

1. Crear vectores en un script
2. Realizar operaciones aritméticas
3. Seleccionar elementos de un vector

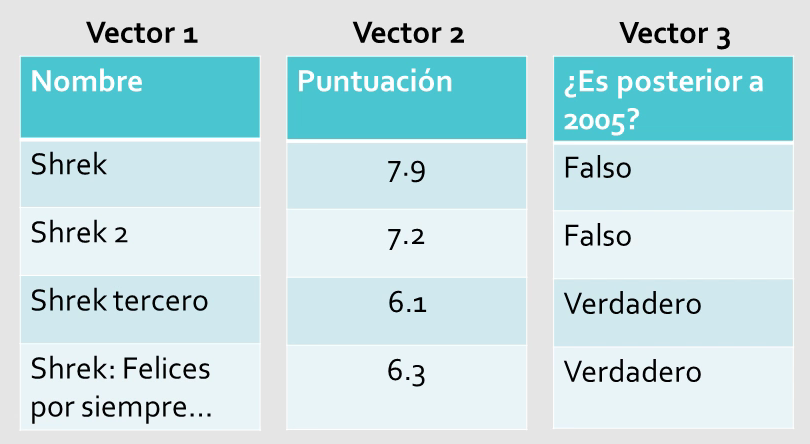
Vectores en matemáticas o física

* Algo que tenía magnitud o modulo, dirección y sentido
* Una cosa como (x,y,z)

Ejemplo:

Queremos guardar información.

Nombre de películas, puntuación, posterior al año 2005, los vectores pueden representar las columnas de una tabla de datos así:

Nota: es importante recordar que en R cada vector debe tener un solo tipo de dato, no podemos mezclar números con caracteres.

Utilizamos la función c(), la c significa combinar valores

* Vector numérico с(1, 2, 3, ...)

puntuacion <- c(7.9, 7.2, 6.1,6.3)

* Vector tipo carácter c("a", "b", "C", ...)

nombre <- c("Shrek", "Shrek 2", "Shrek tercero", "Shrek: Felices por siempre")

* Vector verdadero/falso c(TRUE, FALSE, ...)

posterior\_2005 <- c(FALSE, FALSE, TRUE, TRUE)

puntuacion + 2

puntuacion / 2

* Con valores numéricos

puntuacion\_de\_diana <- c(10, 9, 6, 7)

* Entre vectores

puntuacion\_de\_diana - puntuacion

* Operaciones aritméticas
* Suma, resta, multiplicación y división
* Funciones con vectores
* length(), sum(), mean()
* Longitud del vector:

length(puntuacion)

* Sumar los elementos del vector

sum(puntuacion)

Promedio del vector

mean(puntuacion)

* Selección basada en posición

nombre[3]

Primera, segunda, ... .

nombre[c(1,4)]

* Selección basada en condición

puntuacion\_baja <- puntuacion < 7

* mostrar puntuaciones menores que 7

puntuacion[puntuacion\_baja]

* mostrar nombres que tengan puntuaciones menores que 7

nombre[puntuacion\_baja]

Puntuación mayor a 7, ...

puntuacion\_alta <- puntuacion > 7

Matrices en R

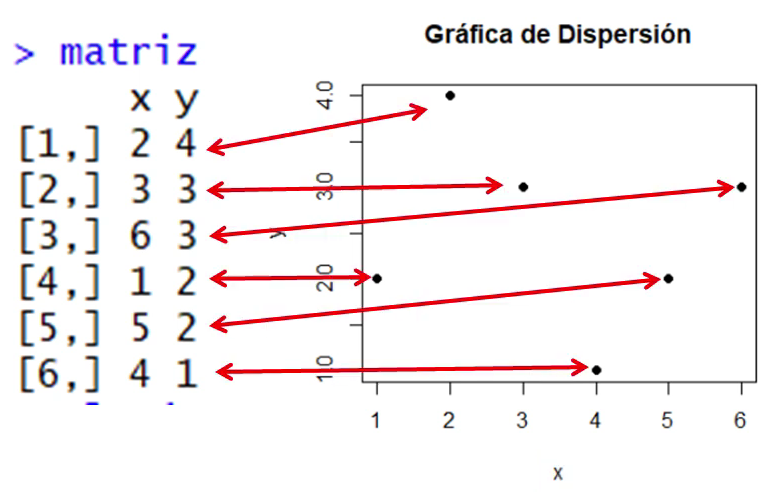
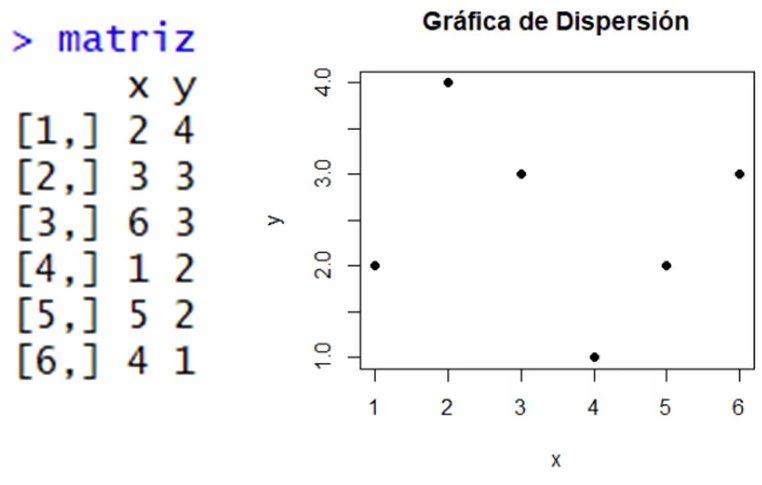
Objetivo

Estudiar qué son matrices en R

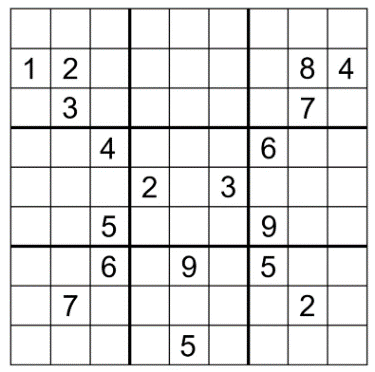
Vamos a

1. Crear matrices en R
2. Realizar operaciones aritméticas
3. Seleccionar elementos de una matriz

Para poder gráficar en R!

Una matriz de dos columnas y un gráfico de dispersión son prácticamente lo mismo

Es una "forma" de acomodar los datos que tiene renglones/filas y columnas.



Datos numéricos

* Cantidad de películas hechas por Warner, Disney
* Periodo 2010 a 2018
* Página web en la descripción.

Una forma es combinando varios vectores.

Utilizar la función matrix(...)

matrix (data , contenido de la matriz

nrow = 1, cuantas filas

ncol=1) cuantas columnas

Cambiar nombres con colname y rowname

warner <- c(20, 20, 16, 17, 17, 22, 17, 18, 19)

disney <- c(11, 13, 11, 8, 12, 11, 12, 8, 10)

fox <- c(18, 15, 15, 15, 16, 17, 15, 13, 11)

peliculas <- matrix(c(warner, disney, fox),

nrow = 9,

ncol = 3)

* Agregar nombres de columnas

colnames(peliculas) <- c('warner', 'disney', 'fox')

* Agregar nombres de filas/renglones

rownames(peliculas) <- c('2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018')

print(películas)

warner disney fox

2010 20 11 18

2011 20 13 15

2012 16 11 15

2013 17 8 15

2014 17 12 16

2015 22 11 17

2016 17 12 15

2017 18 8 13

2018 19 10 11

* Sirve para modificar una matriz
* Las operaciones son la suma, resta, multiplicación y división
* Las operaciones básicas entre matrices son elemento a elemento
  + Sumar matriz consigo misma (se hace elemento a elemento)

peliculas + peliculas

* + Multiplicar la matriz consigo mismo (se hace elemento a elemento)

peliculas \* películas

Nota: Esta multiplicación NO es la de álgebra lineal

En álgebra, la multiplicación es más complicada, esta fuera del alcance de este video

Puedes probar haciendo

peliculas %\*% películas

* Selección de elementos (es como tomar ciertas celdas de una tabla de Excel)

peliculas[3,2]

peliculas['2012', 'disney']

* Selección de filas/reglones

peliculas[c(3,2), c(2,3)]

peliculas[c(3,2), c('disney', 'fox')]

* Selección de columnas

peliculas[,2]

peliculas[,'disney']

¿Qué son los factores en R?

Es una estructura de datos para manejar variables categóricas

Son datos que toman una cantidad finita de valores

* Rango de edad: niño, joven, adulto
* Colores: rojo, verde, amarillo.
* Día de la semana: L/M/X/J/V/S/D

En análisis de datos, variables categóricas se tratan diferente a las continuas o numéricas.

¡Necesitamos un objeto en R para eso! Gráficas y modelos.

Ejemplo:

Playera/franela/camiseta/remera

Ventas por talla (S/M/G)

tallas <- c('m', 'g', 'S', 'S','m', 'M')

Creamos un factor a partir de tallas

tallas\_factor <- factor(tallas)

mirar niveles de un factor

levels(tallas\_factor)

Utilizamos la función factor() con el vector como argumento

factor (vector)

las categórias o clases de un factor se llaman niveles y podemos mirarlo utilizando la función levels pasando el factor como argumento

levels (factor)

tallas\_recodificado <- factor(tallas,

levels = c('g', 'm', 'M', 'S'),

labels = c('G', 'M', 'M', 'S'))

Modificar la función factor()

factor (vector,

levels = vector\_niveles,

labels = vector\_etiquetas)

Agregar imagen con la función y sus argumentos

tallas\_ordenado <- factor(tallas,

ordered = TRUE,

levels = c('S', 'm', 'M', 'g'),

labels = c('S', 'M', 'M', 'G'))

print(tallas\_ordenado)

[1] M G S S M M

Levels: S < M < G

DataFrame en R

Objetivo

Estudiar qué son DataFrames en R

Vamos a

1. Crear DataFrames
2. Seleccionar elementos de un DataFrames
3. Ordenar DataFrames por columnas

Los DataFrames son como las hojas de calculo en Excel, son la estructura de datos con la que se hace prácticamente todo.

Recordar:

* Variables: espacio para guardar un objeto
* Vectores: una o más variables del mismo tipo
* Matrices: Varias columnas (o vectores) del mismo tipo como columnas
* Pregunta: ¿Cómo utilizo columnas de diferentes tipos en una misma tabla?: utilizando DataFrames, los cuales son unas estructuras de datos donde se utilizan diferentes tipos de variables, estos son importantes para gráficas, análisis y modelos con muchas variables estadísticas.

¿Cómo crear un DataFrame en R?

Hay varias formas, pero la mas sencilla es utilizando vectores.

Utilizar la función data.frame(...)

Se envían los vectores como argumentos, esto vectores deben tener la misma cantidad de datos o si no dará como resultado un error

data. frame (vector\_1,

vector\_2,

vector\_3,

vector\_etc)

Cambiar nombres con names(...)

* Crear vectores de igual dimensión:

nombre <- c("Shrek", "Shrek 2", "Shrek Tercero", "Shrek: Felices por siempre")

puntuacion <- c(7.9, 7.2, 6.1, 6.3)

posterior\_2005 <- c(FALSE, FALSE, TRUE, TRUE)

* Crear DF:

peliculas\_df <- data.frame(nombre,

puntuacion,

posterior\_2005)

* Mostrarlo:

peliculas\_df

* Cambiar nombres de las columnas:

names(peliculas\_df) <- c('NOMBRE',

'PUNTUACION',

'POSTERIOR\_2005')

* Mostrarlo nuevamente:

peliculas\_df

¿Cómo seleccionar elementos de un dataframe?

* Selección de un elemento
* Selección de filas/reglones
* Selección de columnas
  + Aquí es donde viene el $
* Podemos empezar seleccionando un solo elemento

peliculas\_df[3,2]

peliculas\_df[3,'PUNTUACION']

* Podemos escoger dos elementos y escribir cada uno como un vector

peliculas\_df[c(3,4), c(2,3)]

peliculas\_df[c(3,4), c('PUNTUACION', 'POSTERIOR\_2005')]

Nota: Para seleccionar una fila o renglón especifico, escribimos su numero separado de una coma.

