

DATA VISUALIZATION IN R

Presenter: Nguyen Le Duc Minh, MD

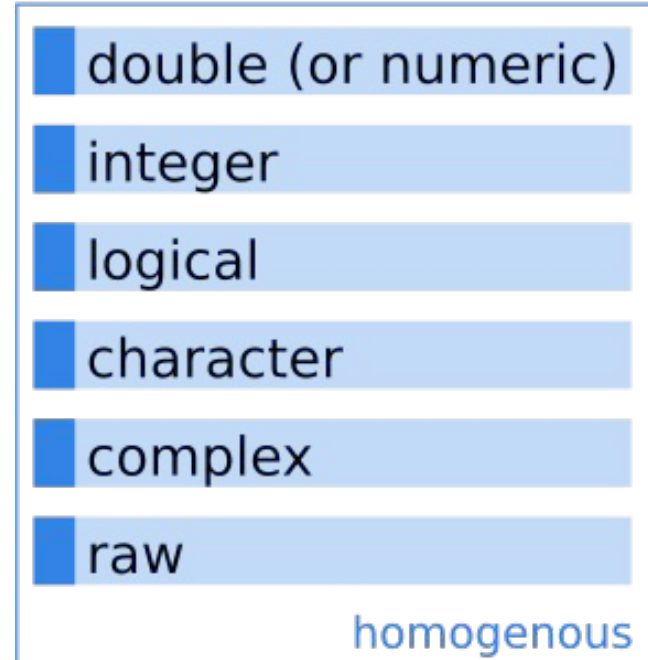
Sep-8th-2024

Contents

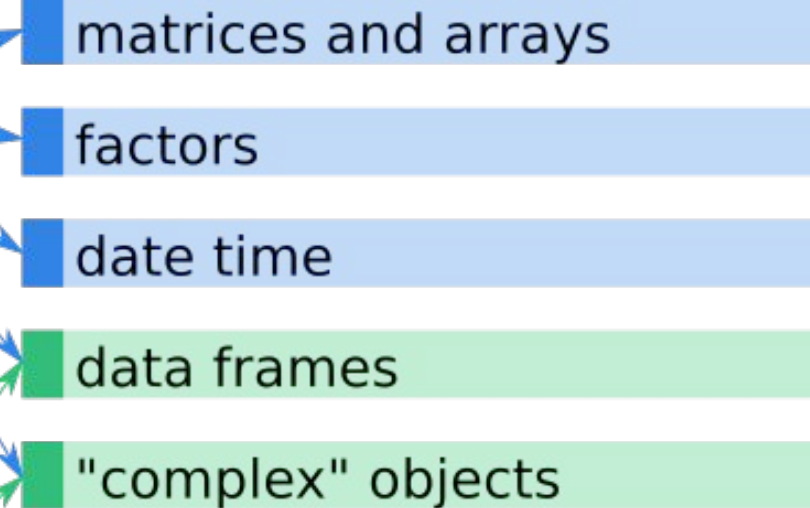
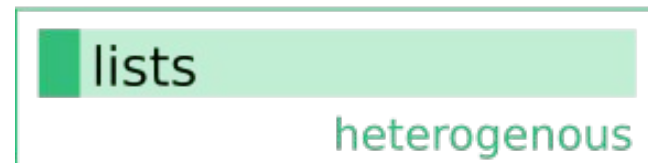
1. Data types
2. Data structure
3. Import and export data
4. Data visualization
5. Homework

