Modelización Estadística: ¿Qué hacemos con una base de datos?

Introducción a R
Estadística Descriptiva

Mathias Bourel
mathias.bourel@fcea.edu.uy

Diciembre 2025

FCEA FACULTAD DE CIENCIAS ECONÓMICAS Y DE ADMINISTRACIÓN
IESTA INSTITUTO DE ESTADÍSTICA
UNIVERSIDAD DE LA REPÚBLICA URUGUAY
IRL-2030 IFUMI

Plan

1. ¿Qué es R?
2. Distribuciones
3. Variable cuantitativas
4. Variables cualitativas

¿Qué es R?

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.
- Es **gratuito** y **libre**.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.
- Es **gratuito** y **libre**.
- Funciona en Windows, Mac y Linux.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.
- Es **gratuito** y **libre**.
- Funciona en Windows, Mac y Linux.
- Dispone de numerosas herramientas para la visualización de datos.

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.
- Es **gratuito** y **libre**.
- Funciona en Windows, Mac y Linux.
- Dispone de numerosas herramientas para la visualización de datos.

Se puede descargar RStudio que tiene una interfase mucho más amigable en `https://posit.co/downloads/` o usar en la nube `https://posit.cloud`

¿Qué es R?

- R es uno de los software estadístico más potente y más ampliamente utilizado.
- Es un lenguaje y un entorno para el cálculo estadístico y la visualización gráfica.
- Es un conjunto integrado de herramientas para la manipulación de datos, el cálculo y la representación gráfica.
- Proporciona una amplia variedad de técnicas estadísticas clásicas y modernas.
- Es un proyecto colaborativo, donde todos los usuarios pueden subir sus propios códigos y paquetes.
- Es **gratuito** y **libre**.
- Funciona en Windows, Mac y Linux.
- Dispone de numerosas herramientas para la visualización de datos.

Se puede descargar RStudio que tiene una interfase mucho más amigable en `https://posit.co/downloads/` o usar en la nube `https://posit.cloud`

Existen muchos tutoriales disponibles: Introduction to R, R RefCard, etc.

Paquetes y librerías

R se basa en paquetes y librerías donde hay funciones, conjuntos de datos, documentaciones,...

- La lista de las librerias disponibles

```
library()
```

Paquetes y librerías

R se basa en paquetes y librerías donde hay funciones, conjuntos de datos, documentaciones,...

- La lista de las librerias disponibles

  `library()`

- Descargar e instalar un paquete desde el CRAN (Comprehensive R Archive Network)

  `install.packages (...)`

Paquetes y librerías

R se basa en paquetes y librerías donde hay funciones, conjuntos de datos, documentaciones,...

- La lista de las librerias disponibles

  ```
  library()
  ```

- Descargar e instalar un paquete desde el CRAN (Comprehensive R Archive Network)

  ```
  install.packages (...)
  ```

- Cargar paquetes

```
library(MASS) # funciones y bases del libro
 "Modern Applied Statistics with S" para hacer análisis discriminante,
 regresiones, multidimensional
library(rpart) #para hacer arboles de clasificación y regresión
library(e1071) #para Support Vector Machines
library(ks) #estimación de densidad por núcleo
```

Paquetes y librerías

R se basa en paquetes y librerías donde hay funciones, conjuntos de datos, documentaciones,...

- La lista de las librerias disponibles

  ```
  library()
  ```

- Descargar e instalar un paquete desde el CRAN (Comprehensive R Archive Network)

  ```
  install.packages (...)
  ```

- Cargar paquetes

  ```
  library(MASS) # funciones y bases del libro
   "Modern Applied Statistics with S" para hacer análisis discriminante,
   regresiones, multidimensional
  library(rpart) #para hacer arboles de clasificación y regresión
  library(e1071) #para Support Vector Machines
  library(ks) #estimación de densidad por núcleo
  ```

- Documentación sobre R

  ```
  help.start()
  ```

Paquetes y librerías

R se basa en paquetes y librerías donde hay funciones, conjuntos de datos, documentaciones,...

- La lista de las librerias disponibles

```
library()
```

- Descargar e instalar un paquete desde el CRAN (Comprehensive R Archive Network)

```
install.packages (...)
```

- Cargar paquetes

```
library(MASS) # funciones y bases del libro
 "Modern Applied Statistics with S" para hacer análisis discriminante,
 regresiones, multidimensional
library(rpart) #para hacer arboles de clasificación y regresión
library(e1071) #para Support Vector Machines
library(ks) #estimación de densidad por núcleo
```

- Documentación sobre R

```
help.start()
```

- Documentación sobre libreria

```
library(help = rpart)
```

Importación y exportación de bases de datos

- Ayuda sobre funciones:

```
help(mvrnorm) #ayuda para la función que genera datos normales multivariados
?lm #ayuda sobre la función lm para modelos lineales
```

Importación y exportación de bases de datos

- Ayuda sobre funciones:

```
help(mvrnorm) #ayuda para la función que genera datos normales multivariados
?lm #ayuda sobre la función lm para modelos lineales
```

- Cargar bases de datos:
  - Algunas están en R en las paquetes bases, por ejemplo iris, airquality, cars
  - Otras hay que cargarlas con por ejemplo estas lineas de código:

```
datos1=read.table("miejemplo.txt", sep= ",", header=T)
datos2=read.csv("miejemplo2.csv", sep= ",", header=T)
datos3=read.csv("/home/mathiasbourel/Escritorio/Investigacion/DATA/
Historica2009_2019.csv",header=T, sep=";")
base=read.csv("Desgargas/spam.csv",sep=",",header=T)
```

Importación y exportación de bases de datos

- Ayuda sobre funciones:

```
help(mvrnorm) #ayuda para la función que genera datos normales multivariados
?lm #ayuda sobre la función lm para modelos lineales
```

- Cargar bases de datos:
  - Algunas están en R en las paquetes bases, por ejemplo iris, airquality, cars
  - Otras hay que cargarlas con por ejemplo estas lineas de código:

```
datos1=read.table("miejemplo.txt", sep= ",", header=T)
datos2=read.csv("miejemplo2.csv", sep= ",", header=T)
datos3=read.csv("/home/mathiasbourel/Escritorio/Investigacion/DATA/
Historica2009_2019.csv",header=T, sep=";")
base=read.csv("Desgargas/spam.csv",sep=",",header=T)
```

- Escribir tablas y bases de datos

```
write.table()    write.csv()
```

Importación y exportación de bases de datos

- Ayuda sobre funciones:

```
help(mvrnorm) #ayuda para la función que genera datos normales multivariados
?lm #ayuda sobre la función lm para modelos lineales
```

- Cargar bases de datos:
  - Algunas están en R en las paquetes bases, por ejemplo iris, airquality, cars
  - Otras hay que cargarlas con por ejemplo estas lineas de código:

```
datos1=read.table("miejemplo.txt", sep= ",", header=T)
datos2=read.csv("miejemplo2.csv", sep= ",", header=T)
datos3=read.csv("/home/mathiasbourel/Escritorio/Investigacion/DATA/
Historica2009_2019.csv",header=T, sep=";")
base=read.csv("Desgargas/spam.csv",sep=",",header=T)
```

- Escribir tablas y bases de datos

```
write.table()     write.csv()
```

- Ejemplo completo

```
miejemplo <- data.frame(cbind(matrix(round(runif(24)*100,0),6,4)),
CAT=rep(c('A','B'),3))
attr(miejemplo,'names') <- paste('Var',1:5,sep='')
row.names(miejemplo) <- c('I1','I2','I3','I4','I5','I6')
miejemplo <- cbind(miejemplo,suma13=colSums(miejemplo[,1:3]),
logica2=miejemplo[,2]>50)
ej2 <- miejemplo[-1,-ncol(miejemplo)]; summary(ej2)
write.table(miejemplo,file='miejemplo.txt',sep='\t')
```

Posit Cheatseets

Las Posit Cheatsheets son guías rápidas (hojas resumen) creadas por Posit (antes RStudio) que sirven como material de referencia práctico para trabajar con R, RStudio y el ecosistema tidyverse, entre otras herramientas.

Posit Cheatseets

Las Posit Cheatsheets son guías rápidas (hojas resumen) creadas por Posit (antes RStudio) que sirven como material de referencia práctico para trabajar con R, RStudio y el ecosistema tidyverse, entre otras herramientas. ¿Para qué sirven?

- Recordar funciones clave y su sintaxis sin tener que ir a la documentación completa.
- Ver de un vistazo los flujos de trabajo más comunes.
- Aprender buenas prácticas de uso.
- Acompañar clases, talleres o trabajo cotidiano con R.

Posit Cheatseets

Las Posit Cheatsheets son guías rápidas (hojas resumen) creadas por Posit (antes RStudio) que sirven como material de referencia práctico para trabajar con R, RStudio y el ecosistema tidyverse, entre otras herramientas. ¿Para qué sirven?

- Recordar funciones clave y su sintaxis sin tener que ir a la documentación completa.
- Ver de un vistazo los flujos de trabajo más comunes.
- Aprender buenas prácticas de uso.
- Acompañar clases, talleres o trabajo cotidiano con R.

Cada cheatsheet suele incluir:

- Funciones más usadas (con ejemplos breves).
- Esquemas visuales (pipes, gramática de gráficos, manipulación de datos).
- Tablas resumen de argumentos importantes.
- Recordatorios de atajos y conceptos clave.

Posit Cheatseets

Las Posit Cheatsheets son guías rápidas (hojas resumen) creadas por Posit (antes RStudio) que sirven como material de referencia práctico para trabajar con R, RStudio y el ecosistema tidyverse, entre otras herramientas. ¿Para qué sirven?

- Recordar funciones clave y su sintaxis sin tener que ir a la documentación completa.
- Ver de un vistazo los flujos de trabajo más comunes.
- Aprender buenas prácticas de uso.
- Acompañar clases, talleres o trabajo cotidiano con R.

Cada cheatsheet suele incluir:

- Funciones más usadas (con ejemplos breves).
- Esquemas visuales (pipes, gramática de gráficos, manipulación de datos).
- Tablas resumen de argumentos importantes.
- Recordatorios de atajos y conceptos clave.

Se consiguen en `https://rstudio.github.io/cheatsheets/contributed-cheatsheets.html` y van algunas a continuación:

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)
Return x sorted.
rev(x)
Return x reversed.
table(x)
See counts of values.
unique(x)
See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/ write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Learn more at **web page or vignette** • package version • Updated: 3/15

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1', levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

Maths Functions

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`signif(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

`ls()`	List all variables in the environment.
`rm(x)`	Remove x from the environment.
`rm(list = ls())`	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

Matrices

`m <- matrix(x, nrow = 3, ncol = 3)`
Create a matrix from x.

`m[2, ]` - Select a row

`m[ , 1]` - Select a column

`m[2, 3]` - Select an element

`t(m)` Transpose

`m %*% n` Matrix Multiplication

`solve(m, n)` Find x in: m * x = n

Lists

`l <- list(x = 1:5, y = c('a', 'b'))`
A list is a collection of elements which can be of different types.

`l[[2]]`	`l[1]`	`l$x`	`l['y']`
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Also see the **dplyr** package.

Data Frames

`df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))`
A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

List subsetting

`df$x` `df[[2]]`

Understanding a data frame

`View(df)` See the full data frame.

`head(df)` See the first 6 rows.

Matrix subsetting

`df[ , 2]`

`df[2, ]`

`df[2, 2]`

`nrow(df)` Number of rows.

`ncol(df)` Number of columns.

`dim(df)` Number of columns and rows.