* Seleccionar una fila o renglón del dataframe

peliculas\_df[3,]

* Seleccionar una columna del dataframe

peliculas\_df[,2]

peliculas\_df[,'PUNTUACION']

peliculas\_df$PUNTUACION

¿Cómo ordenar el DataFrame?

* Queremos ordenar el dataframe por una columna
* Tenemos que hacer dos pasos:

1. Guardar los índices ordenados.

Índices <- order(dataframe$COLUMNA)

* Mostrar el indice de la columna de puntuacion con order

order(peliculas\_df$PUNTUACION)

* Funcion order (menor a mayor) seleccionamos las filas en el orden creado

orden\_menor\_mayor <- order(peliculas\_df$PUNTUACION, decreasing = FALSE)

1. Seleccionar el DataFrame con índices obtenidos en el paso 1.

* Mostrar el dataframe ordenado

peliculas\_df[orden\_menor\_mayor,]

* Funcion order (mayor a menor)

orden\_mayor\_menor <- order(peliculas\_df$PUNTUACION, decreasing = TRUE)

* Mostrar el dataframe ordenado

peliculas\_df[orden\_mayor\_menor,]

* Guardar el dataframe ordenado

df\_ordenado <- peliculas\_df[orden\_mayor\_menor, ]

* Mostrar el dataframe

df\_ordenado

Listas en R

Objetivo

Estudiar qué son las listas en R

Vamos a

1. Crear listas en R
2. Seleccionar elementos de una lista
3. Agregar/eliminar elementos a una lista

* Los dataframe guardan diferentes tipos de datos en columnas
* Las listas guardan diferentes objetos en un mismo objeto en R, son usadas por otras funciones de R para guardar sus resultados

Caso de regresión lineal

* Para crear una Lista, utilizar la función list(...) y pasamos como argumentos los elementos que queremos incluir

list (objeto1,

objeto2,

...)

lista\_curso <- list(vector\_titulos,

matriz\_peliculas)

* Cambiar nombres con names(...)

names(lista\_curso) <- c('VECTOR', 'MATRIZ')

* Seleccionar un elemento

lista [['nombre']]

lista\_curso[['VECTOR']]

* Vector de una lista:

lista[['vector']][4]

lista\_curso[['VECTOR']][3]

* Df o matriz:

lista[['dataframe'] [8, 'columna']

lista\_curso[['MATRIZ']][5,3]

* Para agregar un elemento

lista[['nombre’]] <- objeto

lista\_curso[['data\_frame']] <- peliculas\_df

* Para eliminar un elemento

lista[['nombre']] <- NULL

lista\_curso[['VECTOR']] <- NULL

Importar de Excel a R

Objetivo

importar datos de excel a R

Vamos a ver

Que necesitas antes de empezar

1. Paquete readxl

2. Ruta de archivo

Cómo importar datos con código de R

Cómo importar datos con interfaz de RStudio

Importar detos de Excel

Paquete readxl

En mi opinión es el más sencillo

* Instalar paquete readxl

install.packages("readxl")

* Cargar paquete readxl

library(readxl)

* Buscar la ruta del archivo de excel(comando para poder acceder a la ruta del archivo)

file.choose()

* Copiar ruta de la consola y guardar en variable

ruta\_excel <- "D:\\Users\\diana\\Downloads\\DATA MINING\\data-mining-main1\\R-STUDIO\\PRACTICA\\gapminder\_importar\_a\_r.xlsx"

* Cómo mirar las hojas de tu excel

excel\_sheets(ruta\_excel)

* Importar datos de Excel

read\_excel (ruta\_excel)

Casos como viene una tabla Excel

* Ideal: primera hoja, A1

caso\_ideal <- read\_excel(ruta\_excel)

si vemos la variable por la consola nos sale una estructura de datos “tibble, que es equivalente a un dataframe”

* Medio: no primera hoja, A1

caso\_medio <- read\_excel(ruta\_excel,

sheet = 'Hoja2')

* Final Boss: no primera hora, no A1

final\_boss <- read\_excel(ruta\_excel,

sheet = 'Hoja3',

range = 'C7:F17')

**Graficar en R**

Objetivo

Diferenciar formas de graficar en R

Vamos a ver

* Cómo se grafica con clicks VS con código
* Como graficar con R base graphics
* Otros paquetes para graficar en R

ggplot2

Más

Películas hechas por Disney cada año

Varaible carácter "year" para años

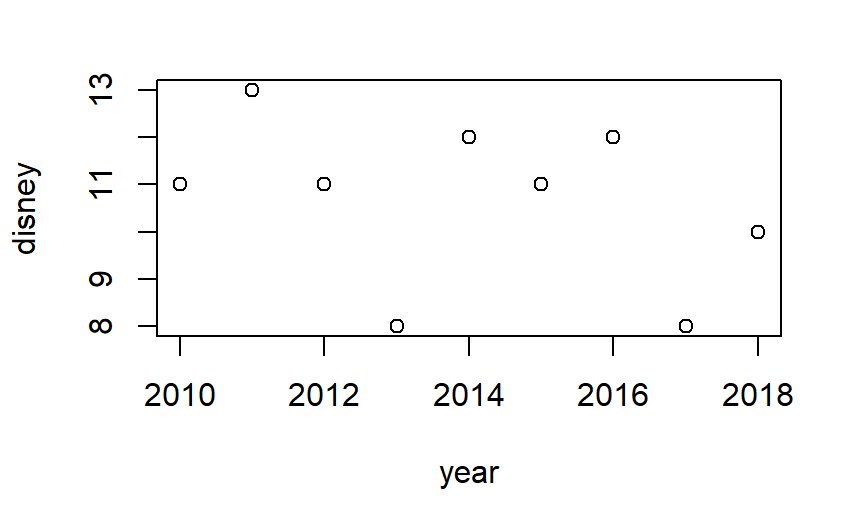
Variable numérica "disney" para peliculas

* Lo bueno
* Replicable y reproducible
* ¡Más opciones y gráficas!
* Lo no tan bueno
* Hay que aprender más código
* Edición puede tomar tiempo
* Mira está página (galería con ejemplos de código)

https:/ [www.r-graph-galleny.com/](http://www.r-graph-galleny.com/)

* Datos

year <- c('2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018')

disney <- c(11, 13, 11, 8, 12, 11, 12, 8, 10)

* Graficando con codigo

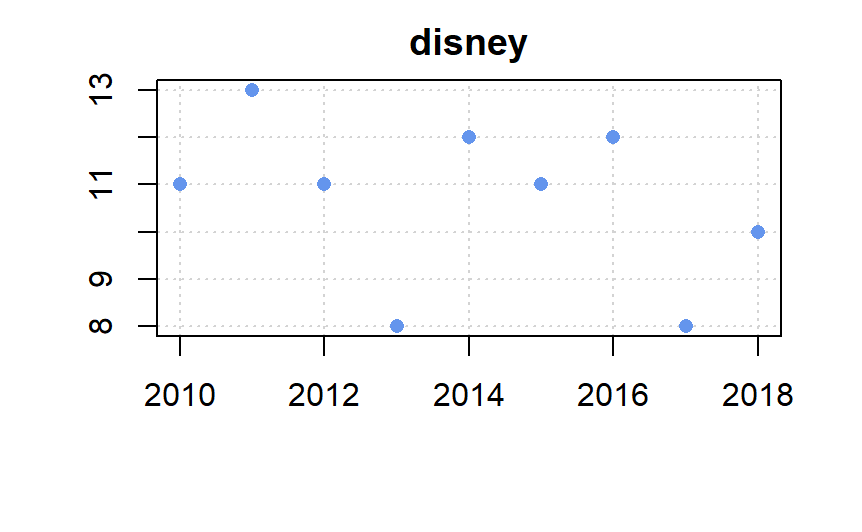
plot(x = year,

y = disney)

* Editando la grafica

plot(x = year,

y = disney,

 main = 'disney',

xlab = '',

ylab = '',

col = 'cornflowerblue',

pch = 16,

panel.first = grid())

* Otras funciones para hacer gráficas básicas en R
* barplot()
* hist()
* pie()
* Utiliza la gramática de los gráficos
* Partes de una oración:

Sujeto, verbo, predicado

* Partes de una gráfica

Layers o capas

* Principales capas
  1. Datos
  2. Mapping
  3. Geometrías
* Elementos visuales (aesthetics)
* Eje X
* Eje y
* Color
* Tamaño
* Para instalar ggplot2

install.package("ggplot2")

library(ggplot2)

* Hacer dataframe

peliculas <- data.frame(year,

disney)

* Graficar utilizando ggplot

ggplot(data = peliculas, # dataframe o primera capa

mapping = aes(x = year,

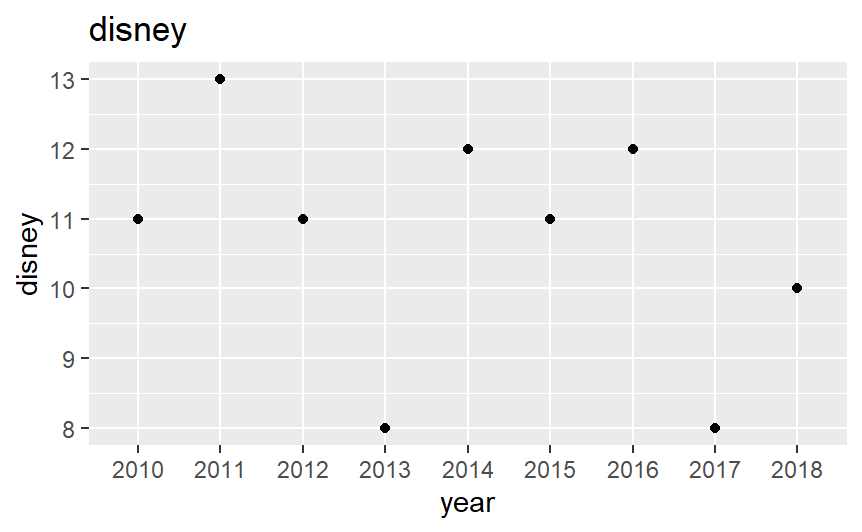
y = disney)) + # escoger qué campo o columna de los campos va en qué elemento

visual o elemento estético

geom\_point() + # esta capa cambia, según el tipo de grafica que queremos

labs(title = 'Disney') # modificar el titulo se añade otra capa llamada labs, para

poner varias capas se usa el más +

# Con esto, hemos hecho una gráfica básica en ggplot

* Otros paquetes

Flexdashboard

Highcharter

RGL

Dygraphs

sunburstR

Plotly

Lattice

Leaflet

RColorBrewer

Ggvis

Shiny

**Histograma en R**

Objetivo

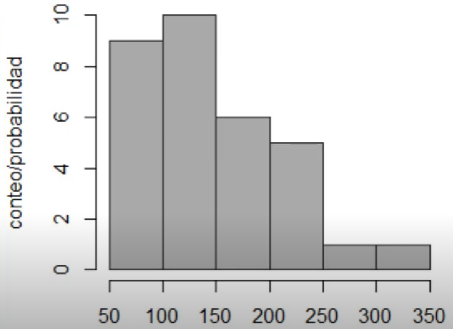
Hacer histogramas en R

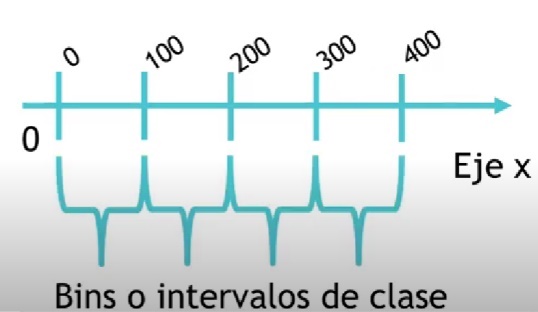
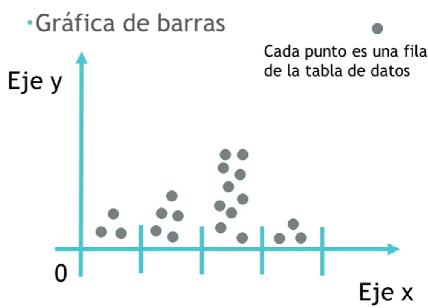
Vamos a ver