DIFFERENT R OBJECTS

Atomic vectors

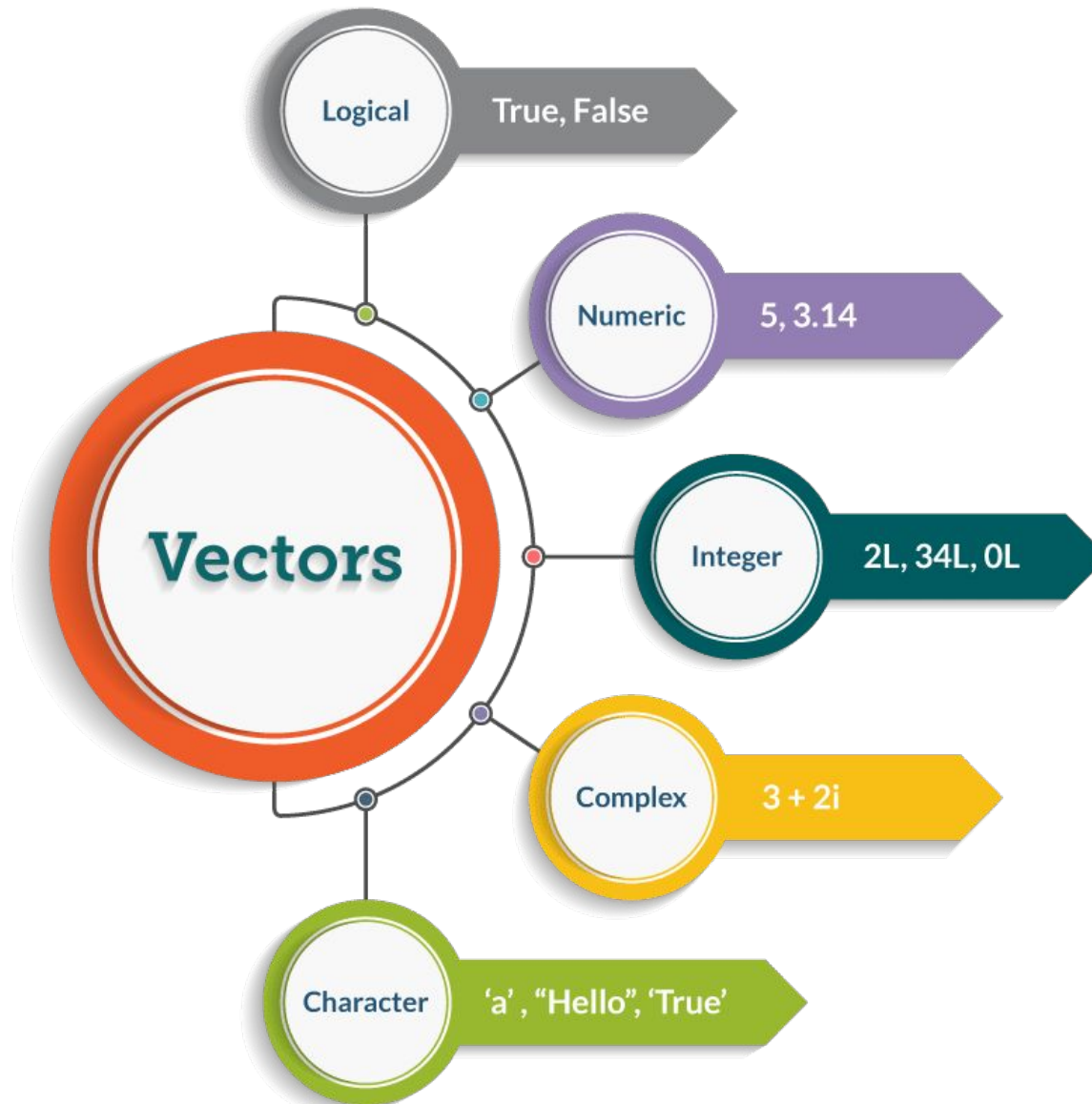


Lists



DATA TYPES IN R

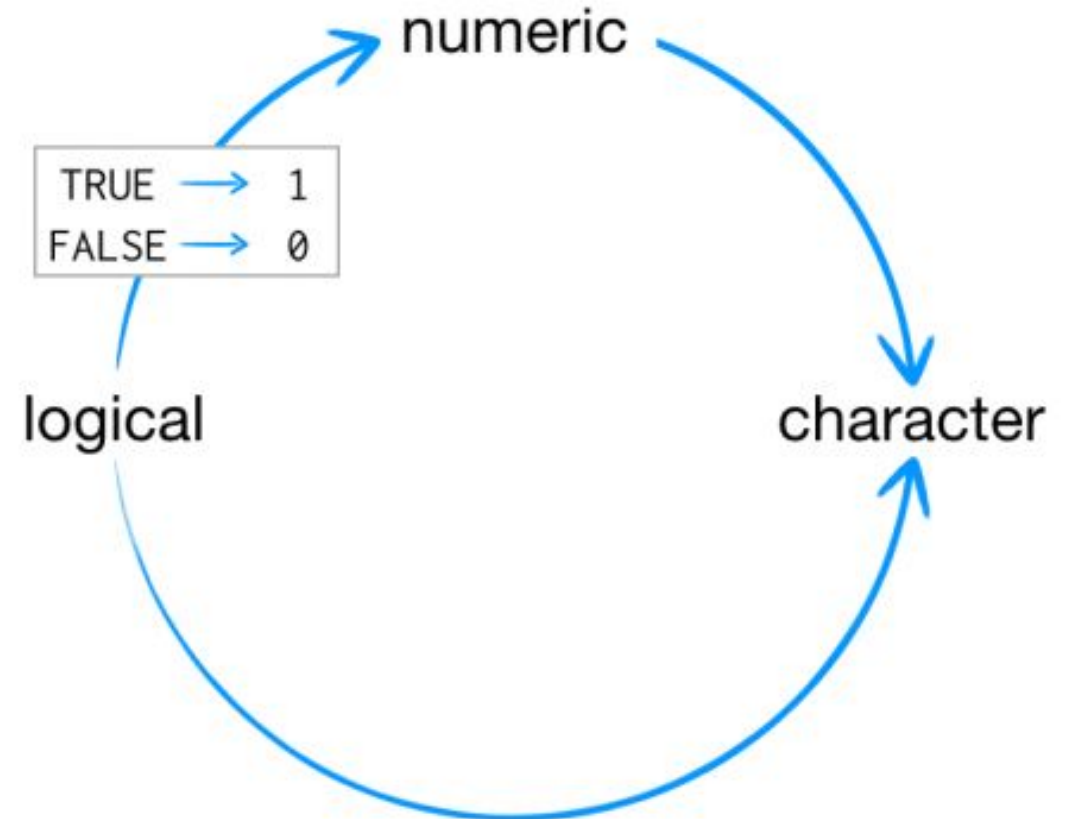
DATA TYPES IN R



What is Coercion in R?

Here's a summary table of some of the logical test and coercion functions available to you.

Type	Logical test	Coercing
Character	<code>is.character</code>	<code>as.character</code>
Numeric	<code>is.numeric</code>	<code>as.numeric</code>
Logical	<code>is.logical</code>	<code>as.logical</code>
Factor	<code>is.factor</code>	<code>as.factor</code>
Complex	<code>is.complex</code>	<code>as.complex</code>



Caculation and Comparison in R

Basic arithmetic and variable assignment

- **Add:** +
- **Subtract:** -
- **Multiply:** *
- **Divide:** /
- **Power:** ^ or **
- **Integer divide:** %/%
- **Modulo (remainder after division):** %%
- **Variable assignment:** = or <-

Comparison and logical operators

- **Equal to:** ==
- **Not equal to:** !=
- **Greater than:** >
- **Less than:** <
- **Greater than or equal to:** >=
- **Less than or equal to:** <=
- **And:** &
- **Or:** |