`cbind` - Bind columns.

`rbind` - Bind rows.

Strings

Also see the **stringr** package.

`paste(x, y, sep = ' ')`	Join multiple vectors together.
`paste(x, collapse = ' ')`	Join elements of a vector together.
`grep(pattern, x)`	Find regular expression matches in x.
`gsub(pattern, replace, x)`	Replace matches in x with a string.
`toupper(x)`	Convert to uppercase.
`tolower(x)`	Convert to lowercase.
`nchar(x)`	Number of characters in a string.

Factors

`factor(x)` Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)` Turn a numeric vector into a factor by 'cutting' into sections.

Statistics

`lm(y ~ x, data=df)` Linear model.

`glm(y ~ x, data=df)` Generalised linear model.

`summary` Get more detailed information out a model.

`t.test(x, y)` Perform a t-test for difference between means.

`pairwise.t.test` Perform a t-test for paired data.

`prop.test` Test for a difference between proportions.

`aov` Analysis of variance.

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	`rnorm`	`dnorm`	`pnorm`	`qnorm`
Poisson	`rpois`	`dpois`	`ppois`	`qpois`
Binomial	`rbinom`	`dbinom`	`pbinom`	`qbinom`
Uniform	`runif`	`dunif`	`punif`	`qunif`

Plotting

Also see the **ggplot2** package.

`plot(x)` Values of x in order.

`plot(x, y)` Values of x against y.

`hist(x)` Histogram of x.

Dates

See the **lubridate** package.

Using Git and GitHub with RStudio: : CHEATSHEET

Version control control, also known as **source control**, is the practice of tracking and managing changes to software code.

Version control systems are software tools that help software teams manage changes to source code over time.

Git is an **open-source** software for version control, originally developed in 2005 by Linus Torvalds, the creator of the Linux operating system kernel.

Git it is a version control tool to track the changes in the source code of a project.

GitHub is the most popular hosting service for collaborating on code using Git.

Requirements

1. R and RStudio installed
2. Git installed
3. Register a free GitHub account

Check that Git is installed

In the Terminal of RStudio, enter `which git` to request the path to your Git executable:

```
which git
## /usr/bin/git
```

and `git --version` to see its version:

```
git --version
## git version 2.34.1
```

Introduce yourself to Git

Open a shell from RStudio *Tools > Shell* and type each line separately by substituting your name and the email associated with your GitHub account:

```
git config --global user.name 'Jane Doe'
git config --global user.email
'jane@example.com'
```

Github Glossary

This glossary introduces common Git and GitHub terminology.

Basics

git init <directory>	Create empty Git repository in specified directory.
git clone <repository>	Clone a repository located at <repository> on your local machine.
git config user.name <username>	Define author name to be used for all commits in current repository.
git add <directory>	Stage all changes in <directory> for the next commit.
git commit -m <"message">	Commit the staged snapshot, but instead of launching a text editor, use <"message"> as the commit message.
git status	List which files are staged, unstaged, and untracked.
git log	Display the entire commit history using the default format.
git diff	Show unstaged changes between your index and working directory.

Remote Repositories

git remote add <name> <url>	Create a new connection to a remote repository. After adding a remote, you can use <name> as a shortcut for <url> in other commands.
git fetch <remote> <branch>	Fetches a specific <branch>, from the repository. Leave off <branch> to fetch all remote refs.
git pull <remote>	Fetch the specified remote's copy of current branch and **immediately** merge it into the local copy.
git push <remote> <branch>	Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repository if it doesn't exist.

Undoing Changes

git revert <commit>	Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.
git reset <file>	Remove <file> from the staging area but leave the working directory unchanged. This unstages a file without overwriting any changes.
git clean -n	Shows which files would be removed from working directory. Use the –f flag in place of the –n flag to execute the clean.

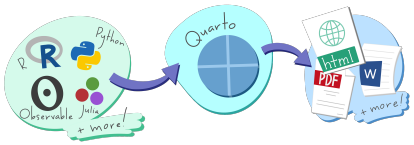
Rewriting Git History

git commit --amend	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
git rebase <base>	Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
git reflog	Show a log of changes to the local repository's HEAD. Add --relative-date flag to show date info or --all to show all refs.

Git Branches

git branch	List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.
git checkout –b <branch>	Create and check out a new named <branch>. Drop the –b flag to checkout an existing branch.
git merge <branch>	Merge <branch> into the current branch.

Publish and Share with Quarto : : CHEATSHEET



Author

WRITE AND CODE IN PLAIN TEXT

Author documents as .qmd files or Jupyter notebooks. Write in a rich Markdown syntax.

Render

GENERATE DOCUMENTS, PRESENTATIONS AND MORE

Produce HTML, PDF, MS Word reveal.js, MS Powerpoint, Beamer Websites, blogs, books...

Publish

SHARE YOUR WORK WITH THE WORLD

Quickly deploy to GitHub Pages, Netlify, Quarto Pub, Posit Cloud, or Posit Connect

GET QUARTO

https://quarto.org/docs/download/

Or use version **bundled with Positron or RStudio**

GET STARTED

https://quarto.org/docs/get-started/

Author

SOURCE FILE: hello.qmd

```
---
title: "Hello, Penguins"
format: html
execute:
  echo: false
---

## Meet the penguins

The `penguins` data contain
from three islands in the P

The three species of penguin
distributions of physical dimensions (@fig-penguins).

```{r}
#| label: fig-penguins
#| fig-cap: "Dimensions of penguins a
#| warning: false
library(tidyverse, quietly = TRUE)
library(palmerpenguins)
penguins |>
  ggplot(aes(x = flipper_length_mm, y = bill_length_mm)) +
  geom_point(aes(color = species)) +
  scale_color_manual(
    values = c("darkorange", "purple", "cyan4")) +
```

Set format(s) and options
Use YAML Syntax

```
## Write with **Markdown**
```
RStudio: Help > Markdown Quick Reference

Use **Visual Editor**

Include code
R, Python, Julia, Observable, or any language with a Jupyter kernel

USE A TOOL WITH A RICH EDITING EXPERIENCE

RStudio Positron VS Code + Quarto extension

Run code cells as you write

Render with a button or keyboard shortcut

Edit Quarto documents with a **Visual Editor**

OR ANY TEXT EDITOR

Quarto documents (.qmd) can be edited in any tool that edits text.

Apply formatting in Visual Editor. Saved as Markdown in source.

Insert elements like code cells, cross references, and more.

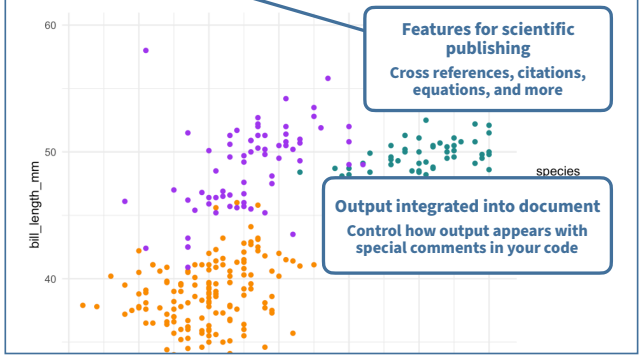
Render

RENDERED OUTPUT: hello.html

Hello, Penguins

Meet the penguins

The three species of penguins have quite distinct distributions of physical dimensions (Figure 1).



Features for scientific publishing
Cross references, citations, equations, and more

Output integrated into document
Control how output appears with special comments in your code

Save, then render to **preview** the document output.

```
Terminal
quarto preview hello.qmd
```

Use **Render** button

Use **Preview** button

The resulting HTML/PDF/MS Word/etc. document will be created and saved in the same directory as the source .qmd file.

BEHIND THE SCENES

When you render a document, Quarto:

1. Runs the code and embeds results and text into an .md file with: **Knitr**, if any {r} cells or, **Jupyter**, if any other cells.
2. Converts the .md file into the output format with Pandoc.

Publish

```
Terminal
quarto publish {venue} hello.qmd
```

{venue}: quarto-pub, connect, gh-pages, netlify, confluence

Quarto Pub — Free publishing service for Quarto content.

posit Connect — Org-hosted, control access, schedule updates.

Use **Publish** button

Use **Posit Publisher** extension

Quarto Projects

CREATE WEBSITES, BOOKS, AND MORE

A directory of Quarto documents + a configuration file (_quarto.yml)

See examples at https://quarto.org/docs/gallery/

Get started from the command line:

```
Terminal
quarto create project {type}
```

{type}: default, website, blog, book, confluence, manuscript

Use **File > New Project**

Use **Quarto: Create Project** command

Artwork from "Hello, Quarto" keynote by Julia Lowndes and Mine Çetinkaya-Rundel, presented at RStudio Conference 2022. Illustrated by Allison Horst.
```

Include Code

CODE CELLS

Code cells start with ```{language} and end with ```.

Use **Insert Code Chunk/Cell**

```r
#| label: chunk-id
library(tidyverse)
```

```python
#| label: chunk-id
import pandas as pd
```

Other languages: `{julia}`, `{ojs}`

Add code cell options with `#|` comments.

Cell options control **execution**, figures, tables, layout and more. See them all at:
https://quarto.org/docs/reference/cells

EXECUTION OPTIONS

OPTION	DEFAULT	EFFECTS
echo	true	false: hide code fenced: include code cell syntax
eval	true	false: don't run code
include	true	false: don't include code or results
output	true	false: don't include results asis: treat results as raw markdown
warning	true	false: don't include warnings in output
error	false	true: include error in output and continue with render

Set execution options at the **cell level**:

```r
#| echo: false
```

```python
#| echo: false
```

Or, **globally** in the YAML header with the **execute** option:

```
---
execute:
  echo: false
---
```

Set options in code cells with `#|` comments and YAML syntax:
`key: value`

INLINE CODE

Use computed values directly in text sections.
Code is evaluated at render and results appear as text.

KNITR	JUPYTER	OUTPUT
Value is `` `r 2+2` ``.	Value is `` `{python} 2+2` ``.	Value is 4.