Qué son los histogramas

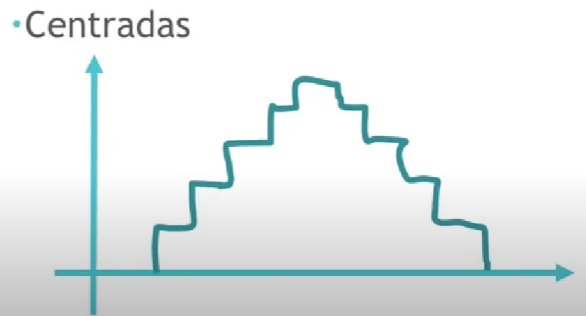
Histogramas con base graphics

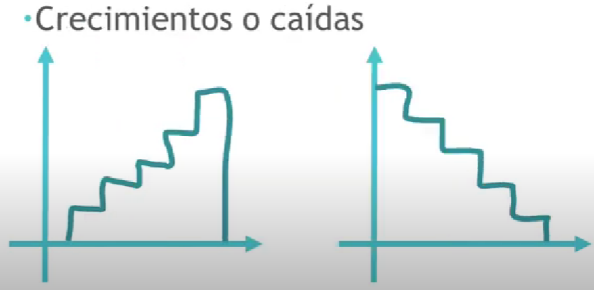
Histogramas con ggplot2

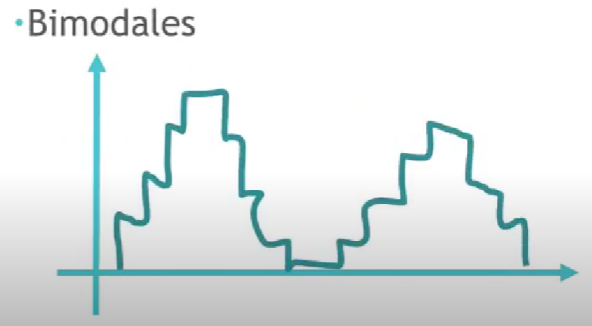
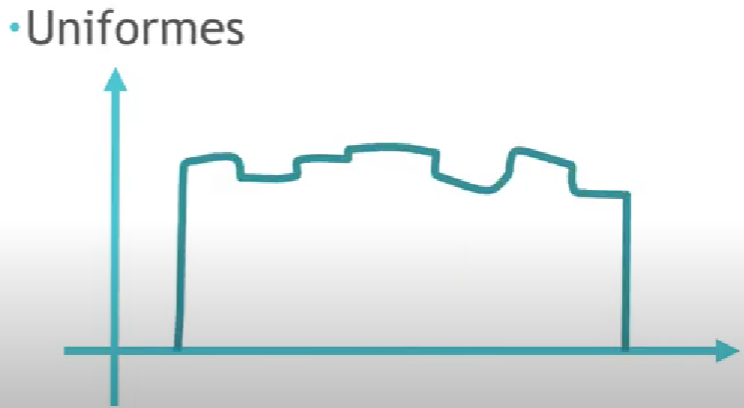
Un histograma es una gráfica de barras con la distribución de la frecuencia.



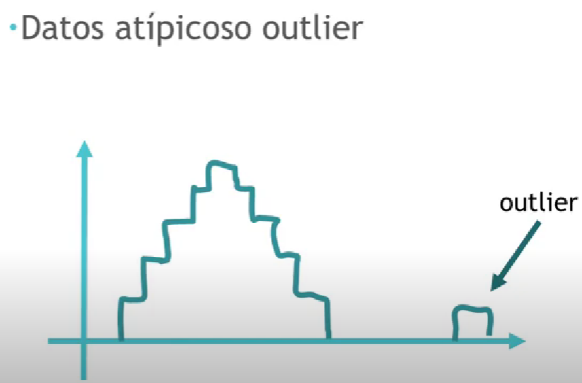
* En el eje x, va la variable continua que estamos estudiando, esta variable la dividimos en espacios para contar cuantos datos caben en cada espacio, estos espacios se llaman bins o intervalos de clase.
* En el eje y, va el conteo de datos por bin, hacer estos conteos nos da una distribución de la frecuencia y sirve para tener una idea de donde está la mayoría de los datos.

Algunas posibles distribuciones que podemos encontrar son:





En algunos casos, los histogramas ayudan a detectar outlayers o datos atípicos



**En estadística**

* La frecuencia es la cantidad de veces que se repite una cosa
* La distribución es como se reparten esas cosas en una variable
* La tabla de frecuencia contiene los conteos, por lo tanto un histograma es una visualización de esa tabla, es decir una gráfica de barras con la distribución de la frecuencia.

Nota: cuando se necesite explorar datos con histogramas, se rrecomienda utilizar rbase ya que es más rápido de escribir el código y los histogramas que debamos presentar, los podemos hacer con ggplot2

**Datos precargados en R**

MTCARS: rendimientos de coches/carros/autos

32 observaciones, 11 variables

Caballos de fuerza, tipo de motor, cilindros, y esas cosas...

* Cargar los datos

data("mtcars")

* Para hacer un histograma, utilizamos la función hist() y le pasamos como argumento la variable que tiene los datos.
* Si es un vector: hist(vector)
* Si es un dataframe: hist(dataframe$columna)

En este caso, la información precargada sería un histograma

hist(mtcars$hp) # Caballos de Fuerza

* Si aumentamos los bins, damos mas detalle al histograma, solo que no hay que pasarse, ya que después el histograma no es informativo

hist (mtcars$hp,

breaks = 10)

* + También podemos usar la opción break o puntos de corte para decir con un vector numérico donde queremos hacer las separaciones, para cambiar los breaks, se puede utilizar la función seq que crea números seguidos empezando con el primero, terminando con el segundo y avanzando según el tercero

hist(mtcars$hp,

* Hay más opciones para cambiar cosas como el color, el relleno, el título y el nombre que va en los ejes, con esto, ya podemos empezar a extraer más información de los datos y hacer histogramas sencillos

hist(mtcars$hp,

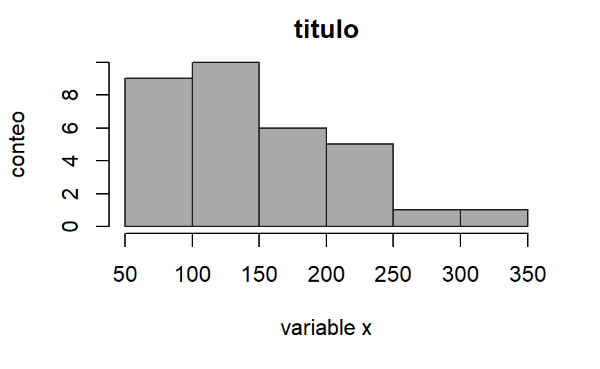
col = 'darkgray',

border = ‘gray10’,

main = 'titulo',

xlab = 'variable x',

ylab = 'conteo')

hist(mtcars$hp,

breaks = seq(50, 350, 50),

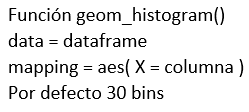
col = 'darkgray',

border = 'gray10',

main = 'titulo',

xlab = 'variable x',

ylab = 'conteo')

* Lo mismo, podemos hacerlo utilizando el paquete ggplot2, acá las graficas se ven un poco más profesionales, pero la información es exactamente la misma, la forma del código si cambia un poco
* Después de cargar la librería, llamamos a la función ggplot, pasamos como argumento los datos y el mapin, allí incluimos solo la variable de estudio en x, luego se agrega la capa de geometrías (geom\_histogram)

ggplot(data = mtcars,

mapping = aes (x = hp)) +

geom\_histogram(bins = 9)

* + Tenemos 2 histogramas de la variable x, separados por otra variable, la otra variable es el tipo de motor

Separando por colores

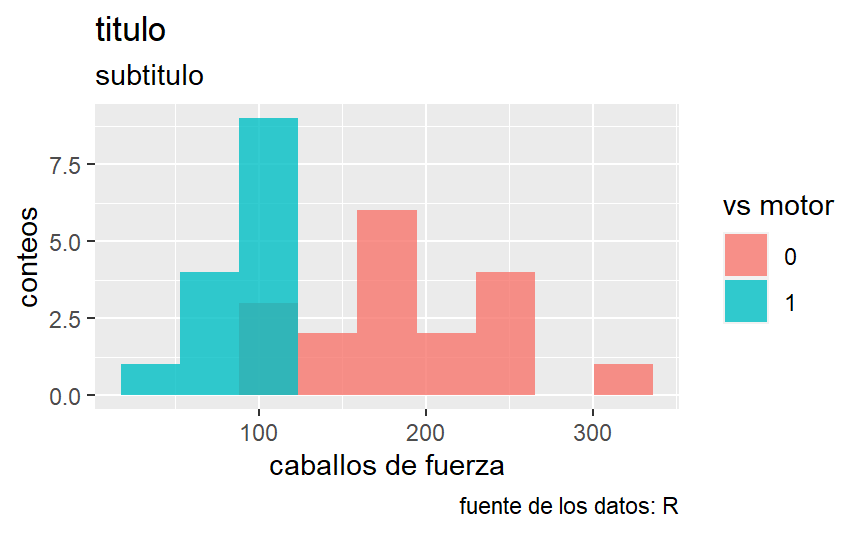
1 es v-shaped, si el motor es en forma de v

0 es manual, si el motor es manual

Lo que hacemos es explorar los caballos de fuerza considerando el tipo de motor

en el maping anexamos el tipo de motor, para decirle que queremos que cambie en los colores de relleno fill = factor(vs)

en este caso usamos factores porque vs es por defecto una variable numérica, además podemos cambiar otras opciones para que el histograma se vea diferente, títulos, subtitulo y más.



ggplot(data = mtcars,

mapping = aes (x = hp,

fill = factor (vs)))+

geom\_histogram(bins = 9,

position = 'identity',

alpha = 0.8) +

labs(title = 'titulo',

fill = 'vs motor',

x = 'caballos de fuerza',

y = 'conteos',

subtitle = 'subtitulo',

caption = 'fuente de los datos: R')

**Graficas de barras en R**

Objetivo

Hacer gráficas de barras en R

Vamos a ver

* Qué son las gráficas de barras
* Gráficas de barras con base graphics
* Gráficas de barras con ggplot2

Las gráficas de barra son un tipo de visualización que utilizan variables categóricas con barras rectangulares, con alturas proporcionales al valor de la medida que representan.