DATA STRUCTURES IN R

DATA STRUCTURES IN R

Scalars and vectors

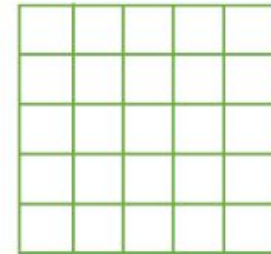


scalar

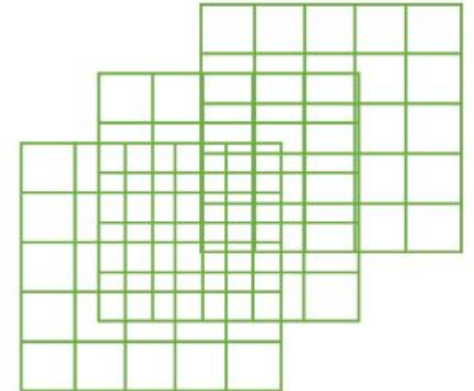


vector

Matrices and arrays



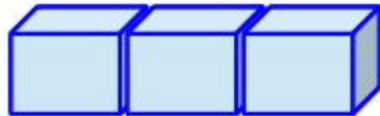
matrix



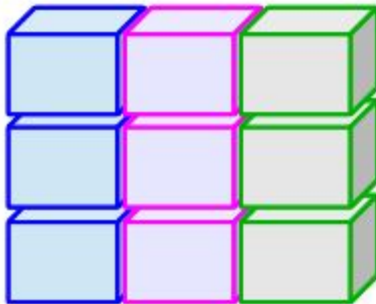
array

DATA STRUCTURES IN R

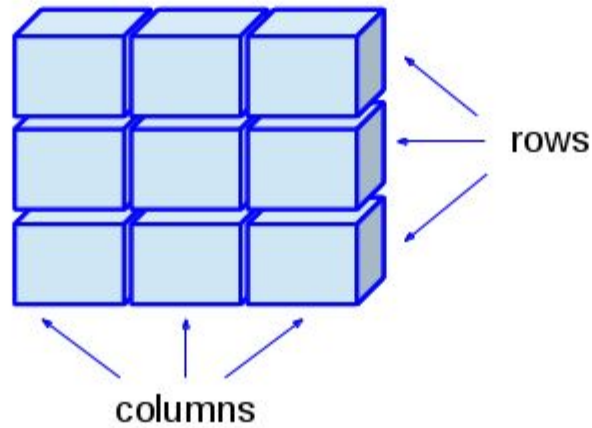
Vector



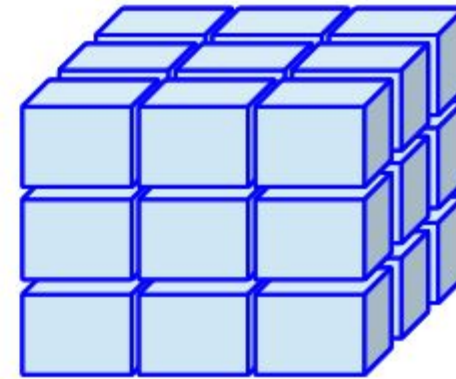
Data Frame
(Table)



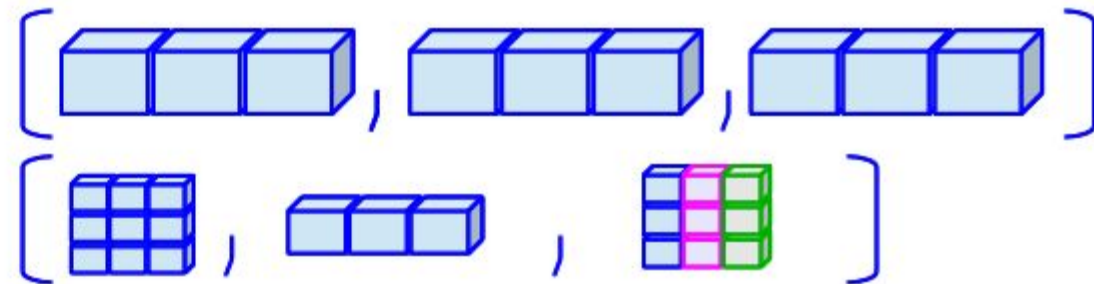
Matrix



Array



Lists



Importing data

FILE TYPES

CSV (Comma-Separated Values): CSV files are plain text files that store tabular data, with each line representing a row of data, and values separated by commas (or other delimiters). CSV is one of the most widely used formats for exchanging data between different applications because of its simplicity and compatibility.

TSV (Tab-Separated Values): TSV files are similar to CSV files but use tabs as the delimiter between values instead of commas. TSV files are also plain text files and are often used when data includes commas within the values.

TXT (Text File): TXT files are plain text files that do not have a standardized format for storing tabular data. They can be used to store any text-based information but do not have specific rules for structuring tabular data like CSV or TSV.

Excel Files (XLS, XLSX): Microsoft Excel files that can store tabular data in multiple sheets along with formatting, formulas, and charts. XLS is the older format used by Excel versions before 2007, while XLSX is the newer XML-based format used by Excel 2007 and later.