Set Format and Options

SET FORMAT OPTIONS

```
---
title: "My Document"
format:
  html:
    code-fold: true
    toc: true
---
```

Indent options 4 spaces

Indent format 2 spaces

MULTIPLE FORMATS

```
---
title: "My Document"
toc: true
format:
  html:
    code-fold: true
  pdf: default
---
```

Top-level options apply to all formats

Common formats: **html, pdf, docx, odt, rtf, gfm, pptx, revealjs, beamer**

Render **all** formats:

Terminal
```
quarto render hello.qmd
```

Render a **specific** format:

Terminal
```
quarto render hello.qmd --to pdf
```

	OPTION	html/revealjs	pdf/beamer	docx/pptx	DESCRIPTION	Also use in code cells
Nav	**toc**	X	X	X	Add a table of contents (true or false)	
Nav	**toc-depth**	X	X	X	Lowest level of headings to add to table of contents (e.g. 2, 3)	
Nav	**anchor-sections**	X			Show section anchors on mouse hover (true or false)	
Style	**highlight-style**	X	X	X	Syntax highlighting theme (e.g. arrow, pygments, kate, zenburn)	
Style	**mainfont, monofont**	X	X		Font name. HTML: sets CSS font-family; LaTeX: via fontspec package	
Style	**theme**	X			Bootswatch theme name (e.g. cosmo, darkly, solar etc.)	
Style	**css**	X			CSS or SCSS file to use to style the document (e.g. "style.css")	
Style	**reference-doc**			X	docx/pptx file containing template styles (e.g. file.docx, file.pptx)	
Style	**include-in-header**	X	X		Files of content to include in header of output document, also **include-before-body, include-after-body**	
Style	**keep-md**	X	X	X	Keep intermediate markdown (true or false), also **keep-ipynb**, **keep-tex**	
LaTeX	**documentclass**		X		LaTeX document class, set document class options with **classoption**	
LaTeX	**pdf-engine**		X		LaTeX engine to produce PDF output (xelatex, pdflatex, lualatex)	
LaTeX	**cite-method**		X		Method used to format citations (citeproc, natbib, biblatex)	
Code	**code-fold**	X			Let readers toggle the display of R code (false, true, or show)	✓
Code	**code-tools**	X			Add menu for hiding, showing, and downloading code (true or false)	
Code	**code-overflow**	X			Display of wide code (scroll, or wrap)	✓
Figures	**fig-align**	X	X	/	Alignment of figures (default, left, right, or center)	✓
Figures	**fig-width, fig-height**	X	X	X	Default width and height for figures in inches	Knitr
Figures	**fig-format**	X	X	X	Format for Matplotlib or R figures (retina, png, jpeg, svg, or pdf)	

Visit https://quarto.org/docs/reference/ to see **all options** by format

Add Content

FIGURES

MARKDOWN

```
![CAP](image.png){#fig-LABEL fig-alt="ALT"}
```

COMPUTATION

```
```{python}                    Or {r}
#| label: fig-LABEL
#| fig-cap: CAP
#| fig-alt: ALT
{{ plot code here }}
```
```

CROSS REFERENCES

1. **Add labels**
   Code cell: add option `label: prefix-LABEL`
   Markdown: add attribute `#prefix-LABEL`
2. **Add references** @prefix-LABEL, e.g.

```
You can see in @fig-scatterplot,
that...
```

Prefix	Renders	Prefix	Renders
fig-	Figure 1	eq-	Equation 1
tbl-	Table 1	sec-	Section 1

TABLES

MARKDOWN

```
|object | radius|
|:------|------:|
|Sun    | 696000|
|Earth  |   6371|

: CAPTION {#tbl-LABEL}
```

Insert Table in **Visual Editor**

COMPUTATION Output a Markdown table or an HTML table from your code

KNITR

Use knitr::kable() to produce Markdown:

```
```{r}
#| label: tbl-LABEL
#| tbl-cap: CAPTION
knitr::kable(head(cars))
```
```

Also see the R packages: gt, flextable, kableExtra.

JUPYTER Add `Markdown()` to Markdown output:

```
```{python}
#| label: tbl-LABEL
#| tbl-cap: CAPTION
import pandas as pd, tabulate
from IPython.display import Markdown
df = pd.DataFrame({"A": [1, 2],
                   "B": [1, 2]})
Markdown(df.to_markdown(index=False))
```
```

CITATIONS

1. Add a bibliography **file** to the YAML header:

```
---
bibliography: references.bib
---
```

2. Add citations: `[@citation]`, or `@citation`

Use **Insert Citations** dialog in the **Visual Editor**

Build your bibliography file from your Zotero library, DOI, Crossref, DataCite, or PubMed

CALLOUTS

tip

```
::: {.callout-tip}
## Title

Text
:::
```

Instead of tip use one of: note, caution, warning, or important.

note | warning | caution | important

SHORTCODES

```
{{< include _file.qmd >}}
{{< embed file.ipynb#id >}}
{{< video video.mp4 >}}
```

Data visualization with ggplot2 : : CHEATSHEET

Basics

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

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat =<STAT>, position =<POSITION>) +                    Not required, sensible defaults supplied
<COORDINATE_FUNCTION>+
<FACET_FUNCTION>+
<SCALE_FUNCTION>+
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

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.

Aes

Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

size - integer (in mm for size of points and text)

linewidth - integer (in mm for widths of lines)

shape - integer/shape name or a single character ("a")

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()** Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1**)** x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)**)** - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

b + geom_segment(aes(yend = lat + 1, xend = long + 1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin"**)** x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian"**)** x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot() x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

both continuous

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty)**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point() x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile() x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)** x, y, alpha, color, linetype, size

e + geom_smooth(method = lm**)** x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty)**)** - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

f <- ggplot(mpg, aes(class, hwy))

f + geom_col() x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot() x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"**)** x, y, alpha, color, fill, group

f + geom_violin(scale = "area"**)** x, y, alpha, color, fill, group, linetype, size, weight

both discrete

g <- ggplot(diamonds, aes(cut, color))

g + geom_count() x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)** x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d() x, y, alpha, color, group, linetype, size

h + geom_hex() x, y, alpha, color, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv"**)** x, y, alpha, color, group, linetype, size

visualizing error

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

j + geom_crossbar(fatten = 2**)** - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width Also **geom_errorbarh()**.

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

Draw the appropriate geometric object depending on the simple features present in the data. aes() arguments: map_id, alpha, color, fill, linetype, linewidth.

nc <- **sf::st_read(**system.file("shape/nc.shp", package = "sf"**))**

ggplot(nc) + **geom_sf(**aes(fill = AREA)**)**

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z)**)** x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z)**)** x, y, alpha, color, fill, group, linetype, size, subgroup

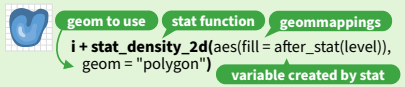
l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE**)** x, y, alpha, fill

l + geom_tile(aes(fill = z)**)** x, y, alpha, color, fill, linetype, size, width

Stats An alternative way to build a layer.

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function).
Use **after_stat(name)** syntax to map the stat variable **name** to an aesthetic.



c + stat_bin(binwidth = 1, boundary = 10**)**
x, y | count, ncount, density, ndensity

c + stat_count(width = 1**) x, y** | count, prop

c + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y | count, density, scaled

e + stat_bin_2d(bins = 30, drop = T**)**
x, y, fill | count, density

e + stat_bin_hex(bins = 30**) x, y, fill** | count, density

e + stat_density_2d(contour = TRUE, n = 100**)**
x, y, color, size | level

e + stat_ellipse(level = 0.95, segments = 51, type = "t"**)**

l + stat_contour(aes(z = z)**) x, y, z, order** | level

l + stat_summary_hex(aes(z = z), bins = 30, fun = max**)**
x, y, z, fill | value

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | value

f + stat_boxplot(coef = 1.5**)**
x, y | lower, middle, upper, width , ymin, ymax

f + stat_ydensity(kernel = "gaussian", scale = "area"**) x, y** | density, scaled, count, n, violinwidth, width

e + stat_ecdf(n = 40**) x, y** | x, y

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq"**) x, y** | quantile

e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95**) x, y** | se, x, y, ymin, ymax

ggplot() + xlim(-5, 5**) + stat_function(**fun = dnorm, n = 20, geom = "point"**) x** | x, y

ggplot() + stat_qq(aes(sample = 1:100)**)**
x, y, sample | sample, theoretical

e + stat_sum() x, y, size | n, prop

e + stat_summary(fun.data = "mean_cl_boot"**)**

h + stat_summary_bin(fun = "mean", geom = "bar"**)**

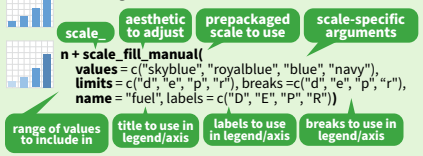
e + stat_identity()

e + stat_unique()

Scales Override defaults with **scales** package.

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

n <- d + geom_bar(aes(fill = fl))



GENERAL PURPOSE SCALES

Use with most aesthetics

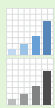
scale_*_continuous() - Map cont' values to visual ones.
scale_*_discrete() - Map discrete values to visual ones.
scale_*_binned() - Map continuous values to discrete bins.
scale_*_identity() - Use data values as visual ones.
scale_*_manual(values = c()**)** - Map discrete values to manually chosen visual ones.
scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks"**)** - Treat data values as dates.
scale_*_datetime() - Treat data values as date times. Same as scale_*_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale.
scale_x_reverse() - Reverse the direction of the x axis.
scale_x_sqrt() - Plot x on square root scale.

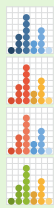
COLOR AND FILL SCALES (DISCRETE)



n + scale_fill_brewer(palette = "Blues"**)**
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red"**)**

COLOR AND FILL SCALES (CONTINUOUS)



o <- c + geom_dotplot(aes(fill = x))

o + scale_fill_distiller(palette = "Blues"**)**

o + scale_fill_gradient(low="red", high="yellow"**)**

o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25**)**

o + scale_fill_gradientn(colors = topo.colors(6)**)**
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES

p <- e + geom_point(aes(shape = fl, size = cyl))

p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7)**)**

p + scale_radius(range = c(1,6)**)**
p + scale_size_area(max_size = 6**)**

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)** - xlim, ylim
The default cartesian coordinate system.

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim - Cartesian coordinates with fixed aspect ratio between x and y units.

r + coord_flip()
Flip cartesian coordinates by switching x and y aesthetic mappings.

r + coord_polar(theta = "x", direction=1**)**
theta, start, direction - Polar coordinates.

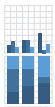
r + coord_trans(y = "sqrt"**)** - x, y, xlim, ylim
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_sf() - xlim, ylim, crs. Ensures all layers use a common Coordinate Reference System.

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))



s + geom_bar(position = "dodge")
Arrange elements side by side.

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height.

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting.

e + geom_label(position = "nudge")
Nudge labels away from points.

s + geom_bar(position = "stack")
Stack elements on top of one another.

Each position adjustment can be recast as a function with manual **width** and **height** arguments:
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines.

r + theme_gray()
Grey background (default theme).

r + theme_dark()
Dark for contrast.

r + theme_classic()

r + theme_light()

r + theme_linedraw()

r + theme_minimal()
Minimal theme.

r + theme_void()
Empty theme.

r + theme() Customize aspects of the theme such as axis, legend, panel, and facet properties.
r + labs(title = "Title") + theme(plot.title.position = "plot")
r + theme(panel.background = element_rect(fill = "blue"))

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
Facet into columns based on fl.

t + facet_grid(year ~ .)
Facet into rows based on year.

t + facet_grid(year ~ fl)
Facet into both rows and columns.

t + facet_wrap(~ fl)
Wrap facets into a rectangular layout.

Set **scales** to let axis limits vary across facets.

t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets:
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet label:

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., labeller = label_bquote(alpha ^ .(fl)**))**

α^c	α^d	α^e	α^p	α^r

Labels and Legends

Use **labs()** to label the elements of your plot.

t + labs(x = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
alt = "Add alt text to the plot",
<AES> = "New **<AES>** legend title"**)**

t + annotate(geom = "text", x = 8, y = 9, label = "A"**)**
Places a geom with manually selected aesthetics.

p + guides(x = guide_axis(n.dodge = 2)**)** Avoid crowded or overlapping labels with guide_axis(n.dodge or angle).

n + guides(fill = "none"**)** Set legend type for each aesthetic: colorbar, legend, or none (no legend).

n + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right".