Utiliza variables categóricas

La altura de las barras es proporcional al valor de la medida que representa

Fáciles de entender por todos

* R base graphics(viene con R)
* Paquete ggplot2(instalar paquete)
* Datos mtcars(viene con R)

Podemos utilizar R para hacer gráficas de barras

Función barplot()

Esta función, debe contener como argumento una tabla donde esté la variable categórica y las alturas, para utilizar los datos, necesitamos agrupar la información por cilindros

barplot(table(dataframe$columna))

Agrupa los datos

barplot(table(mtcars$cyl),

horiz = 'TRUE',

col = 'green',

border = 'red',

main = 'gráfica de barras',

sub = 'subtitulo',

xlab = 'cilindros',

ylab = 'cantidad')

Función geom\_bar()

data = dataframe

mapping = aes(x = columna)

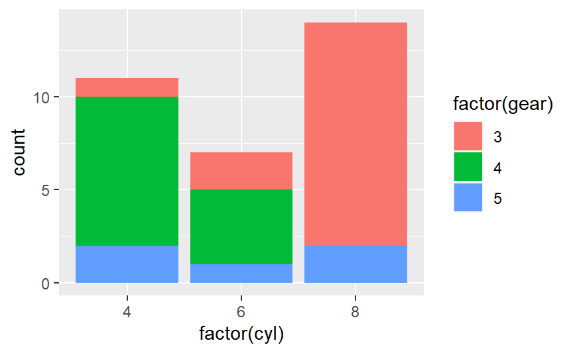
Función geom\_bar()

Cambiar opción position con

"stack"

"dodge"

"fill"

* guardando grafica en variable

p <- ggplot(data = mtcars,

mapping = aes(x = factor(cyl),

fill = factor(gear)))

* stacked bar chart

p + geom\_bar(position = 'stack', stat = 'count')

* dodge bar chart

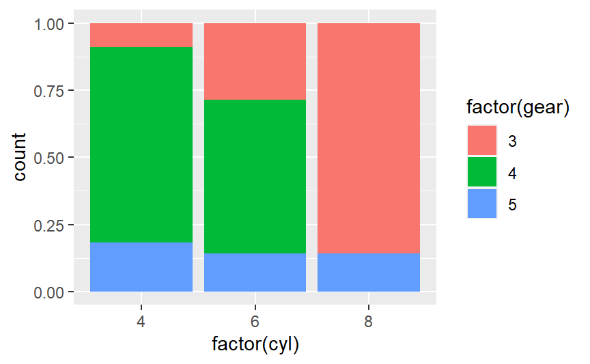
p + geom\_bar(position = 'dodge', stat = 'count')

Gráfico, Gráfico de barras

Descripción generada automáticamente

* stacked + percent barchart

p + geom\_bar(position = 'fill', stat = 'count')



**Objetivo**

Importar datos de CSV a R

Vamos a ver

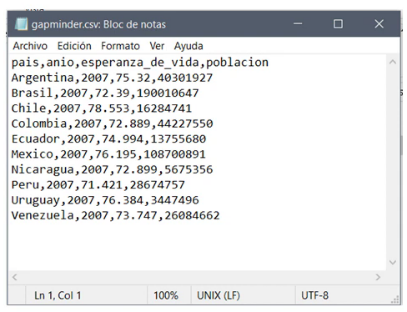
Que necesitas antes de empezar

1. Paquete readr

2. Ruta de archivo

Cómo importar datos con código de R

Cómo importar datos con interfaz de RStudio



Archivo de texto que tiene columnas separadas por comas y una fila en cada línea.

Coma separated values

Cuando importamos a R, convertimos este texto en un DataFrame.

Paquete readr

En mi opinión es el más sencillo

¡Si conoces otro déjalo en un comentario!

* Instalar paquete

install.packages("readr")

* Cargar paquete

library(readr)

* Buscar la ruta del archivo de csv

file.choose()

* Pegamos la ruta obtenida con el comando file.choose() en una variable ruta\_csv

ruta\_csv <- "D:\\Users\\diana\\Downloads\\DATA MINING\\data-mining-main1\\R-STUDIO\\PRACTICA\\gapminder.csv"

* Para cargar datos utilizamos la función

read\_csv (ruta\_csv)

Casos posibles

* CSV con títulos

gapminder <- read\_csv (ruta\_csv)

* CSV sin títulos

ruta\_sintitulo <- "D:\\Users\\diana\\Downloads\\DATA MINING\\data-mining-main1\\R-STUDIO\\PRACTICA\\gapminder\_col\_names.csv"

gapminder\_sintitulo <- read\_csv(ruta\_sintitulo,

col\_names = FALSE)

Debemos agregar un vector de texto a la opcion colnames, debe tener tantos nombres como columnas del df

De esta manera podemos cambiar el nombre de las columnas de cualquier archivo

gapminder\_contitulo <- read\_csv(ruta\_sintitulo,

col\_names = c('pais',

'anio',

'vida',

'poblacion'))

* CSV separado por punto y coma

ruta\_puntoycoma <- "D:\\Users\\diana\\Downloads\\DATA MINING\\data-mining-main1\\R-STUDIO\\PRACTICA\\gapminder\_puntoycoma.csv"

gapminder\_puntoycoma <- read\_csv2(ruta\_puntoycoma)

* ir a File > Import Dataset > From Text (readr)

Seleccionamos el archivo csv pra importar, podemos ajustar cosas como el nombre de las variables en R, si la primera columna tiene nombre, el tipo de separador, y otras opciones más avanzadas.

**Varias graficas con ggplot2**

Objetivo

Estudiar casos de uso de geometrías

Vamos a

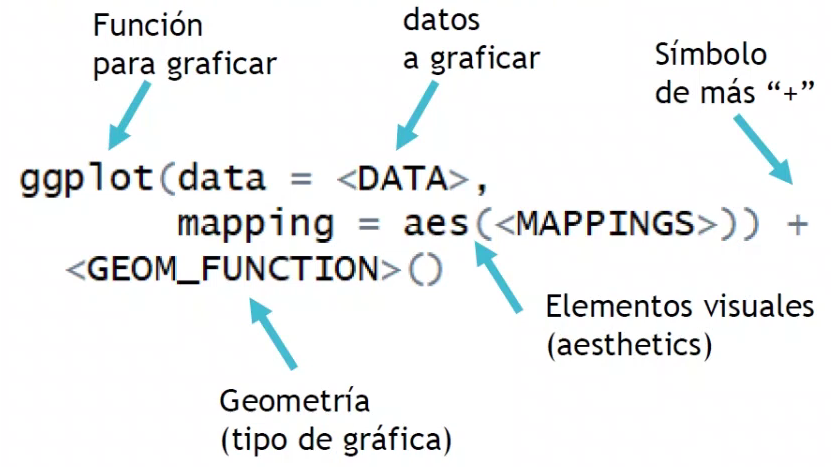
1. Entender la plantilla básica para armar (casi) cualquier gráfico en ggplot2

2. Caso puntos y líneas

3. Caso varías líneas

4. Caso boxplot y puntos

* Las gráficas que veremos en ggplot tienen la siguiente estructura

ggplot (data = <DATA>,

mapping = aes (<MAPPINGS>)) +

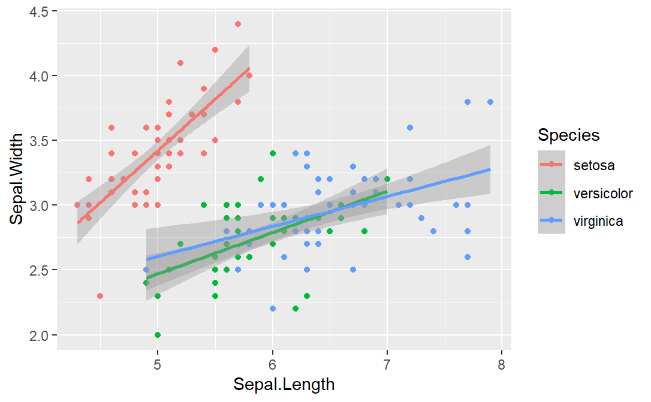
<GEOM\_FUNCTION>

* Geometrías geom\_point, geom\_smooth, geom\_line, geom\_boxplot y geom\_jitter. Veremos como superponer o sumar distintas geometrías para obtener gráficas más interesantes!
  + Bargar la librería

library(ggplot2)

* + Datos iris, flores utilizadas en machine learning

data("iris")

* + Primeras filas de los datos

head(iris)

* Ejemplo 1: puntos y líneas

ggplot(data = iris,

mapping = aes(x = Sepal.Length,

y = Sepal.Width,

# Variable con los tres tipos de flores

color = Species))+

geom\_point() +

# Linea que pase por encima de los puntos y describa una tendencia más o menos lineal, para usar lineas rectas debemos usar method = "lm"

geom\_smooth(method = 'lm')

# Esto es una especie de regresión lineal, si ejecutamos, vemos líneas rectas con tendencia positiva

# vamos a escoger el subconjunto de todas las flores que tengan especie igual a setosa, colocamos una coma antes de cerrar el corchete para que nos selecciones solamente esos datos

ggplot(data = iris[iris$Species == 'setosa',],

# elementos estéticos, como queremos ver el compportamiento por variable, esto significa un vector que va de 1 a 50

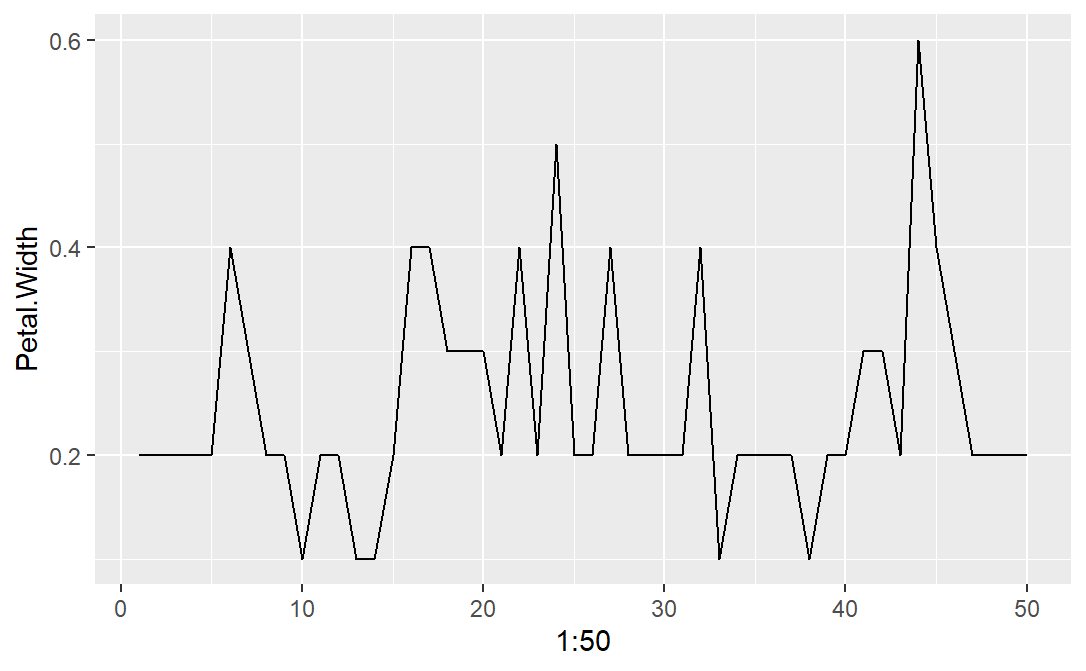
mapping = aes(x = 1:50,

y = Petal.Width))+

# para hacer las lineas, debemos agregar la geometría de geom\_line()

geom\_line()

# Al ejecurtar, vemos el comportamiento de las lineas a lo largo de los 50 datos que conforman la variable iiris$Species con el Petal.Width para la variable setosa



* todas las líneas para las 3 variables

ggplot(data = iris,

# elementos estéticos, como queremos ver el compportamiento por variable, esto significa un vector que va de 1 a 50, solicitamos que lo haga 3 veces con rep(1:50,3)

mapping = aes(x = rep(1:50, 3),

y = Petal.Width,

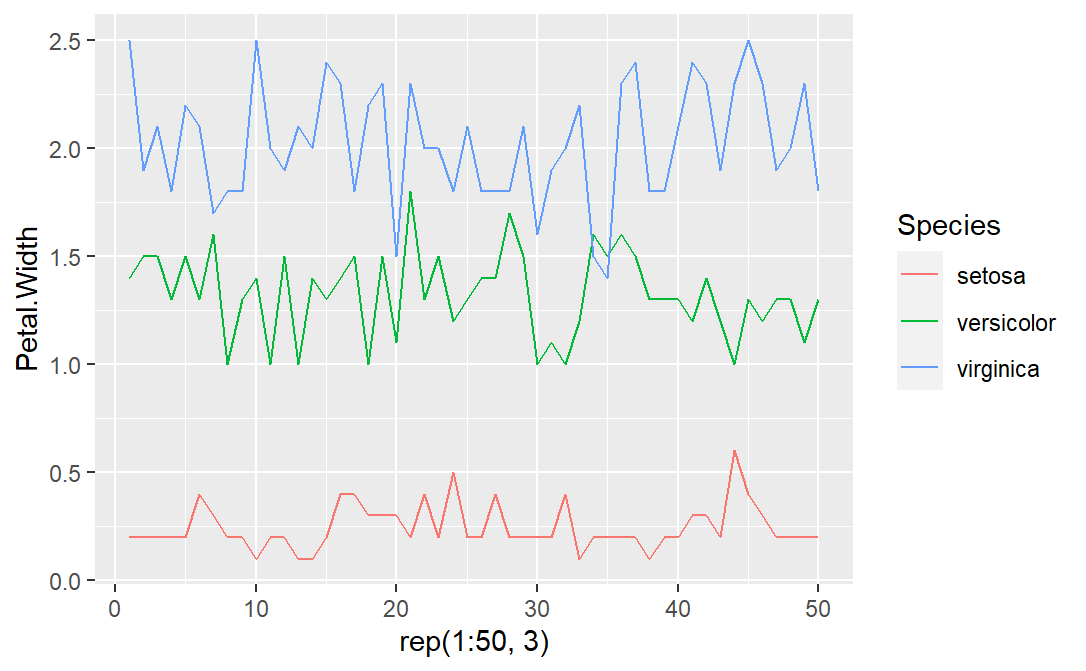
color = Species))+

# para hacer las lineas, debemos agregar la geometría de geom\_line()

geom\_line()