CSV

```
Name, Age, Occupation
John, 30, Engineer
Alice, 25, Teacher
Bob, 28, Doctor
```

TSV

```
Name    Age    Occupation
John    30     Engineer
Alice    25     Teacher
Bob      28     Doctor
```

TXT

```
This is a text file.
It can store any text-based information.
```

```
names  sex  age  weight  height
ALFRED  M    14    69     112
BARBARA F    13    62     102
JAMES  M    12    57      83
JANE   F    12    59      84
JOHN   M    12    59      99
JUDY   F    14    64      90
LOUISE F    12    56      77
MARY   F    15    66     112
RONALD M    15    67     133
WILLIAM M    15    66     112
```

READ SINGLE FILE IN R

Importing a CSV file in R

```
data_1 <- read_csv('path_to_file.csv', header=T)
```

```
data_2 <- read_table('path_to_file.csv', header=T)
```

Importing a TXT, TSV file in R

```
data_3 <- read.delim('path_to_file.txt',header = T)
```

Importing data from Excel into R

```
library(readxl)
```

```
data_4 <- read_excel("path_to_file.xlsx", sheet = 1)
```

DEAL WITH MULTI LARGE DATASET

```
install.packages("data.table")
```

```
library(data.table)
```

Why use data.table packages?

- concise syntax: fast to type, fast to read
- fast speed
- memory efficient
- careful API lifecycle management
- community
- feature rich

```
data <- fread('/home/acer/Downloads/download/data.tsv', header=T)
```

<https://www.youtube.com/watch?v=opai4Wmp6Zo>

do.call vs. Reduce

'Reduce' uses a binary function to successively combine the elements of a given vector and a possibly given initial value.

'do.call' constructs and executes a function call from a name or a function and a list of arguments to be passed to it.

```
[[1]]
cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1      0.35      0.5      0.5      0.39      0.15      0.47      0.33      0.08      0.44
cg15639951 cg02192520 cg08675585 cg07447773 subtype
1      0.25      0.67      0.53      0.27      NIFTP

[[2]]
cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1      0.82      0.83      0.83      0.72      0.16      0.28      0.19      0.63      0.79
cg15639951 cg02192520 cg08675585 cg07447773 subtype
1      0.66      0.85      0.87      0.76      FA

[[3]]
cg01796223 cg01091565 cg04705866 cg11484872 cg26607785 cg03409548 cg06454226 cg03001305 cg19282250
1      0.62      0.81      0.82      0.72      0.14      0.18      0.21      0.17      0.47
cg15639951 cg02192520 cg08675585 cg07447773 subtype
1      0.34      0.32      0.86      0.49      FA
```

```
# do.call
data_1 <- do.call("rbind", data_f )

# reduce
data_2<- Reduce(function(x, y) rbind(x,y), data_f)
```

Compare do.call vs. Reduce

do.call () vs. Reduce() allow you to call any R function, but instead of writing out the arguments one by one, you can use a list to hold the arguments of the function.

```
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> Reduce(sum, x)
[1] 55
> do.call(sum, list(x))
[1] 55
> Reduce(function(A, B) sum(c(A, B)), x)
[1] 55
> do.call(function(A, B) sum(c(A, B)), list(x))
Error in (function (A, B) : argument "B" is missing, with no default
> Reduce(function(A, B, C) sum(c(A, B, C)), x)
Error in f(init, x[[i]]) : argument "C" is missing, with no default
```

Do.call

apply function for all elements in list

Reduce

Take 2 elements to apply function for each (only 2 element)

READ MULTI FILES IN R

```
[4.0K] .
├── [2.9K] Petal.Length.tsv
├── [2.9K] Petal.Width.tsv
├── [2.9K] Sepal.Length.tsv
└── [2.8K] Sepal.Width.tsv

0 directories, 4 files
```

ID	Species	Petal.Length
ID_1	setosa	1.4
ID_2	setosa	1.4
ID_3	setosa	1.3
ID_4	setosa	1.5
ID_5	setosa	1.4
ID_6	setosa	1.7
ID_7	setosa	1.4
ID_8	setosa	1.5
ID_9	setosa	1.4

Target

```
head(10)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species ID
1          5.1          3.5          1.4          0.2 setosa ID_1
2          4.9          3.0          1.4          0.2 setosa ID_2
3          4.7          3.2          1.3          0.2 setosa ID_3
4          4.6          3.1          1.5          0.2 setosa ID_4
5          5.0          3.6          1.4          0.2 setosa ID_5
6          5.4          3.9          1.7          0.4 setosa ID_6
```

```
## list path of all file
list_file <- list.files("data/multi_files", pattern = ".tsv", full.names = T)

# read all files
read_file <- lapply(list_file, function(x) read.delim(x, header = T, sep = "\t"))

# merge all files
data <- Reduce(function(x, y) merge(x, y, by = "ID"), read_file)

# Remove duplicate species column
data_rm <- data[!duplicated(as.list(data))]
```