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred):
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points):
t + xlim(0, 100**) + ylim(**10, 20**)**
t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**

Machine Learning Modelling in R : : CHEAT SHEET

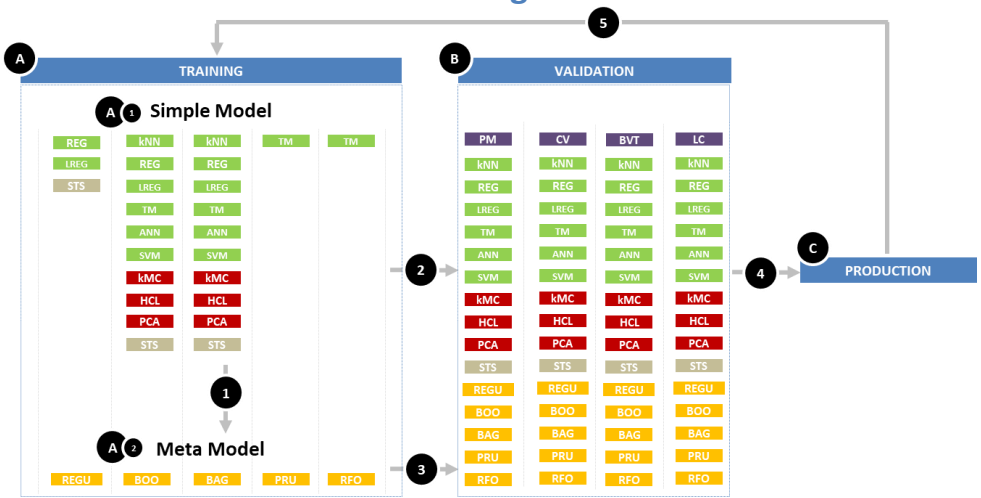
Supervised & Unsupervised Learning

	ALGORITHM	DESCRIPTION	R PACKAGE::FUNCTION	SAMPLE CODE
SUPERVISED LEARNING	NBC Naïve Bayes classifier	A classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature	e1071::naiveBayes	naiveBayes(class ~ ., data = x)
	kNN k–Nearest Neighbours	A non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression	class::knn	knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
	REG Linear Regression	Model the linear relationship between a scalar dependant variable *Y* and one or more explanatory variables (or independent variables) denoted *X*	stats::lm	lm(dist ~ speed, data=cars)
	LREG Logistic Regression	Used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.	stats::glm	glm(Y ~ ., family = binomial (link = 'logit'), data = X)
	TM Tree-Based Models	The idea is to consecutively divide (branch) the training dataset based on the input features until an assignment criterion with respect to the target variable into a "data bucket" (leaf) is reached	rpart::rpart	rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
	ANN Artificial Neural Network	Neural networks are built from units called perceptrons. Perceptrons have one or more inputs, an activation function and an output. An ANN model is built up by combining perceptrons in structured layers.	neuralnet::neuralnet	neuralnet(f,data=train_,hidden=c(5,3),linear.output=T)
	SVM Support Vector Machine	A data classification method that separates data using hyperplanes	e1071::svm	svm(formula, data = NULL, ..., subset, na.action = na.omit, scale = TRUE)
UNSUPERVISED LEARNING	PCA Principal Component Analysis	A procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.	stats::prcomp stats::princomp FactoMineR::PCA ade4::dudi.pca amap::acp	**stats** : prcomp(formula, data = NULL, subset, na.action, ...) **stats** : princomp(formula, data = NULL, subset, na.action, ...) **FactoMineR** : PCA(decathlon, quanti.sup = 11:12, quali.sup=13) **ade4** : dudi.pca(deug$tab, center = deug$cent, scale = FALSE, scan = FALSE) **amap** : acp(lubisch)
	kMC k-Mean Clustering	Aims at partitioning *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean	stats::kmeans	kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)
	HCL Hierarchical Clustering	An approach which builds a hierarchy from the bottom-up, and doesn't require the number of clusters to be specified beforehand.	stats::hclust	hclust(d, method = "complete", members = NULL)

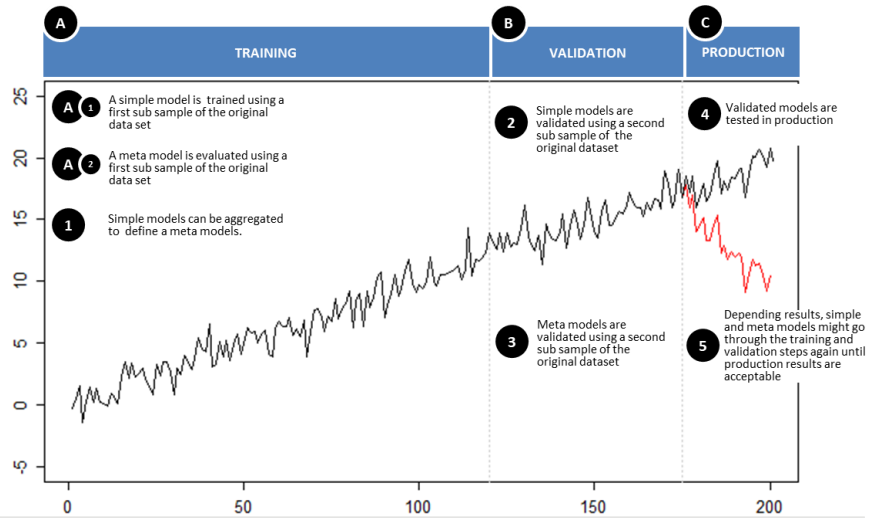
Meta-Algorithm, Time Series & Model Validation

	ALGORITHM	DESCRIPTION	R PACKAGE::FUNCTION	SAMPLE CODE
META ALGORITHM	REGU Regularisation L1 (Lasso) L2 (Ridge)	Regularization adds a penalty on the different parameters of a model to reduce the freedom of the model. Hence, the model will be less likely to fit the noise of the training data and will improve the generalization abilities of the model	glmnet::glmnet	**L1 :** glmnet(myMatrixA, myMatrixB , family = "gaussian", alpha = 1) **L2 :** glmnet(myMatrixA, myMatrixB , family = "gaussian", alpha = 0)
	BOO Boosting	A process of iteratively refining, e.g. by reweighting, of estimated regression and classification functions (though it has primarily been applied to the latter), in order to improve predictive ability.	**Paramatric model** - mboost::glmboost	glmboost(Yen ~ ., data = curr1[trnidxs,])
	BAG Bagging	Bagging is a way to increase the power of a predictive statistical model by taking multiple random samples (with replacement) of the training data set, and using each of them to construct a separate model and separate predictions for the original test set	**All models**: foreach **Tree models**: ipred::bagging	**foreach :** d <- data.frame(x=1:10, y=rnorm(10)) s <- foreach(d=iter(d, by='row'), .combine=rbind) %dopar%d identical(s, d) **ipred :** bagging(formula, data, subset, na.action=na.rpart, \dots)
	PRU Pruning	Pruning is a technique that reduces the size of decision tree by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier and hence improves predictive accuracy by reducing overfitting	rpart::prune	prune(x, cp = 0.1)
	RFO Random Forrest	An ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression)	randomForest::randomForest	randomForest(X ~ ., data = Y , subset = mySub)
TIME SERIES	STS Lead-lag analysis, Auto-correlation, Spectral analysis, Time series clustering, Seasonality, Trend....	Random sampling of observations for training and testing a model can be an issue when faced with a times dimension. Random sampling may either destroy serial correlation properties in the data which we would like to exploit	stats xts forecast spectral TTR	**Auto-correlation**: acf(x, lag.max = NULL, type = c("correlation", "covariance", "partial")) **Spectral Analysis**: spec.pgram(myTs, spans = NULL) **Seasonal Decomposition of Time Series** - stl(x, s.window = 7, t.window = 50, t.jump = 1) .....
MODEL VALIDATION	PM Performance metrics	Depends on the problem: • **Regression**: squared errors, outliers, error rate... • **Classification**: Accuracy, precision, recall, F-score...	**Regression**-stats::outlierTest, stats:: qqPlot .... **Classification**-ROCR:: **Tree**: caret:: confusionMatrix	**Regresion**: fit <- lm(Y~X,data=myData) outlierTest(fit) qqPlot(fit, main="QQ Plot")
	BVT Biais-Variance Tradeoff	• Simple models with few parameters are easier to compute but may lead to poorer fits (**high bias**). • Complex models may provide more accurate fits but may over-fit the data (**high variance**)	Tailored to the analysis	Tailored to the analysis
	CV Cross validation	Cross validation compares the test performances of different model realisations with different sets or values of parameters	caret::createDataPartition caret::createFolds	createDataPartition(classes, p = 0.8, list = FALSE)
	LC Learning Curves	Learning curves plot a model's training and test errors, or the chosen performance metric, depending on the training set size	caret:: learing_curve_dat	learing_curve_dat(dat, outcome = NULL, proportion = (1:10)/10, test_prop = 0, verbose = TRUE, ...)

Standard Modelling Workflow



Time Series View



Distribuciones

Si $(\Omega, \mathcal{A}, \mathbb{P})$ es un espacio de probabilidad, X una variable aleatoria entonces se puede calcular y simular numeros aleatorios que tienen la misma distribución que X.

- $P(X \in A) = \begin{cases} \sum_{x \in R_X} \mathbb{P}(X = x) & \text{si } X \text{ es discreta} \\ \int_A f_X(x)\, dx & \text{si } X \text{ es continua} \end{cases}$
- la función de distribución de X es $F_X : \mathbb{R} \to \mathbb{R}$ definida por

$$F_X(x) = \mathbb{P}(X \leq x) \ \forall x \in \mathbb{R}$$

- la densidad $\begin{cases} \mathbb{P}(X = x) & \text{si } X \text{ es discreta} \\ f_X(x) & \text{si } X \text{ es continua} \end{cases}$
- los cuantiles $F^-(p) = inf\{x : F(x) \geq p\}$[1]

[1] Si F es biyectiva, $F^- = F^{-1}$

Distribuciones clásicas

1. Distribuciones discretas
   - binomial (n, p) — binom
   - hypergeométrica (N, n, k) — hyper
   - poisson λ — pois
   - geométrica p — geom
   - ley con soporte finito $\{(a_i, p_i), i = 1...m\}$ — sample
2. Distribuciones continuas
   - normal (μ, σ^2) — norm
   - uniforme en $[a, b]$ — unif
   - t-student con p grados de libertad — t
   - chi cuadrado con p grados de libertad — chisq

Si zzz representa el nombre de alguna distribución entonces:

- dzzz$(x, ...)$ calcula la densidad de la distribución zzz en x
- pzzz$(x, ...)$ calcula la función de distribución en x
- qzzz$(p, ...)$ calcula el cuántil de orden p
- rzzz$(x, ...)$ da una muestra de tamaño n