# Al ejecurtar, vemos el comportamiento de las lineas a lo largo de los 50 datos que conforman cada variable

# este tipo de gráfico es muy util cuando vamos a trabajar con series de tiempo, a lo largo de la semana o mes, y quieres ver el comportamiento de una variable como el precio o la temperatura etc a traves del tiempo para distintas categorías, sean ciudades, productos etc



* boxplot con puntos

# un gráfico o diagrama de cajas es un tipo de una gráfica que sirve para estudiar la distribución de una variable con respecto a distintas categorías

# vamos a estudiar la distribucion de ese ancho con respecto a cada una de las 3 variables

ggplot(data = iris,

mapping = aes(x = Species,

y = Petal.Width,

# relleno para pintar de color diferente las distintas especies

fill = Species)) +

# para hacer el boxplot, agregamos una geometría con este mismo nombre

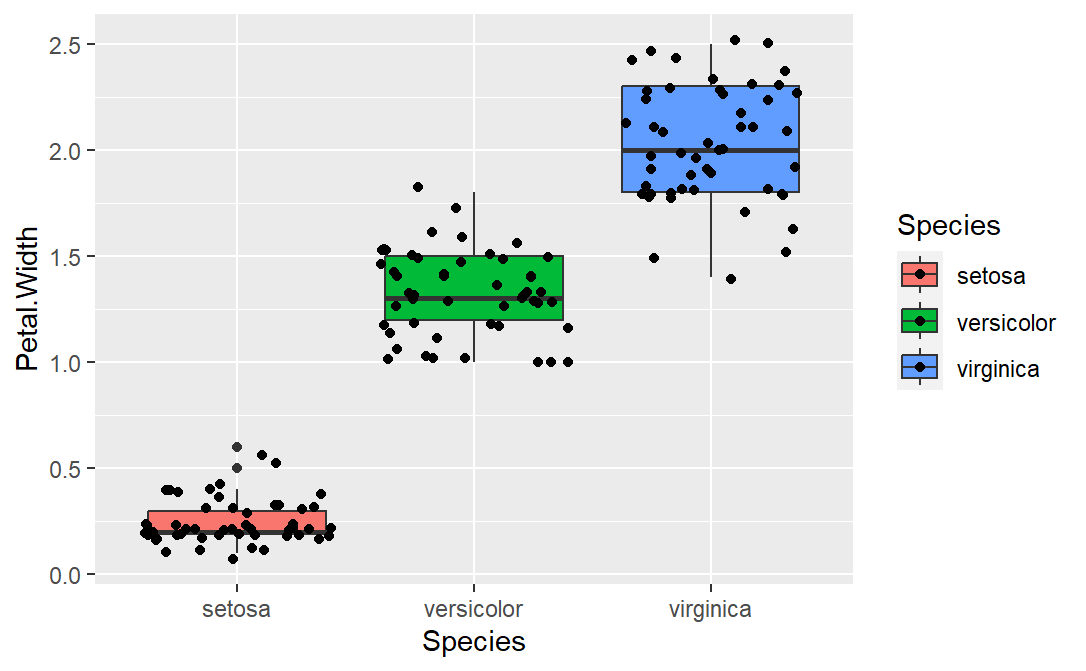
geom\_boxplot() +

# esta geometria me va a colocar dentro de las cajas los puntos del ancho de los pétalos, lo que nos ayuda a entender la mediana

geom\_jitter()

# Lo anterior nos indica que con la suma de comandos podemos hacer gráficas e ir superponiendo puntos y gráficas de cajas en la misma

# si queremos interpretar este gráfico, nos diría que la Setosa, tiene los gráficos más pequeños, mientras que la virginica tiene los datos más grandes, la línea significa la mediana, es decir que de la línea para arriba, está la mitad de todos los datos y de ella para abajo, la otra mitad, dentro de la caja estarian todos los datos y la línea vertical sería, el rango de lo más o menos normal que esperaríamos encontrar, todos los puntos fuera de esta línea, pueden ser considerados outlayers, ejemplo en el casoo de la Setosa



Tidyverse

Objetivo

Presentar Tidyverse para ciencia de datos

Vamos a

1. Qué es Tidyverse?

2. Ventajas/desventajas?

3. 8 paquetes principales de Tidyverse

4. En qué orden aprenderlos?

* ¿Qué es tidyverse?

Conjunto de paquetes en R diseñados para Data Science, sirve para Importar, transformar, visualizar, modelar y comunicar

Características

Todos comparten nombres y estructuras comunes, ejemplo, todos los nombres en minúscula o barra baja para llamar a las funciones, esto mantiene una consistencia a lo largo de todos sus paquetes que permite que sea más fácil utilizarlos, estos paquetes son mantenidos por la comunidad y RStudio

* Ventajas

Odiados por unos, amado por otros en el mundo de R

Facilita el análisis y manipulación de datos

Más rápido

* Desventajas

Deja de lado la forma usual de programación en R

Utiliza pipes ||, que es para conectar varias funciones

* Instalar Tidyverse

install.package('tidyverse')

library(tiydverse)

* Básico Intermedio

ggplot2 purrr

dplyr tibble

tidyr stringr

readr forcats

1. ggplot2

Es un paquete para visualización que utiliza la gramática de los gráficos



1. dplyr

Ayuda con problemas comunes de manipulación de datos en un lenguaje basado en acciones sobre los datos

Verbos dplyr permite

* + mutate() mutar o crear nuevas variables
  + select() seleccionar
  + filter() filtrar
  + summarise() sumarizar
  + arrange() acomodar
  + group\_by() agrupar



1. tidyr

Ayuda a transformar los datos para hacerlos más eficientes o "tidy", quiere decir que podemos transformar las filas y columnas de una forma eficiente para que los datos cumplan con las 3 condiciones:

* Cada columna es una variable
* Cada fila una observación
* Cada celda un valor

Esto ayuda mucho a la hora de modelar y graficar los datos



1. readr

Permite leer de manera amigable y rápida archivos de texto plano (CSV) y otros tipos de archivos



1. purrr (avanzado)

Facilita el trabajo con vectores y funciones en un lenguaje consistente



1. tibble

Es una reinvención del dataframe, haciendo más eficiente algunas rutinas

Lo hemos utilizado, cargando paquetes de reader y readerexcel, la ventaja de usarlo, es que aprovecha lo bueno de los dataframe y mejora algunas opciones obsoletas en versiones anteriores

Un dibujo de una caja

Descripción generada automáticamente con confianza baja

1. stringr

Paquete para trabajar con análisis de textos y manipulación de strings, en casos concretos donde tenemos cantidad de texto y queremos buscar cierta información o cuando queremos buscar dentro de la URL de las paginas y texto de HTML



1. forcats

Es un paquete especial para lidiar con factores y datos categóricos

Otros paquetes:

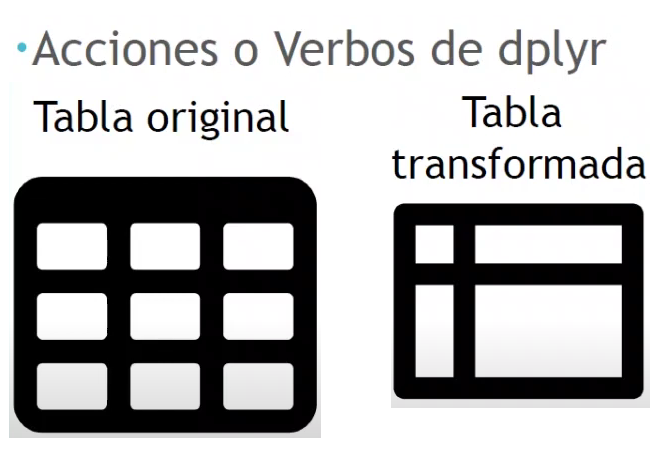
* Para leer datos
* readxl (Excel)
* haven (SPSS, Stata, SAS)
* Manipulación de datos
* lubridate (fechas y tiempos)
* hms (para horas, minutos, segundos)
* blob (para datos binarios)

Nota:

* Empieza por lo básico

ggplot2, dplyr, tidyr, readr.

* Aprende el resto, solo cuando sea necesario.

**Manipular datos con dplyr**

Objetivo

Dominar aspectos básicos de dplyr

Vamos a

1. Qué es dplyr?

* Verbos para manipular datos
* Cómo usar el operador pipe %>%?

1. Cómo filtrar datos
2. Cómo resumir datos agrupados

**Acciones o Verbos de dplyr**

* mutate() := crea nuevas variables
* select() := toma variables según nombre
* filter() := toma variables según condición
* summarise() := reduce muchos valores en un único resumen
* arrange() := cambia el orden de los datos
* group\_by() := operaciones por grupos

Cómo funcionan los verbos?

* filter(data, var1 == 'condición')
* se puede utilizar %>%

data %>%

filter(var1 == 'condición')

1. cargar datos de gapminder

* instalando paquete con los datos

install.packages("gapminder")

* cargando paquete con los datos

library(gapminder)

* cargando datos a entorno

data("gapminder")

* cargando datos a entorno

head(gapminder)

1. filtrar datos con dplyr

* Instalar paquetes necesarios

install.packages("tidyverse")

library(tidyverse)

* filtrar datos por pais sin %>%

filter(gapminder, country == 'Mexico')

* filtrar datos por pais

# para hacer %>% en RStudio (cntrl + shift + M)

gapminder %>%

filter(country == 'Mexico')

* filtrar datos por año

gapminder %>%

filter(year == 2002)

* filtrar paises con esperanza de vida menor o igual a 40 y el año en 2002

gapminder %>%

filter(lifeExp <= 40,

year == 2002)

1. hacer resumenes de datos

* cantidad de paises en Asia

# Traemos todos los datos que cumplan las dos condiciones

gapminder %>%

filter(continent == 'Asia',

year == 2007) %>%

# creamos la nueva variable conteo y asignamos la funcion de conteo

summarise(conteo = n())

* maxima esparanza de vida

# traemos todos los datos sin filtrar

gapminder %>%

# creamos una variable max\_lifeExp, y traemos el mayor de la experanza de vida

summarise(max\_lifeExp = max(lifeExp))

* agrupando esperanza de vida promedio por año

# separa por partes todo el conjunto de datos por año y a cada año, va a tomar la estadística o el resumen

gapminder %>%

# Agrupamos segun el año

group\_by(year) %>%

# volvemos a aplicar el operador %>% y ahora creamos la variable prom\_vida, vamos a calcular el promedio de la esperanza de vida

summarise(prom\_vida = mean(lifeExp))