EXPORT DATA IN R

Write data to txt file:

```
write.table(object, file = "name_file.txt", sep = "\t", row.names = TRUE, col.names = NA)
```

Write data to csv files

```
write.csv(object, file = "name_file.csv")
```

Writing data from R to Excel files

```
library("xlsx")
```

```
# Write the first data set in a new workbook
```

```
write.xlsx(USArrests, file = "myworkbook.xlsx", sheetName = "USA-ARRESTS", append = FALSE)
```

```
# Add a second data set in a new worksheet
```

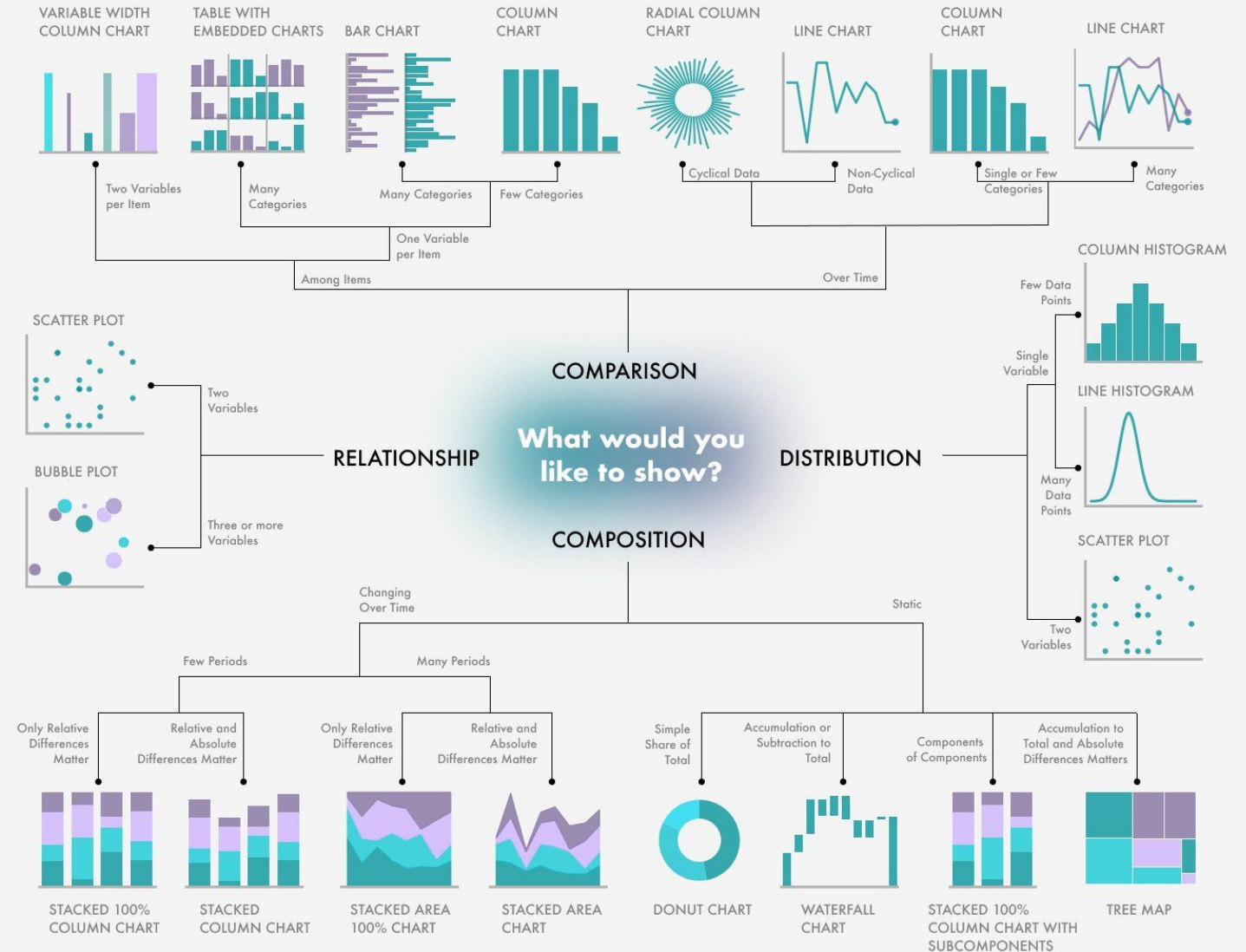
```
write.xlsx(mtcars, file = "myworkbook.xlsx", sheetName="MTCARS", append=TRUE)
```