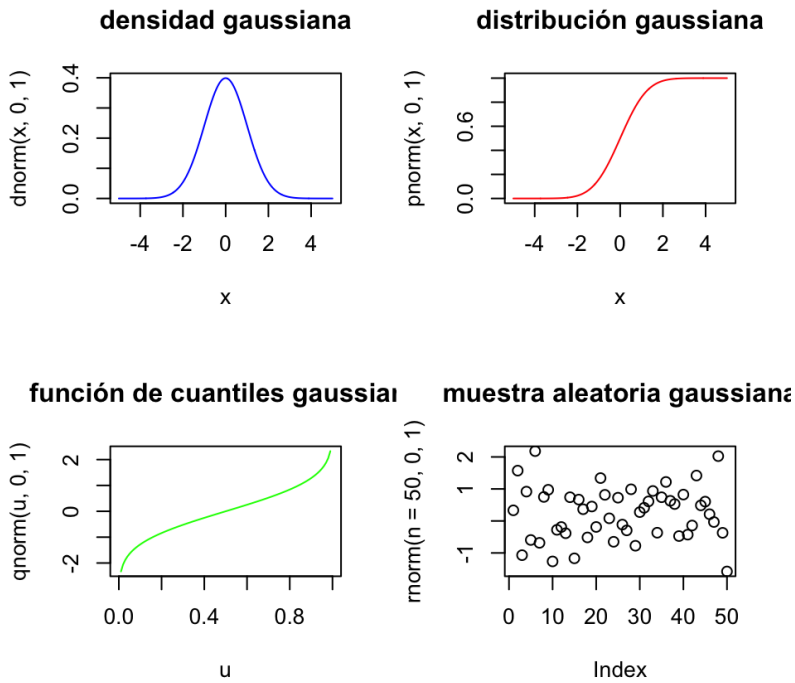
donde .... representan los paramétros específicos de cada distribución

con la gaussiana $\mathcal{N}(0,1)$

- $dnorm(x,0,1)$ densidad en el punto x
- $pnorm(x,0,1)$ función de distribución en el punto x
- $qnorm(p,0,1)$ cuantil $p+$
- $rnorm)(n,0,1)$ muestra de tamaño n

```
x=seq(-5,5,.01)
u=seq(0,1,.01)

par(mfrow=c(2,2))
plot(x,dnorm(x,0,1),type='l', col='blue',main='densidad gaussiana')
plot(x,pnorm(x,0,1),type='l', col='red',main='distribución gaussiana')
plot(u,qnorm(u,0,1),type='l', col='green',main='función de cuantiles gaussiana')
plot(rnorm(n=50,0,1), main='muestra aleatoria gaussiana')
```

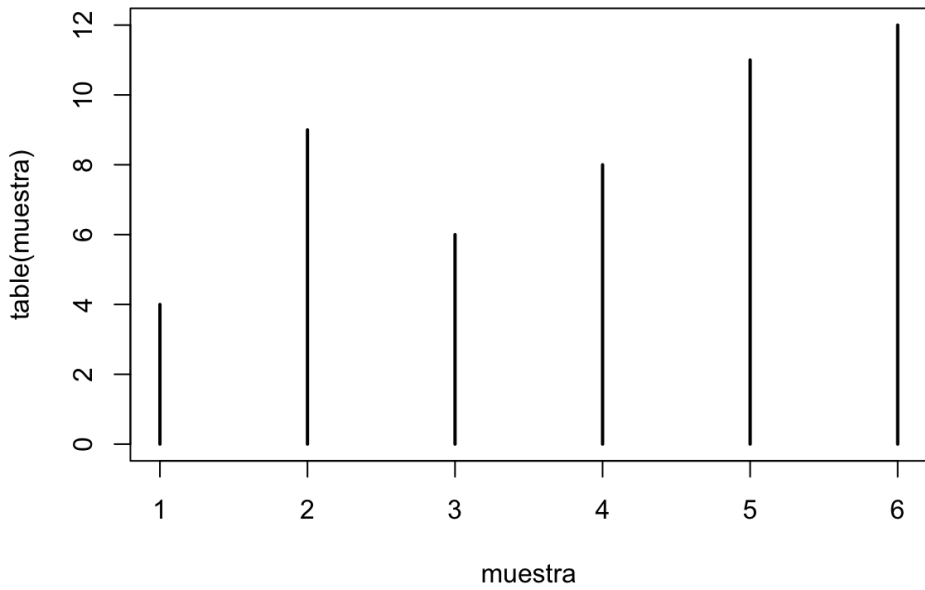


Simulación de una distribución discreta con soporte finito

Sea X una variable aleatoria con soporte en $\{a_1, \ldots, a_L\}$ tal que $\mathbb{P}(X = a_\ell) = p_k$ para todo $\ell = 1, \ldots, L$

- La función sample es un generador de numeros aleatorios
- sample(x, n, replace = T, prob) es el comando

```
muestra=sample(x=c(1:6),size=50,
replace=T)
muestra
 [1] 5 1 5 6 5 2 3 5 6 4 3 5 6
[14] 4 2 3 6 5 2 4 3 2 5 3 6 5
[27] 6 1 2 6 2 6 6 2 6 4 4 5 4
[40] 4 5 5 1 4 3 2 1 6 6 2
table(muestra)
muestra
 1  2  3  4  5  6
 4  9  6  8 11 12
plot(table(muestra))
```



Escribir una función

R tiene muchas funciones, pero podemos y deberemos implementar otras.

```
cuentas = function(x, y){
a = x+2*y
b=x-y
c=x/y
final=c(mean(a),sd(a),mean(b),sd(b),
mean(c),sd(c))
salida=list(x=x,y=y,resultados=result)
return(salida)}
x <- runif(50,0,1)
y <- rnorm(50,0,3)
cuentas(x,y)
```

```
> cuentas(x,y)
$x
 [1] 0.02489827 0.00484174 0.13137825 0.02481710 0.09212945 0.47555822 0.19665676
 [8] 0.81467395 0.24405923 0.48880764 0.64888248 0.15871647 0.83697791 0.43659653
[15] 0.23349553 0.29885599 0.15942134 0.58556710 0.14877798 0.17905956 0.33922150
[22] 0.18390194 0.40554152 0.61355962 0.68567133 0.82352560 0.37398211 0.95284100
[29] 0.01883216 0.61237497 0.46357376 0.35472962 0.88242303 0.07580286 0.88329119
[36] 0.18822184 0.21506675 0.86953229 0.06083806 0.23489960 0.99301551 0.67819773
[43] 0.43548680 0.37730994 0.46118402 0.80185187 0.66296698 0.46600366 0.05325944
[50] 0.23719792

$y
 [1] -2.184657334 -4.621327216 -2.079283842  0.356548299 -4.094128374  1.769948038
 [7]  0.868032086 -2.712645077  0.678974827  2.244243487  3.183285760 -0.638544836
[13] -0.280910382 -0.260142405  4.324385268  3.375215677  2.503204703 -0.862022400
[19]  1.119724302  1.209870994 -3.125019883 -5.184913546  1.925490083 -4.587931594
[25]  0.005051065  0.750743464  1.691602171  0.568278713 -2.198561419  2.959097580
[31]  5.215901301  2.643536427 -5.830952704  4.198728556 -0.168167839  1.574742838
[37]  1.866099708 -0.290058220 -0.225789593  3.057471208  2.134805767  2.970786738
[43]  7.148780086  1.993247591  0.622143472 -6.631899332  8.075142010 -1.448030465
[49]  7.124204145  1.123930704

$resultados
[1]  0.52406972  5.94214664  0.45731884  2.94956123 -0.08869252  1.75717340
```

Variable cuantitativas

Serie estadística continua

Si X es el puntaje de 120 estudiantes en una prueba de probabilidad:

97	63	81	9	27	65	2	59	25	94	21	85	38	32	36
90	79	57	15	76	44	53	34	51	74	31	88	29	28	27
97	81	3	58	66	4	84	59	50	77	50	14	83	72	11
65	39	84	41	1	40	81	70	40	25	41	26	29	23	55
73	36	87	94	73	66	91	20	21	86	59	95	91	88	64
64	85	80	82	7	25	71	58	32	90	92	47	1	64	5
85	26	92	60	5	74	61	52	64	33	77	67	69	89	26
24	52	51	92	87	70	6	61	43	40	36	59	62	92	61

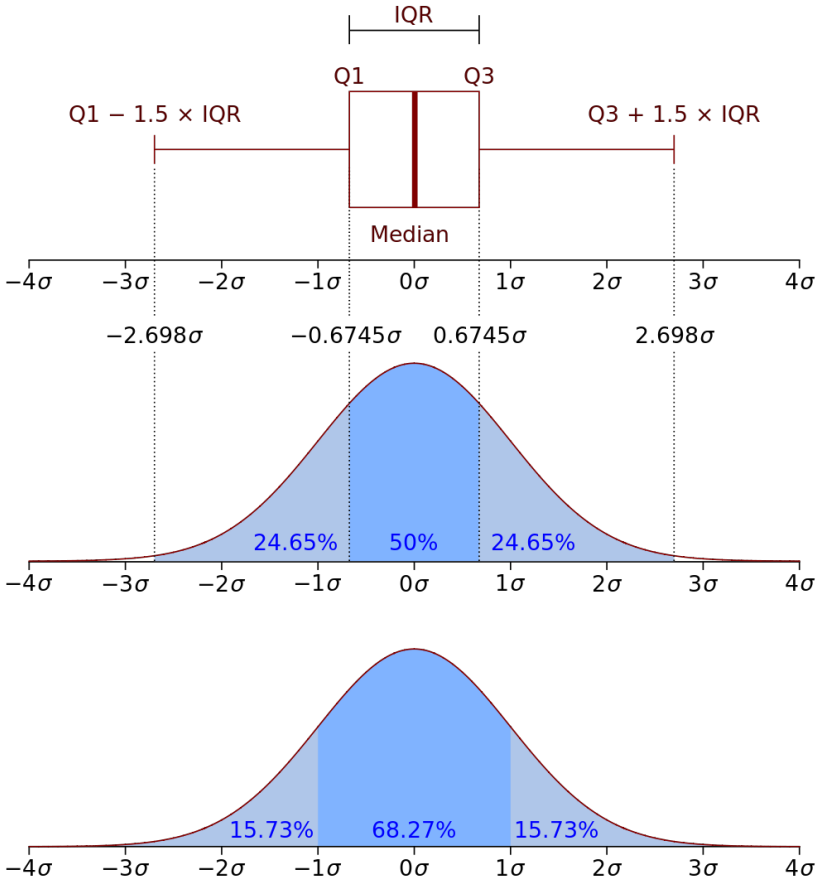
Se pueden identificar algunas cantidades, en particular los 5 números importantes que nos servirán para hacer el diagrama de caja: el mínimo m, el primer cuartil Q_1, la mediana m, el tercer cuartil Q_3, el máximo M (se incluye además el promedio pr):

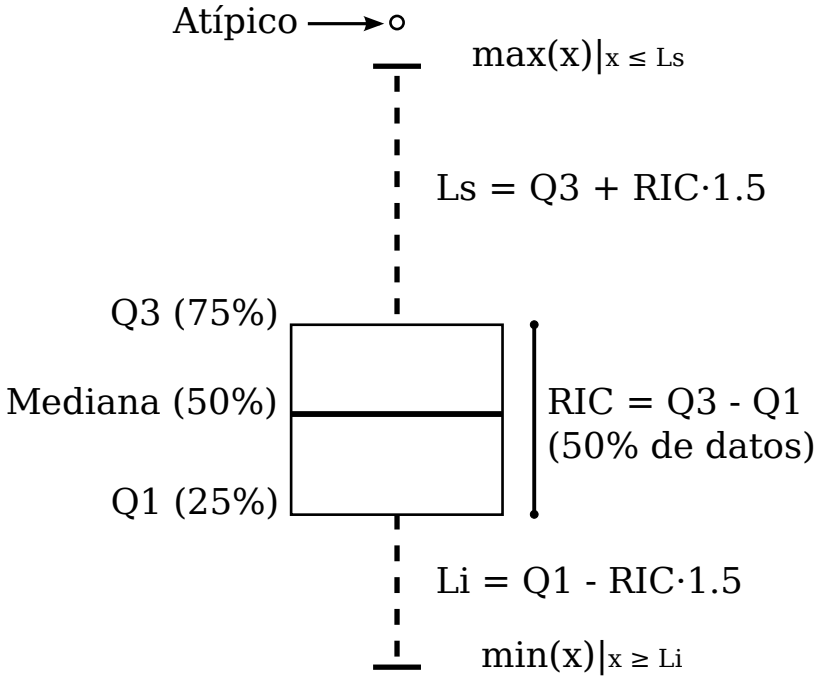
x_m	Q_1	m	$\overline{x}$	Q_3	x_M
1.00	32	59.00	54.52	79	97.00