**Análisis exploratorio de datos en R (EDA)**

Objetivo

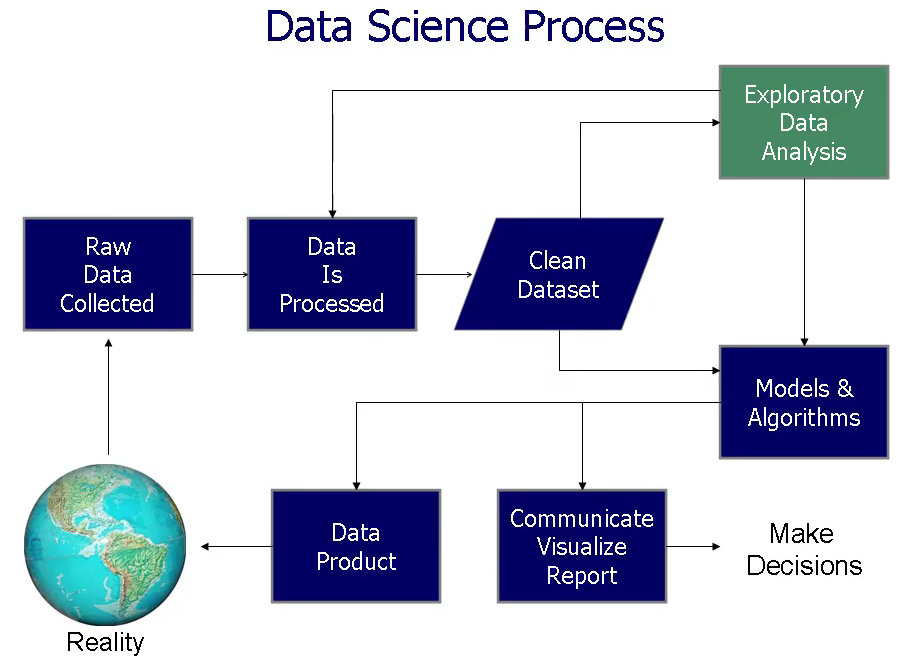
Introducir EDA o análisis exploratorio de datos

Vamos a ver

1. EDA en el proceso Data Science
2. Fases en la exploración de datos
3. Técnicas de EDA

* Resúmenes estadísticos
* Distribuciones de frecuencia
* Percentiles y Boxplots
* Correlaciones y dispersión

EDA en el proceso de Data Science



El mundo son los datos

Raw Data Collected: Recolección a través de diferentes mecanismos o lenguajes de programación.

Data in collected: los datos van a ser procesados y recolectados

Clean Dataset: para pasar por un proceso de limpieza

Exploratory Data Analysis: pasan a un proceso de análisis exploratorio de datos

Models & Algorithms: o a los módelos estadísticos, dependiendo del caso que estemos trabajando

Communicate Visualize Report: puede darse el caso que del análisis y exploración pasemos a la comunicación y visualización de los datos

La parte que le toca a R en todo este proceso sería un poco la parte de limpieza de datos, análisis exploratorio de datos y los modelos, la parte de comunicación, visualización y reporte se puede ver un poco a través de las gráficas

Proceso de análisis exploratorio de datos

Siempre la idea va a ser sacar conclusiones para tomar decisiones, en el caso de una empresa puede ser para reducir sus costos o aumentar sus ganancias o en el caso científico y de investigación, puede ser, extraer información importante sobre preguntas para el avance de la ciencia como en biología, astronomía, física, etc.

En si, la idea de este proceso es:

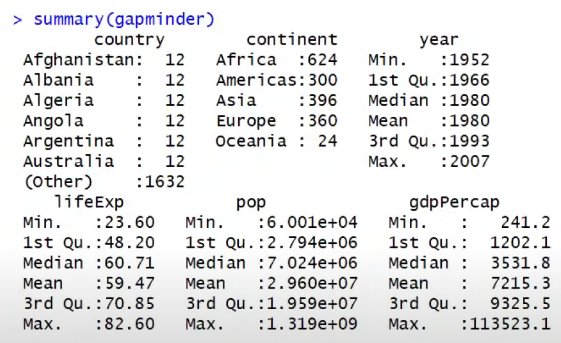
1. Generar preguntas de interés, hipótesis, antes de saltar a explorar los datos.
2. EDA.
3. Dar conclusiones a partir de la exploración que hemos hecho.

Esquema general:



La exploración la podemos separarla en tres grandes fases:

1. Transformar y limpiar datos
2. Generar tablas resumen
3. Crear gráficos explicativos



**Algunas técnicas de EDA**

1. Resúmenes estadísticos
2. Distribuciones de frecuencia
3. Percentiles y Boxplots
4. Correlaciones y dispersión

Checklist para tus análisis de datos

Gráfico, Gráfico de dispersión

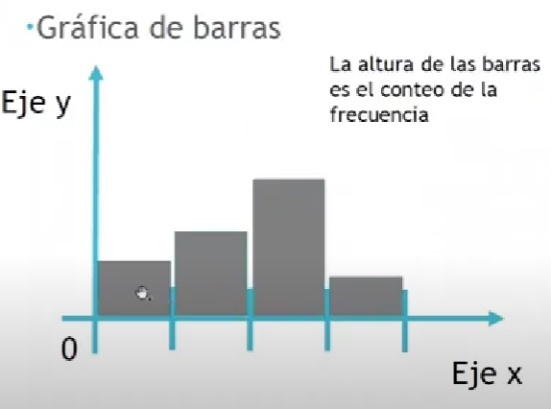
Descripción generada automáticamente

1. **Estadísticas resumen**

* Conteo o número de mediciones
* Media y Mediana
* Error o desviación típica/estándar
* Mínimo y Máximo

1. **Distribuciones de frecuencia**

**Histograma**

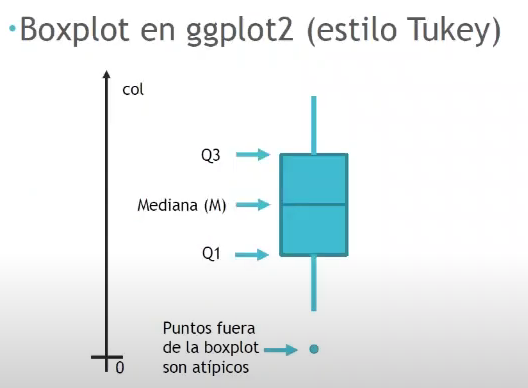
* Gráfica de barras con la distribución de la frecuencia.
* La frecuencia es la cantidad de veces que se repite una cosa.

La gráfica de barra separa el eje x en diferentes intervalos de clase o bins y para cada uno cuenta la cantidad de observaciones que tiene, luego hace una gráfica de barras donde la altura es el conteo de esta frecuencia.

En R podemos hacer histogramas con la función hist(), o si utilizamos la geometría de ggplot2, podemos usar:

geom\_hist()

geom\_density(): estudia la distribución continua de la densidad que es una especie de versión continua del histograma.

1. Percentil y boxplot

Se utilizan para medir la distribución acumulada, estas medidas se agrupan en 3 grandes grupos:

Mediana: Divide a la distribución en dos, es decir, mide hasta qué numero de la distribución, se acumula el 50% de los datos

Cuartil: Divide a la distribución en cuatro: Q1, Q2 y Q3 corresponden al 25%, 50% y 75%

Percentil: Tiene una función similar, pero dividen a la distribución en cien partes iguales

Todo esto de los percentiles es útil para estudiarlos a través de la boxplot o gráficos de cajas y bigotes como la figura, es importante destacar la mediana, que sería la línea que hay entre las dos cajas y diría hasta que punto de toda la columna o de toda la variable se acumula el 50% de mínimo a máximo, además, las otras dos partes de la caja miden Q1 y Q3 que serían los cuartiles, lo que serían la línea vertical o bigotes, se calculan a partir de una proporción a partir de una proporción de la distancia que hay entre Q1 y Q2 que también se le llama rango intercuartílico, esto se usa es para detectar datos que aunque no están tan centrados también se podrían considerar como datos usuales o normales de esta distribución, todos los puntos que caigan fuera del dibujo o la caja de bigotes si se considerarían datos atípicos.

En R, podemos hacer boxplot, utilizando la función boxplot() y también, podemos utilizar distintas geometrías de ggplot como:

geom\_boxplot()

geom\_violin()

* **La correlación es asociación o dependencia**

Puede ser causal o no (correlación no implica causalidad), lo que significa es que si bien la dependencia significa que hay una asociación entre variables, no siempre hay una causa y efecto, el ejemplo de esto es el reloj, yo tengo un reloj que cuando las manecillas marcan las 6 de la tarte, suenan las campanas que están en frente de mi casa y esto ocurre todos los días, es decir que hay una dependencia o sea que hay una asociación en que el reloj marque las 6 y suenen las campanas, sin embargo, el reloj no hace que suenen las campanas ni la campana hace que el reloj marque las 6, son fenómenos aislados que ocurren al mismo tiempo pero no son causales.

Para saber más, ver esta página: [www.tylervigen.com/spurious-correlations](http://www.tylervigen.com/spurious-correlations).

* De todos los tipos de correlación que existe la Correlación lineal, es la más sencilla de ver y esta puede indicar una relación predecible con un modelo lineal detrás.

**¿Cómo identificar correlación?**

* Con tablas:

Matriz de correlación

* Con gráficas:

Gráficas de puntos a pares

Gráficas de matriz de correlación

* En R utilizamos la función

pairs(): hace la grafica de todas las variables continuas contra todas las variables continuas para una primera aproximación

corr(): calcula la correlación estadística entre cada una de las variables

* Geometrías ggplot

geom\_point() : grafica de puntos

geom\_smooth() : para trazar una línea

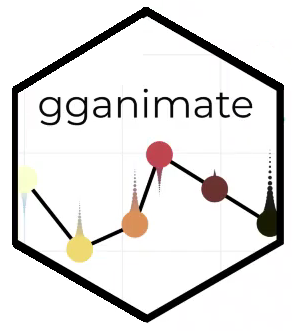
geom\_point() + geom\_smooth(()

* Paquete corrplot

corrplot() : sirve para graficar la matriz de correlación que resulta de la función corr()

* Paquete PerformanceAnalytics

chart. .Correlation() : genera estadística como histogramas, puntos, líneas y la correlación

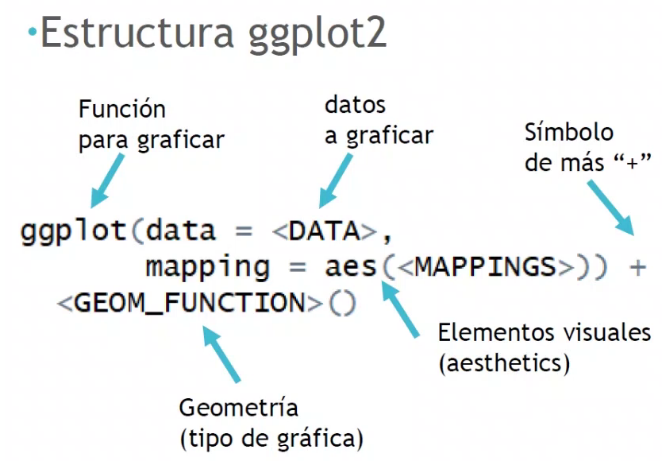


Gráficos animados

Objetivo

Introducir el paquete gganimate en R

Vamos a

1. Qué hace este paquete?

2. Cómo se relaciona con ggplot2?

3. Ejemplo en R

¿Qué hace el paquete gganimate?