Si $\hat{F}(x)$ es la proporción de individuos en la muestra con un valor menor o igual a x entonces:

- La mediana es el valor m de la muestra tal que $\hat{F}(m) = 0.5$
- Cuartiles: $\hat{F}(Q_1) = 0.25$, $\hat{F}(Q_2) = 0.5$, $\hat{F}(Q_3) = 0.75$
- Cuantil p: $\hat{F}(x_p) = p$

Boxplot

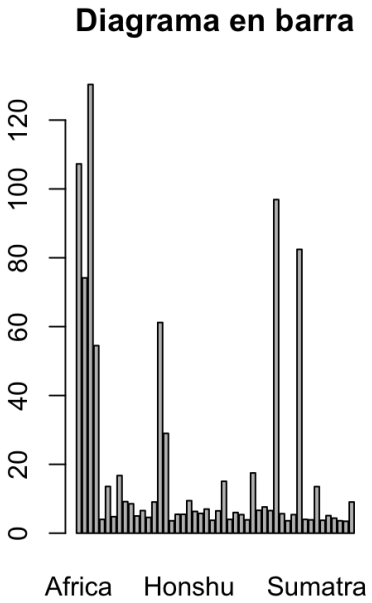


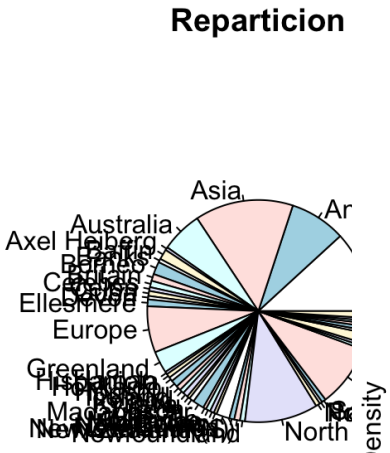


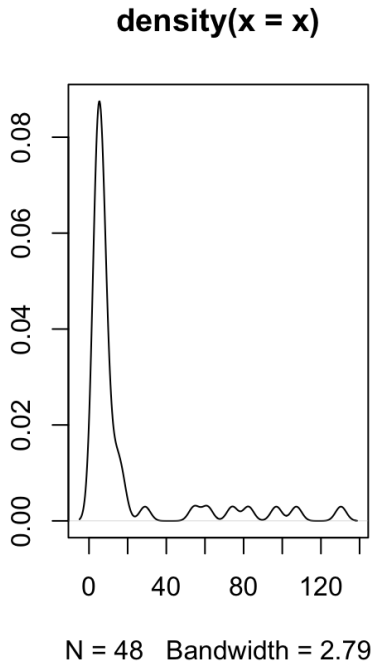
- El valor indicado por la raya en la caja es la mediana
- Las extremidades de la caja corresponden a los cuartiles Q_1 y Q_3
- El rango intercuartílico es $R = Q_3 - Q_1$
- Las extremidades de los bigotes corresponden al valor más grande menor a $Q_3 + 1.5R$ y al valor más chico mayor a $Q_1 - 1.5R$
- Los puntos alineados con los bigotes pero fuera de ellos corresponden a los puntos atípicos o outliers

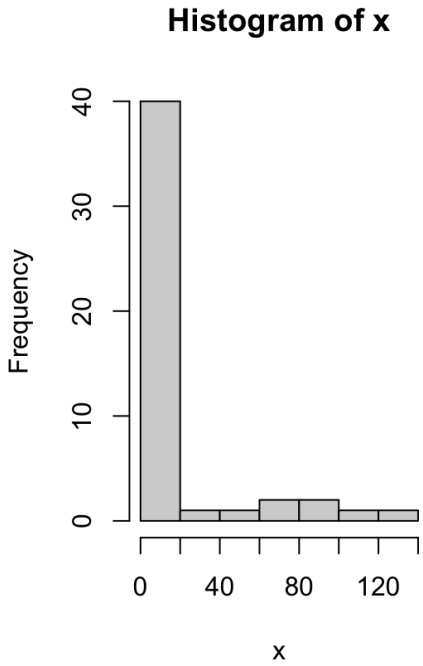
Estimación densidad

El set de datos Island contiene las áreas, en miles de millas cuadradas, de las masas terrestres que superan las 10 000 millas cuadradas. Como las unidades son grandes tomamos la raíz cuadrada.







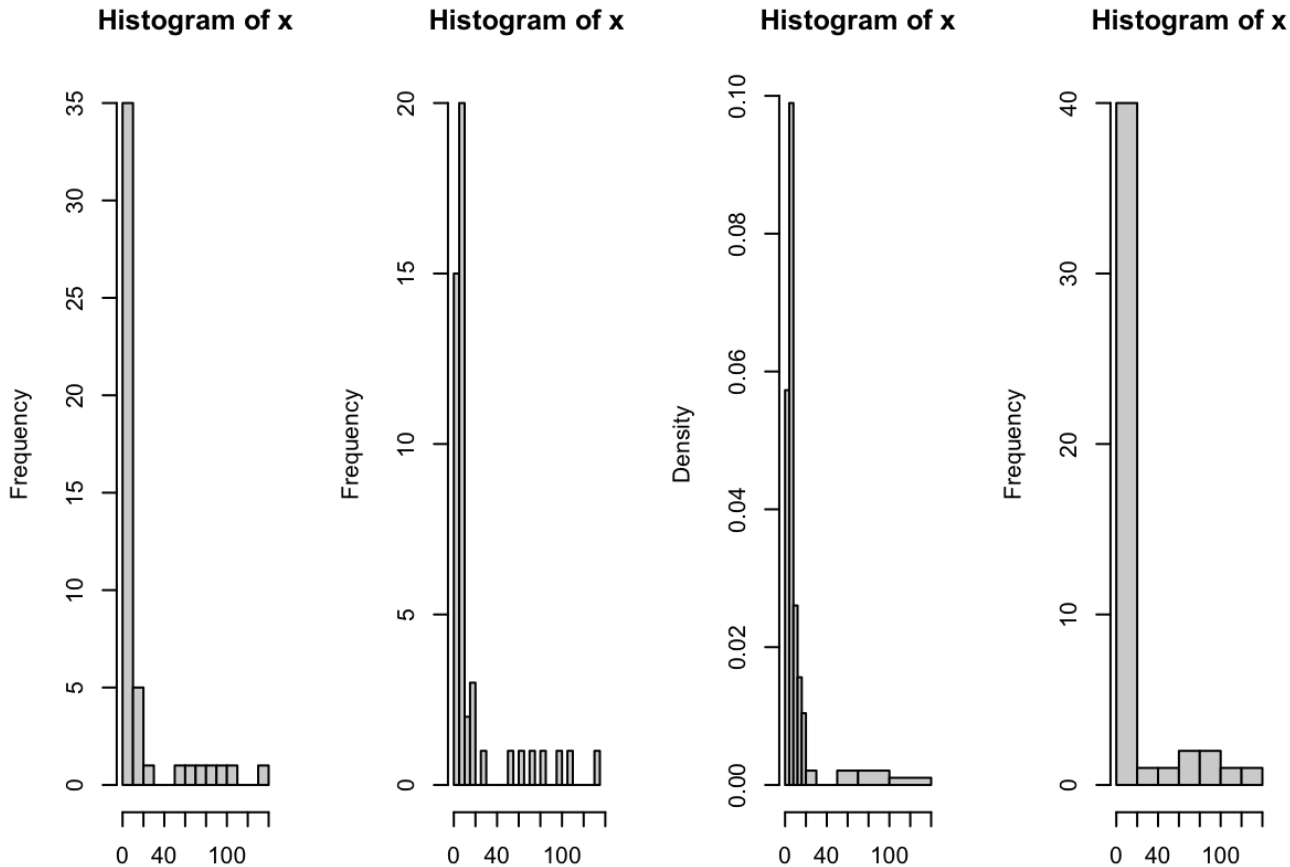



```
x=sqrt(islands)
par(mfrow=c(1,2))
barplot(x,main='Diagrama en barra')
pie(x,main='Reparticion')
plot(density(x))
hist(x)
```

Histogramas

Se puede elegir la cantidad de celdas, o dar un vector, equiespaciado o no, u optimizar según una regla.

```
hist(x,breaks=10)
hist(x,breaks=20)
hist(x,breaks=c(4 *0:5,30, 45, 50 ,70 , 100,140))
hist(x,breaks='Scott')
```

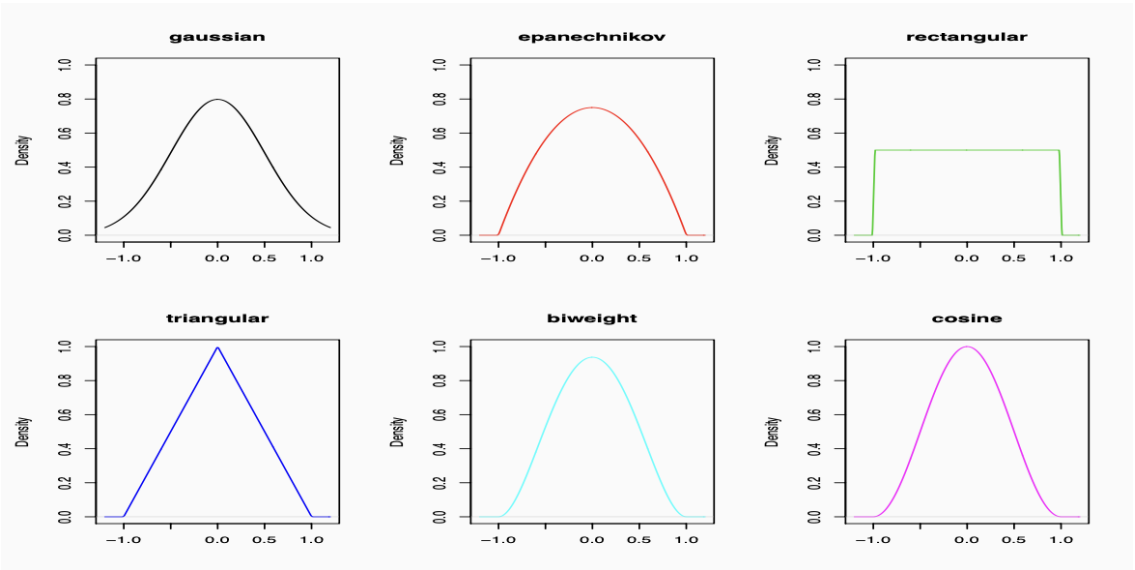


El estimador por núcleo

$X_1, \ldots, X_n$ es una muestra aleatoria con densidad f. La función density calcula un estimador de la densidad f de la forma (Kernel Density Estimator):

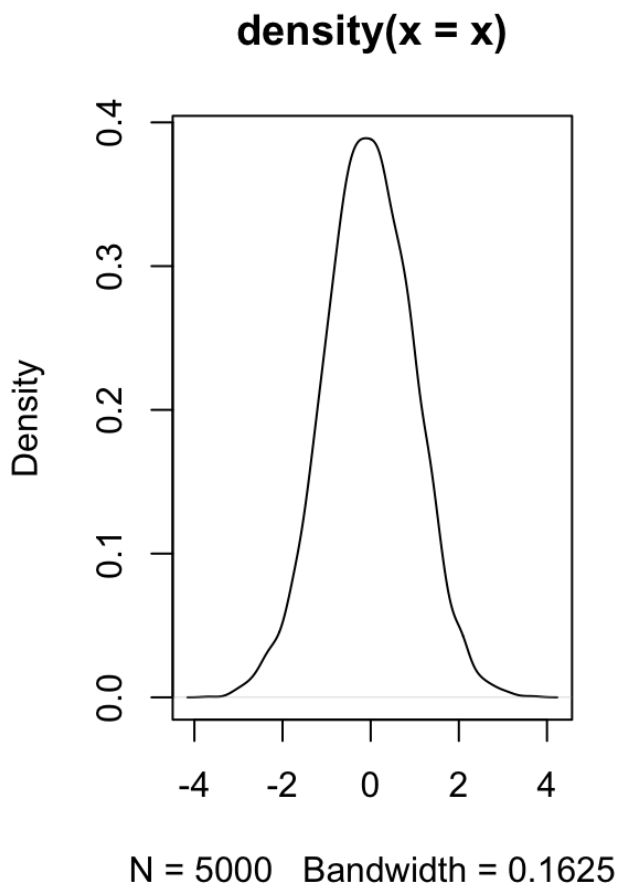
$$f_{KDE}(x) = \frac{1}{n\mathrm{b_w}} \sum_{i=1}^{n} K\left(\frac{x - X_i}{\mathrm{b_w}}\right)$$

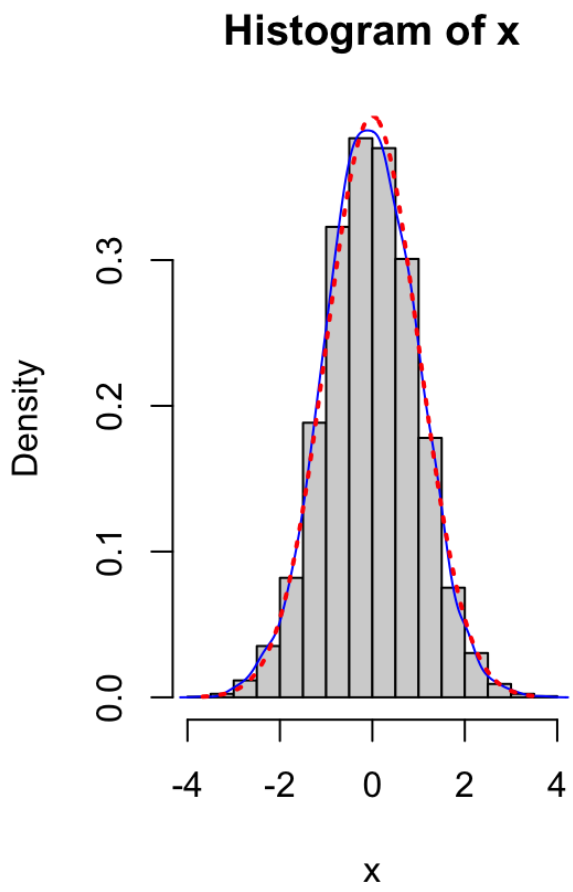
- La elección del núcleo K ($\int K(u)\,du = 1,\ K(u) = K(-u)\,\forall\, u$)



- La elección de la ventana (bandwith $\mathrm{b_w}$). Este parámetro se puede optimizar.

Estimador KDE






```
par(mfrow=c(1,2))
x=rnorm (5000)
y=density(x)
plot(y)
hist(x,proba=T)
lines(y,col='blue')
z=seq(min(x),max(x),0.01)
lines(z, dnorm(z,0,1), lwd=2,lty =3,col='red')
```

Estimador KDE

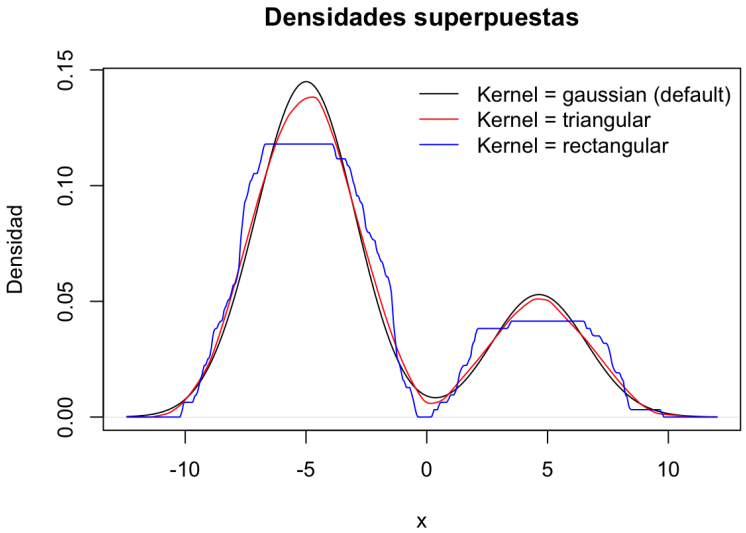
Si consideramos la densidad

$$f(x) = \frac{1}{4}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x+5)^2} + \frac{3}{4}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x-5)^2}$$

Estimador KDE

Si consideramos la densidad

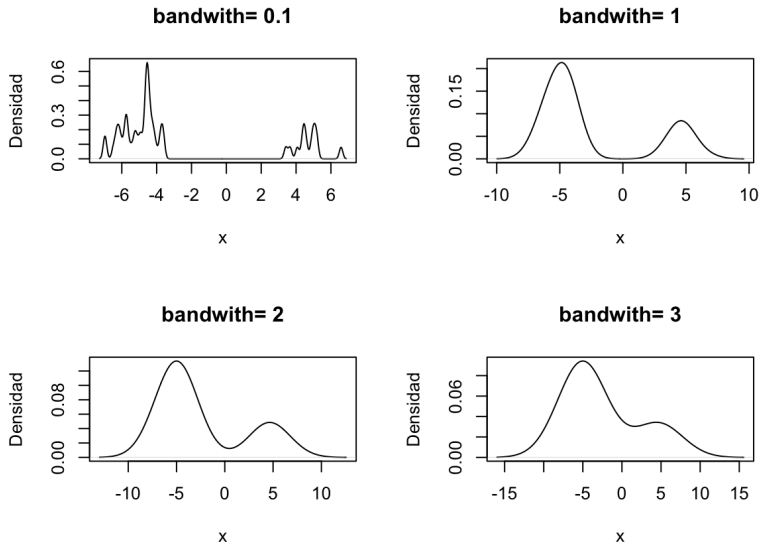
$$f(x) = \frac{1}{4}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x+5)^2} + \frac{3}{4}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(x-5)^2} \quad \text{(¡sí, es una densidad!)}$$




```
w=rbinom(50,1,1/4)
datos = w*rnorm(50,5) +(1-w) * rnorm(50,-5)

plot(density(datos),
     main = "Densidades superpuestas",
     xlab = "x",
     ylab = "Densidad")

lines(density(datos, kernel="triangular"),
      col="red")
lines(density(datos, kernel="rectangular"),
      col="blue")
```

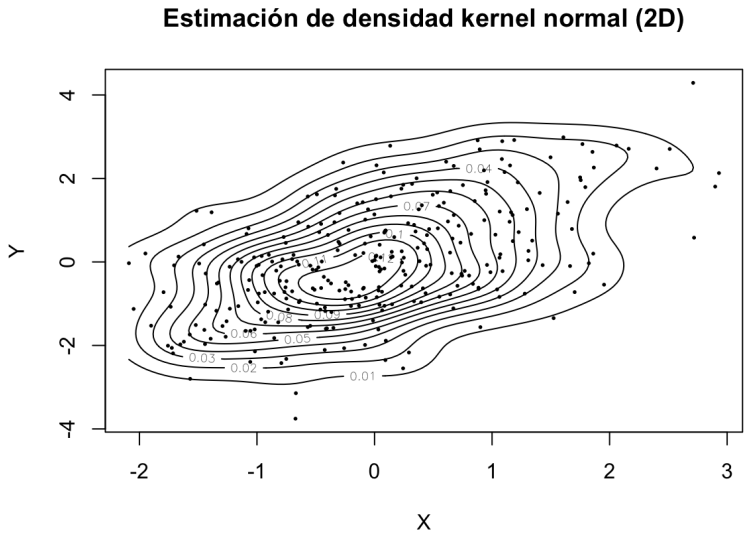


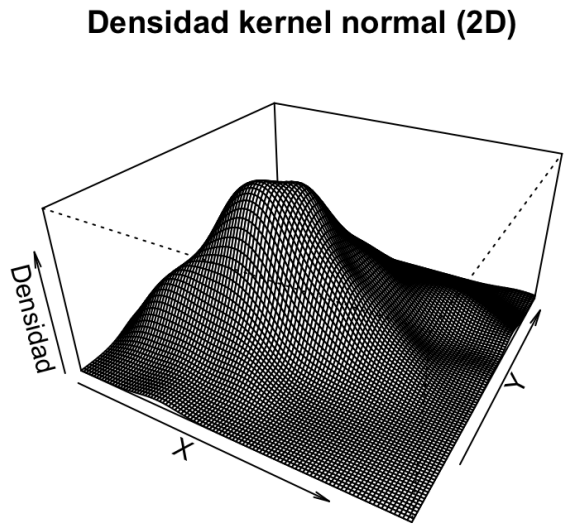

```
par(mfrow=c(2,2))
for(i in c(0.1,1,2,3)){plot(density(datos,bw=i),
                            main = paste("bandwith=",i),
                            xlab = "x",
                            ylab = "Densidad")  }
```

Estimador KDE 2D

$$f_{KDE}(x, y) = \frac{1}{n b_{w_1} b_{w_2}} \sum_{i=1}^{n} K\left(\frac{x - X_i}{b_{w_1}}\right) K\left(\frac{y - Y_i}{b_{w_2}}\right)$$

```
library(MASS)
set.seed(123)
# 1. Parámetros
n <- 300
mu <- c(0, 0)
Sigma <- matrix(c(1, 0.8,
                  0.8, 2),
                nrow = 2, byrow = TRUE)
# 2. Generar datos
datos <- mvrnorm(n = n, mu = mu, Sigma = Sigma)
x <- datos[,1]
y <- datos[,2]
# 3. Gráfico de los datos
plot(x, y,
     pch = 16, cex = 0.5,
     xlab = "X", ylab = "Y",
     main = "Datos normales bivariados")
# 4. Estimación de densidad kernel normal (2D)
dens <- kde2d(x, y, n = 100)
# 5. Contornos de la densidad + puntos
contour(dens,
        xlab = "X",
        ylab = "Y",
        main = "Estimación de densidad kernel normal (2D)
points(x, y, pch = 16, cex = 0.4)
# 6. Superficie 3D de la densidad
persp(dens,
      theta = 30, phi = 30,
      expand = 0.5,
      xlab = "X",
      ylab = "Y",
      zlab = "Densidad",
      main = "Densidad kernel normal (2D)")
```





Variables cualitativas

Serie estadística cualitativa

Disponemos de $n = 96$ observaciones de una variable cualitativa X que representa el nivel de satisfacción de clientes en una escala del 1 al 5.