* Extiende la gramática de los gráficos

Añade capacidad de animación

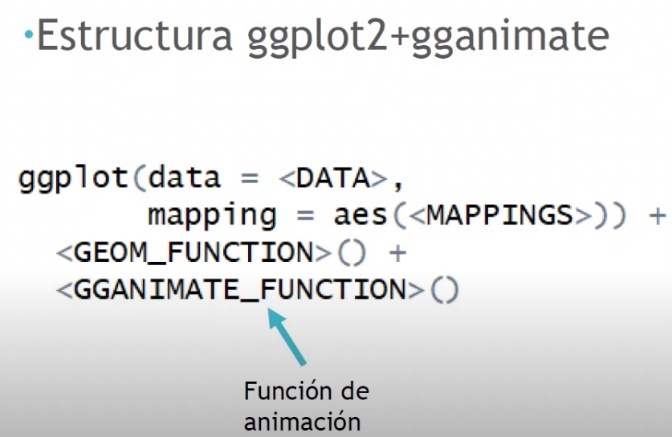
* Tiene nuevas capas para las gráficas

Pueden ser sumadas como ggplot

Personaliza cómo cambia la visualización en el tiempo

(De mis paquetes favoritos)

Para empezar a crear la animación:



* Funciones de animación

transition\_\*() : Transiciones sirven para explicarle a R como hacer los cambios en el tiempo de la visualización

view\_\*()

shadow\_\*()

enter\_\*() / exit\_0)

ease\_aes()

* instalando paquete para animar

#install.packages('gganimate')

#library(ggplot2)

library (gganimate)

#install.packages('gifski')

library(gifski)

* instalando paquete con los datos

#install.packages("gapminder")

* cargando paquete con los datos

library(gapminder)

* cargando datos a entorno

data("gapminder")

* mostrar datos

head(gapminder)

* grafica básica

# Grfica de puntos que muestre la esperanza de vida a lo largo de los años

gapminder %>%

group\_by(year, continent) %>%

# creamos una variable mean\_life que tenga la media de la esperanza de vida

summarize(mean\_life = mean(lifeExp)) %>%

# Elemenros esteticos

ggplot(aes(x = year,

y = mean\_life,

color = continent)) +

geom\_line()

* Animando la grafica

# Grfica de puntos que muestre la esperanza de vida a lo largo de los años

gapminder %>%

group\_by(year, continent) %>%

# creamos una variable mean\_life que tenga la media de la esperanza de vida

summarize(mean\_life = mean(lifeExp)) %>%

# Elemenros esteticos

ggplot(aes(x = year,

y = mean\_life,

color = continent)) +

geom\_line() +

# gganimate, grafica con movimiento, creciendo a lo largo de los años

transition\_reveal(year)

* Mejorando animacion

# Grfica de puntos que muestre la esperanza de vida a lo largo de los años

gapminder %>%

group\_by(year, continent) %>%

# creamos una variable mean\_life que tenga la media de la esperanza de vida

summarize(mean\_life = mean(lifeExp)) %>%

# Elemenros esteticos

ggplot(aes(x = year,

y = mean\_life,

color = continent)) +

# Aumentamos el tamaño

geom\_line(size = 2) +

# colocar lineas y puntos al mismo tiempo

geom\_point(size = 4) +

# se escriben etiquetas en otra capa, frame\_along hace que el año cambien despues de end

labs(title = 'Esperanza de vida en {frame\_along}',

x = 'Fecha',

y = 'Años de Vida') +

theme\_minimal() +

# gganimate, grafica con movimiento, creciendo a lo largo de los años

transition\_reveal(year)

* Varias graficas

# una grafica por continente, viene bien para construir varias graficas de una sola vez

gapminder %>%

filter(year == '2007') %>%

ggplot(aes (x = gdpPercap,

y = lifeExp,

color = continent)) +

geom\_point() +

 facet\_wrap(~ continent)

cuando se usa paréntesis y corchete, como lo diferencio?

Paréntesis: utilizar con funciones, se escribe la función y entre paréntesis los argumentos de las funciones.

Corchete: son para las estructuras de datos como cuando trabajamos con vectores, matrices o dataframe, si queremos seleccionar elementos específicos como conjunto o sub conjunto lo indicamos entre corchetes

Modelos en R

Objetivo

Introducir modelos de machine learning en R

Vamos a ver

1. ¿Qué es un modelo? ¿Para qué sirve?

2. Modelos en el proceso DataScience

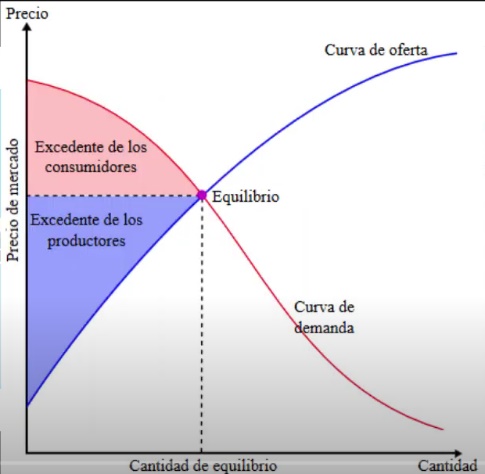
3. Ejemplo de predicción en R

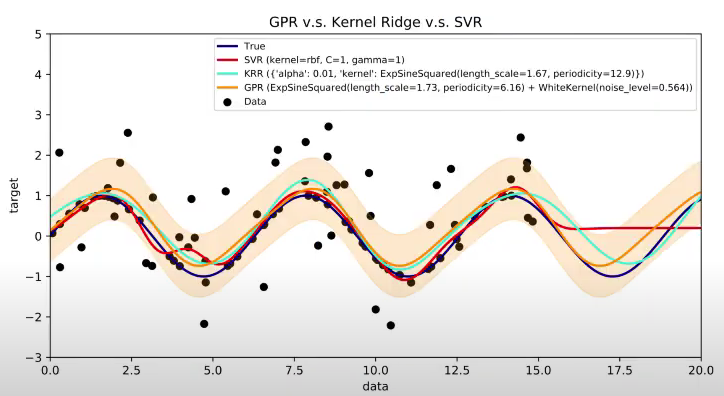
¿Qué es un modelo?: es una versión simple de la realidad, la realidad es algo que no se puede entender en su totalidad por nosotros.

"Todos los modelos están mal, pero algunos son útiles" - George Box

Ejemplos

* Modelos Físicos: Ejemplo teorías de newton y de ainstain
* Modelos de belleza: algún aspecto simplificado, donde consideramos la belleza física como atributo
* Modelos en arquitectura: pequeñas maquetas de los edificios que se quieren construir para prevenir riesgos o ver si es viable el proyecto
* Modelos económicos: algunos modelos como la ley de oferta y demanda pueden darnos una mayor intuición sobre como se comportan los precios en un mercado

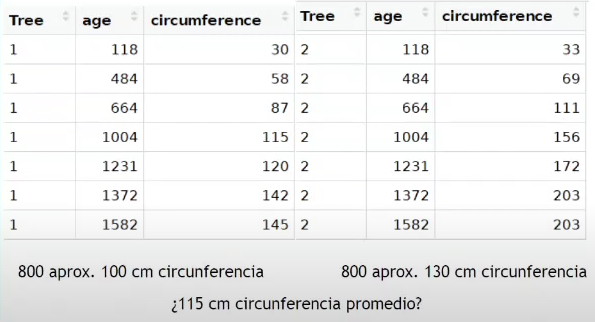


* Modelos de Machine Leaning: Resolver problemas de machine learning
* Predicción o regresión que consiste en estimar un valor numérico dada información previa, un ejemplo son los valores de las casas, basados en precios de otros inmuebles similares
* Clasificación: es similar a la regresión, solo que en este caso, buscaríamos etiquetar o clasificar en diferentes categorías a las observaciones, un caso muy popular, son los datos del titanic en lo que buscamos es clasificar un pasajero para saber si este sobrevivirá o no basándose en sus características
* Clustering o análisis de conglomerados: requiere etiquetar a las observaciones sin información previa de a que grupo pertenece un ejemplo es cuando hacemos análisis y segmentación de mercados en donde tenemos diferentes clientes y queremos agruparlos según intereses para después venderle algo
* Algunos modelos
* Predicción: Regresión lineal, multiple, no lineal...
* Clasificación: Árboles de decisión, KNN, logística...
* Clustering: K-medias o K-means, jerárquico...

Ejemplo de predicción en R

Situación: Árboles de naranja

* Es de interés predecir la medida de la circunferencia
* Datos de medida por días
* Problema/pregunta de interés

¿Cuál será la medida de un árbol de naranja a los 800 días de sembrado?

* **Regresión lineal simple**

Busca utilizar una recta para ajustar los datos y así encontrar el mejor valor posible basándose en la información que tenemos.

**Ecuación: medida = a \* días + b**

* **Esto es la ecuación más simple de la recta**

**Donde**

**a = la pendiente**

**b = intercepto**

**Medida = circunferencia**

* Ejemplos

medida = días (a=1,b=0)

medida = 0.5\*días (a = 0.5, b = 0)

medida = 0.1\*dias + 10 (a =0.5, b = 10)

Muchas posibilidades para a y para b.

¿Cuál sería la "mejor"?

Para resolver esto con exactitud, debemos buscar los mínimos cuadrados y ver como afecta a la ecuación lineal.

Con la regresión y regresión lineal se espera encontrar los valores óptimos para los parámetros del intercepto y la pendiente

La importancia de utilizar un modelo, solo con una ecuación, podemos calcular la medida que tiene el árbol a los 800 días en el caso del ejercicio de la referencia.

* Cargando paquete para analizar datos

library(tidyverse)

* Cargando datos a Entorno

data("Orange")

* Mostrando datos de entorno

head(Orange)

* Problema/pregunta

# Cuanto medirá la circunferencia en promedio de un árbol de naránjas a los 800 días de plantarlo

# 1 Grafica

Orange %>%

# en x = los días y en y = la circunferencia

ggplot(aes(x = age,

y = circumference)) +

geom\_point() +

# esto nos muestra que a medida que pasan los días, crece el tamaño de los árboles

# lo que se busca con la regrecion lineal es trazar una recta que pase por la mitad de los puntos, de tal forma que para el día 800 encontrar cual va a ser el valor

# es la b = al intercepto, cuando a = 1 o 0.1 b = y cuando a = 0.1 b = 10

geom\_abline(intercept = 10,

# es la a = a la pendiente, ejemplos 1, 0.5 y 0.1

slope = 0.1,

col = 'blue')

* # Mejor ajuste de regresión lineal simple, lm() encuentra el mejor valor para la pendiente y el intercepto

# la formula seria circumference con respecto a age

lm(circumference ~ age, data = Orange)

# 1 Grafica

Orange %>%

# en x = los días y en y = la circunferencia

ggplot(aes(x = age,

y = circumference)) +

geom\_point() +

# esto nos muestra que a medida que pasan los días, crece el tamaño de los árboles

# lo que se busca con la regrecion lineal es trazar una recta que pase por la mitad de los puntos, de tal forma que para el día 800 encontrar cual va a ser el valor

# es la b = al intercepto, cuando a = 1 o 0.1 b = y cuando a = 0.1 b = 10

geom\_abline(intercept = 17.3997,

# es la a = a la pendiente, ejemplos 1, 0.5 y 0.1

slope = 0.1068,

col = 'blue') +

# para responder la pregunta, debemos trazar una línea vertical en el 800

geom\_vline(xintercept = 800,

col = 'red')

* # Valor exacto:

Cuanto medirá la circunferencia en promedio de un árbol de naránjas a los 800 días de plantarlo

dias <- 800

# medida = a \* dias + b

medida <- 0.1068 \* 800 + 17.3997

medida

CLASIFICACION EN R

Objetivo

Clasificar datos del Titanic con árboles

Vamos a ver

1. El problema de clasificación?

2. Datos del Titanic

3. Modelo con árboles de decisión

* Problema
* Queremos asignar una clase o categoría a observaciones basados en variables independientes (predictores, entrada)
* Basado en observaciones previas



* **Ejemplo**
* Pasajeros del Titanic: edad y género
* Como respuesta nos interesa, estudiar supervivencia con datos en R
* **Estás interesado en:**
* Variables que más influyen
* ¿Un pasajero de 4 años de edad con género masculino sobreviviría?
* **Una solución:**
* ¡Árboles de decisión!

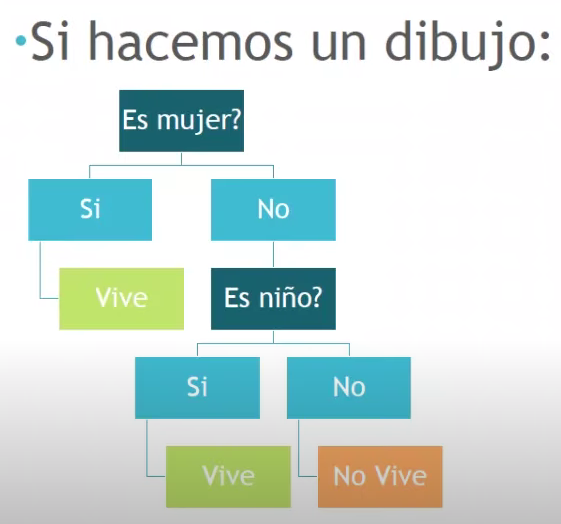
El juego de las preguntas

¿Cuál era su edad?

¿Su género?

¿Era de 1era clase o de 3era clase?

Si viste la película recordaras que decían algo como, salvar primero a mujeres y niños



paquete de datos

```{r}

#install.packages('titanic')

```

cargar librarias de analisis

```{r}

library(tidyverse)

```

cargar librerias de datos

```{r}

library(titanic)

```

cargar datos

```{r}

data ("titanic\_train")

```

visualizar datos

```{r}

head(titanic\_train)

```

cargar librerias para clasificacion

hace los calculos

```{r}

#install.packages('rpart')

library(rpart)

```

dar forma a la gráfica

```{r}

#install.packages('rattle')

library(rattle)

```

dar forma a la grafica

```{r}

#install.packages('rpart.plot')

library(rpart.plot)

```

modelando con arboles de decision

```{r}

arbol <- rpart(

# Modelo que queremos entrenar

formula = Survived ~ Sex + Age,

# Datos donde tiene que buscar la información

data = titanic\_train,

# Para que prediga sobre la clase

method = 'class')

```

graficar el arbol

```{r}

fancyRpartPlot(arbol)

```

prediciendo con el arbol

```{r}

# Predecir sobre los datos con los que se hizo el modelo

# funcion para predecir o clasificar predict(arbol, type = 'class') para calcular el 0 o 1 sobre todos los datos

pred\_arbol <- predict(arbol, type = 'class')

```

crear dataframe con la prediccion

```{r}

# combina la estructura de datos por columna, que tenga la misma cantidad de filas, como argumento se coloca la data original, junto con pred\_arbol que es la variable con la funcion de predicción o clasificacion

titanic\_pred <- cbind(titanic\_train)

```

¿un pasajero masculino de 4 años de edad sobreviviría?

```{r}

predict(object = arbol,

newdata = data\_frame(Age = 4,

Sex = 'male'),

# Calcular el 1 o el 0

type = 'class')

```

Series de tiempo en R

Objetivo

Introducir las series de tiempo en R

Vamos a ver

1. Pronóstico con series de tiempo

2. Ejemplo con datos de ecobici-CDMX

3. Modelo ARIMA estacional en R

* El objetivo es pronosticar o saber que va a pasar en el futuro

Estimar, predecir o pronosticar Cantidades a futuro

Ejemplo: Temperatura, precios de acciones

Nos basamos en el pasado

Generamos un modelo estadístico que toma como primicia que el futuro se puede predecir a través del pasado, un pronóstico de series de tiempo lo que nos da es una idea de lo que puede pasar después, si sigue pasando lo que ha venido ocurriendo.

Datos:

* Sistema de bicicletas públicas
* Utilizado por turistas y habitantes
* Desde 2010 con más de 170mil usuarios
* Más de 6 800 bicicletas y 100 estaciones

Lo que vamos a estudiar es ¿Cuántos usuarios por hora existen?

Ejemplo, vamos a usar los datos de tres días y vamos a predecir lo que sucedería, los tres días siguientes

Ir a la web oficial:

<https://www.ecobici.cdmx.gob.mx/es/informacion-del-servicio/open-data>

Nos interesan viajes por hora

Vamos a trabajar con fechas

Última semana de febrero

Conteos por horas

Autoregresivo integrado de media móvil estacional o arima estacional o sarima

Tiene varias partes unidas y así puede hacer una mejor predicción o pronostico del futuro

AR: "regresión" sobre si mismo, predice lo que va a pasar en el futuro, usando como predictor lo que pasó en el pasado

I: diferencias, que dice cuanto cambia con respecto al pasado

MA: tiene que ver con los "errores" modelos de media móvil y tiene que ver que en las series de tiempo, los comportamientos de errores es bastante específico y particular, tiene que ver con algo que se llama ruido blanco o White noise

S: fenómenos que se repiten, en este caso se trabaja con los días de la semana o días laborales

Parámetros (p,d,q)(Ps) ds, qs) el orden que va a tener cada uno de los elementos y como se unen

P = orden del ar 1, 2 o 3 y si es de orden tres el modelo ar tendrá tres parámetros

D = es la resta de las diferencias

Q = orden o niveles de parámetros de media móvil

En esta segunda parte se considera la estacionalidad, en este caso como son días, vamos a tener una estacionalidad de 24, entonces sería un modelo autoregresivo considerando la estacionalidad de 24 diferencias y la s es para recordar la estacionalidad en el caso que exista

Parámetros (p,d,q)(ps, ds, qs)[S]

[S] 24 si son horas, 7 si son los días de la semanas, 12 si son meses del año.

código para ecobici en R

```{r}

# analizar datos, sirve para Importar, transformar, visualizar, modelar y comunicar

library(tidyverse)

# Especial para todo lo que tiene que ver con fecha en R, muy bueno para trabajar series de tiempo de forma conjunta

#install.packages('libridate')

library(lubridate)

```

# leer CSV

```{r}

file.choose()

```

copiamos la ruta en la variable raw\_data y nos queda estilo dataframe

```{r}

raw\_data <- read.csv("D:\\Users\\diana\\Downloads\\DATA MINING\\R-STUDIO\\PRACTICA\\2020-02.csv")

```

cargamos datos

```{r}

head(raw\_data)

```

procesamiento de datos

```{r}

viajes\_diarios <- raw\_data %>%

# Transformamos la variable raw\_data, creando una variable que se llame fecha\_hora

# organiza el formato de fecha, primero Fecha\_Retiro formato dmy y posterior Hora\_Retiro hms

mutate(fecha\_hora = dmy\_hms(paste(Fecha\_Retiro, Hora\_Retiro))) %>%

# Filtrar para fecha\_hora datos mayores al 24 de febrero

filter(fecha\_hora >= as.Date('2020-02-24'),

fecha\_hora <= as.Date('2020-02-27')) %>%

group\_by(horas = floor\_date(fecha\_hora, unit = 'hour')) %>%

# n() Cuenta los registros por hora

summarise(conteo = n())

```

rellenando ceros

```{r}

horas\_completas <- data.frame(

# creo un dataframe con la variable horas, la creo con una secuencia que inivia con floor\_date del minimo de los viajes diarios en las horas con la unidad en horas y termina en floor\_date del maximo de los viajes diarios en las horas con unidad de hora

horas = seq(

floor\_date(min(viajes\_diarios$horas), unit = 'hour'),

floor\_date(max(viajes\_diarios$horas), unit = 'hour'),

by = 'hour'

))

```

left join para unir viajes con horas

```{r}

viajes\_hora <- horas\_completas %>%

group\_by(horas\_redondeadas = floor\_date(horas, unit = 'hour')) %>%

# se unen los dataframes por las horas

left\_join(viajes\_diarios) %>%

# para agregar los ceros, vamos a mutar la variable conteo, le agregamos la función ifelse, asigna una condicion a cumplir y asignar valores

mutate(conteo = ifelse(is.na(conteo), 0, conteo))

# lo anterior significa que si cuando hemos unido a traves de left join hay un na, agregue cero y de lo contrario agregue el valor

```

grafica de líneas inicial

```{r}

ggplot(data = viajes\_diarios,

aes(x = horas,

y = conteo)) +

# agregamos una capa de geom\_line

geom\_line()

```

creando objeto ts para el modelo

```{r}

# metemos los conteos en una variable (conteo de series de tiempo)

conteos\_ts <- ts(viajes\_hora$conteo,

# asignamos un inicio, en este caso es 1

start = 1,

frequency = 24)

```

visualizar datos conteo\_ts

```{r}

conteos\_ts

```

hacer modelo ARIMA

```{r}

# uno de los mejores paquetes para hacer series de tiempo

#install.packages('forecast')

```

cargar la libreria

```{r}

library(forecast)

```

ajuste del modelo

```{r}

# esta funcion lo que hace es calcular todos los parámetros de la arima

ajuste <- auto.arima(y = conteos\_ts)

```

resumen del ajuste

```{r}

summary(ajuste)

```

hacer las predicciones

```{r}

predicciones <- forecast(ajuste)

```

calcular el valor minimo de las predicciones

```{r}

# como es una lista colocamos lower

min(predicciones[['lower']])

```

calculamos el valor maximo de las predicciones

```{r}

# como es una lista colocamos lower

max(predicciones[['upper']])

```

grafica de las predicciones

```{r}

# autoplot viene en forcast, automáticamente calcula las barras de confianza y lo demás

p\_predict <- autoplot(predicciones)

```

ver la grafica

```{r}

# la linea azul es el valor promedio estimado que es cuanto deberia valer el conteo o cuantas bicicletas estan en circulacion para cierta hora del dia, captura o repite muy bien el patron de la antiguedad, la banda distorcionada, sería como un poco el error o la banda de confianza, para el ultimo día la banda es mas ancha, significa que misntras mas queramos predecir hacia el futuro, mas grande será el margen de error

p\_predict

```

graficando toda la semana

procesamiento de datos

```{r}

viajes\_diarios <- raw\_data %>%

# Transformamos la variable raw\_data, creando una variable que se llame fecha\_hora

# organiza el formato de fecha, primero Fecha\_Retiro formato dmy y posterior Hora\_Retiro hms

mutate(fecha\_hora = dmy\_hms(paste(Fecha\_Retiro, Hora\_Retiro))) %>%

# Filtrar para fecha\_hora datos mayores al 24 de febrero

filter(fecha\_hora >= as.Date('2020-02-24'),

fecha\_hora <= as.Date('2020-02-29')) %>%

group\_by(horas = floor\_date(fecha\_hora, unit = 'hour')) %>%

# n() Cuenta los registros por hora

summarise(conteo = n())

```

rellenando ceros

```{r}

horas\_completas <- data.frame(

# creo un dataframe con la variable horas, la creo con una secuencia que inivia con floor\_date del minimo de los viajes diarios en las horas con la unidad en horas y termina en floor\_date del maximo de los viajes diarios en las horas con unidad de hora

horas = seq(

floor\_date(min(viajes\_diarios$horas), unit = 'hour'),

floor\_date(max(viajes\_diarios$horas), unit = 'hour'),

by = 'hour'

))

```

left join para unir viajes con horas

```{r}

viajes\_hora <- horas\_completas %>%

group\_by(horas\_redondeadas = floor\_date(horas, unit = 'hour')) %>%

# se unen los dataframes por las horas

left\_join(viajes\_diarios) %>%

# para agregar los ceros, vamos a mutar la variable conteo, le agregamos la función ifelse, asigna una condicion a cumplir y asignar valores

mutate(conteo = ifelse(is.na(conteo), 0, conteo))

# lo anterior significa que si cuando hemos unido a traves de left join hay un na, agregue cero y de lo contrario agregue el valor

```

grafica de líneas inicial

```{r}

ggplot(data = viajes\_diarios,

aes(x = horas,

y = conteo)) +

# agregamos una capa de geom\_line

geom\_line()+

ylim(-551.5396, 4103.783) +

labs(title = 'Realidad')

# como al modelo noo había como explicarle lo ocurrido los viernes, no podia capturarlo y solo se basaba en lo que tenia, ya recalculando los datos con las predicciones, ahora si es capaz de mostrarlo

```

Recordar que la media es sensible a outlayers

Siempre es mejor calcular la mediana o la media truncada, donde saquemos esos outlayers por alguna razón

Distancia intercuartilica: es el 50% de los datos entre q1 y q3 o sea (q3-q1)

No hay una regla y hay que estudiarlo, se ponen valores para aleatorios

boxplot(X, ylim=c(0,100)