3	1	5	1	1	2	4	2	5	1	5	5	2	1	2	2
2	2	2	3	2	2	2	4	3	1	4	3	2	1	1	3
4	1	4	4	2	4	1	2	3	4	3	4	2	1	4	5
2	1	3	1	2	1	3	1	2	2	3	2	5	1	2	4
1	4	2	1	3	1	1	2	4	1	5	1	3	2	2	1
2	3	2	2	2	5	3	2	2	2	5	1	5	3	4	1

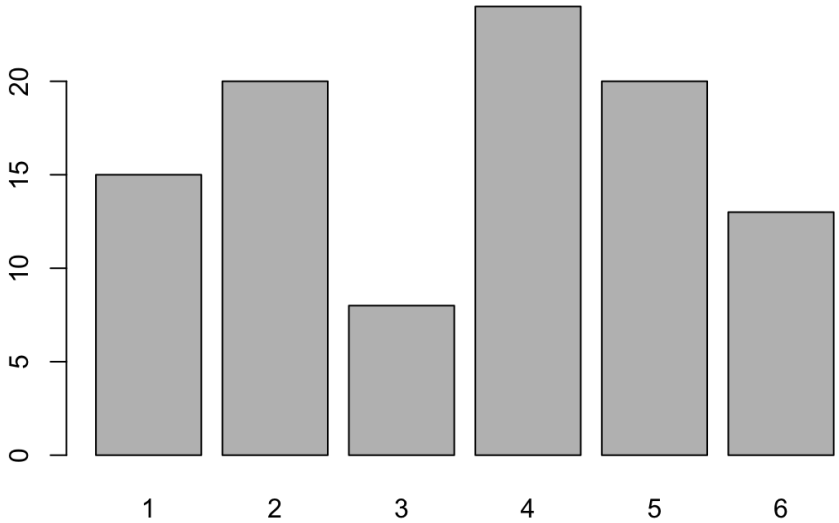
un resumen estadístico de esta serie:

	1	2	3	4	5
Cantidades	25	32	15	14	10
Proporciones	0.26	0.33	0.16	0,15	0,1
Cantidades cumuladas	25	57	72	86	96
Proporciones cumuladas	0.26	0.59	0.75	0.9	1

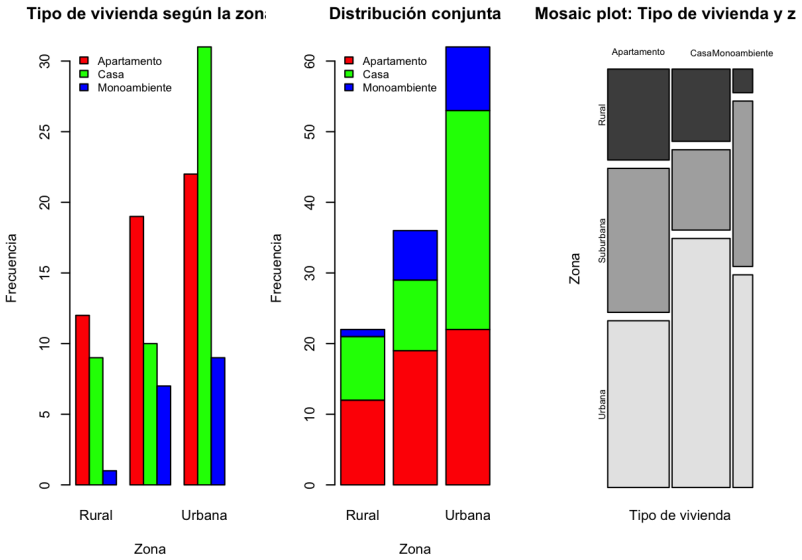
Las proporciones de cada modalidad constituye la distribución empírica de la variable observada.

Otro ejemplo

```
muestra2=sample(x=c(1:6),size=100,
replace=T)
plot(table(muestra2))
barplot(table(muestra2))
summary(muestra2)
   Min. 1st Qu.  Median    Mean 3rd Qu.
   1.00    2.00    4.00    3.53    5.00
cumsum(table(muestra2))
  1   2   3   4   5   6
 15  35  43  67  87 100
cumsum(proportions(table(muestra2)))
   1    2    3    4    5    6
0.15 0.35 0.43 0.67 0.87 1.00
```

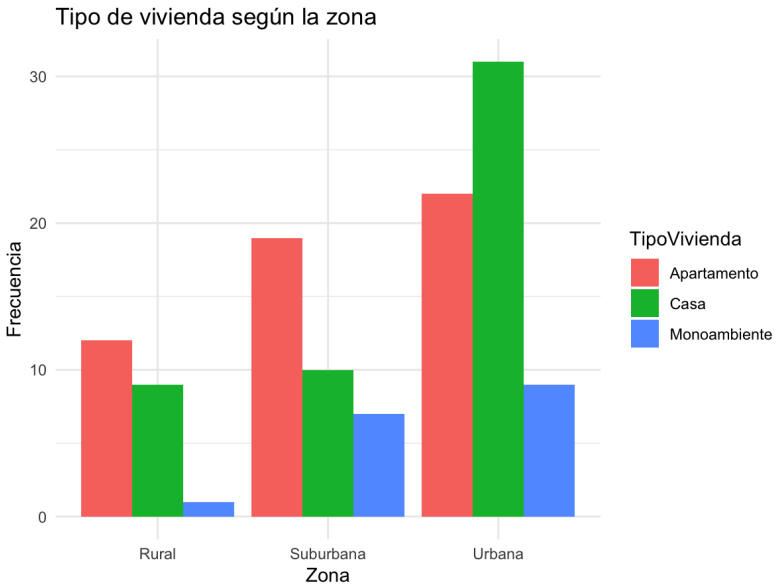


Comparación de dos variables cualitativas




```
par(mfrow=c(1,3))
colores <- rainbow(nrow(tabla))

barplot(tabla,
        beside = TRUE,
        legend = FALSE,
        col=colores,
        xlab = "Zona",
        ylab = "Frecuencia",
        main = "Tipo de vivienda según la zona")
legend("topleft",
       legend = rownames(tabla),
       fill = colores,   # clave para que aparezcan los colores
       cex = 0.8,
       bty = "n")
mosaicplot(tabla,
           main = "Mosaic plot: Tipo de vivienda y zona",
           xlab = "Tipo de vivienda",
           ylab = "Zona",
           color = TRUE)
```




```
library(ggplot2)
ggplot(datos, aes(x = Zona, fill = TipoVivienda)) +
  geom_bar(position = "dodge") +
  labs(title = "Tipo de vivienda según la zona",
       y = "Frecuencia") +
  theme_minimal()
```

Independencia de dos variables cualitativas

Tabla de contingencia de plebiscito de una ley según preferencia política

	a favor	indiferente	en contra	total
democrato	138	83	64	285
republicano	64	67	84	215
total	202	150	148	500

Independencia de dos variables cualitativas

Tabla de contingencia de plebiscito de una ley según preferencia política

	a favor	indiferente	en contra	total
democrato	138	83	64	285
republicano	64	67	84	215
total	202	150	148	500

EL test de independencia χ^2 es

$$\begin{cases} (H_0): & X \text{ e } Y, \text{son independientes} \\ (H_1): & X \text{ e } Y \text{ no son independientes} \end{cases}$$

y tiene como estadística

$$\chi^2 = \sum_{i=1}^{K} \sum_{j=1}^{L} \frac{(n_{ij} - E_{ij})^2}{E_{ij}}$$

donde $E_{ij} = \frac{n_{i\cdot} n_{\cdot j}}{n}$ es la cantidad teórica esperada en el casillero (i, j) si las variables fueran independientes.

Es positivo y tiene distribución χ^2 con $(L-1) \times (K-1)$ grados de libertad. Más cerca está de 0, más las variables tienden a ser independientes.

Comparación de dos variables cualitativas

	a favor	indiferente	en contra	total
democrato	$\frac{285(202)}{500} = 115,14$	$\frac{285(150)}{500} = 85,5$	$\frac{285(148)}{500} = 84,36$	285
republicano	$\frac{215(202)}{500} = 86,86$	$\frac{215(150)}{500} = 64,5$	$\frac{215(148)}{500} = 63,64$	215
total	202	150	148	500

	a favor	indiferente	en contra	total
democrato	138 (115.4)	83 (85.5)	64 (84,36)	285
republicano	64 (86.86)	67 (64.50)	84 (63,64)	215
total	202	150	148	500

El cálculo del estadístico del test es:

$$\chi^2 = \frac{(138-115.4)^2}{115,14} + \frac{(83-85,5)^2}{85,5} + \frac{(64-86,86)^2}{86,86} + \frac{(64-86,86)^2}{86,86}$$

$$+\frac{(67-64,5)^2}{64,5} + \frac{(84-63,64)^2}{63,64} = 22,152$$

con grados de libertad $(2-1)(3-1) = 2$
Como $\mathbb{P}(\chi^2 > 22.152) = 0.001 < 0.05$ rechazamos la hipótesis nula de independencia.

¿Como hacemos esto en R?

Veamos como realizar un test de independencia entre preferencia política y posición frente al plebiscito, usando la tabla de contingencia:

```
# Crear la tabla de contingencia
tabla <- matrix( c(138, 83, 64,64, 67, 84), nrow = 2,byrow = TRUE)

# Nombrar filas y columnas
rownames(tabla) <- c("Democrata", "Republicano")
colnames(tabla) <- c("A favor", "Indiferente", "En contra")

# Ver la tabla
tabla

# Test chi-cuadrado de independencia
test_chi <- chisq.test(tabla)
test_chi
test_chi$expected
if(test_chi$p.value < 0.05){
  "Se rechaza H0: hay asociación entre preferencia política y postura frente al plebiscito."
} else {
  "No se rechaza H0: no se detecta asociación."
}
```

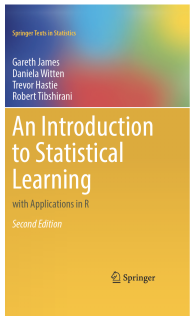
```
> tabla
           A_favor Indiferente En_contra
Democrata      138          83        64
Republicano     64          67        84
> # Test chi-cuadrado de independencia
> test_chi <- chisq.test(tabla)
> test_chi

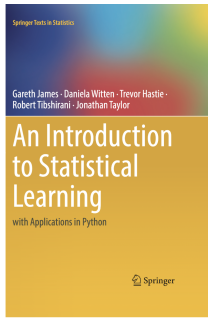
        Pearson's Chi-squared test

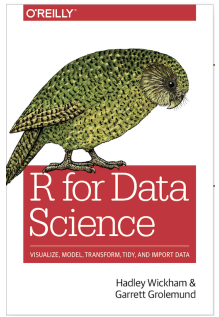
data:  tabla
X-squared = 22.152, df = 2, p-value = 1.548e-05

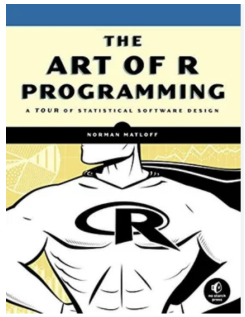
> test_chi$expected
           A_favor Indiferente En_contra
Democrata   115.14        85.5     84.36
Republicano  86.86        64.5     63.64
```

Referencias









- Venables, William N., et al. An introduction to R (2009) `https://cran.r-project.org/doc/manuals/R-intro.pdf`
- Wickham, Hadley; Cetinkaya-rundel, Mine; Grolemund, Garrett. R for data science: import, tidy, transform, visualize, and model data. O'Reilly Media, Inc., 2023. `https://r4ds.had.co.nz`
- Matloff, Norman. The art of R programming: A tour of statistical software design. No Starch Press, 2011.
- James, Gareth, Witten, Daniela, Hastie, Trevor, Tibshirani, Robert. An Introduction to Statistical Learning with Applications in R. Springer Texts in Statistics, 2021 `https://www.statlearning.com`
- Philippe, Anne. "Notes de Cours sur le logiciel R." Université de Nantes, Laboratoire de Mathématiques Jean Leray (2018)