DATA VISUALIZATION

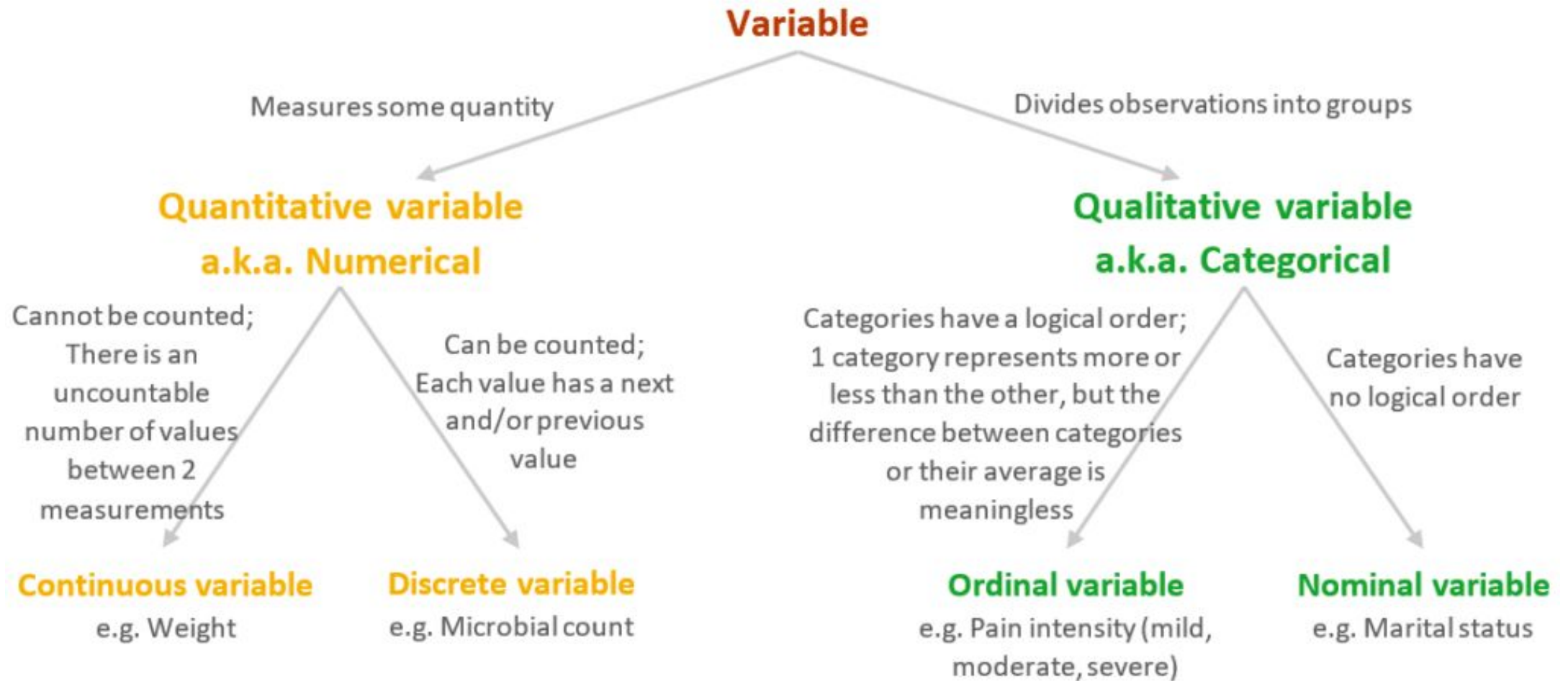
INTRODUCTION

THERE ARE FOUR BASIC PRESENTATION TYPES THAT YOU CAN USE PRESENT YOUR DATA:

1. Comparison
2. Relationship
3. Distribution
4. Composition

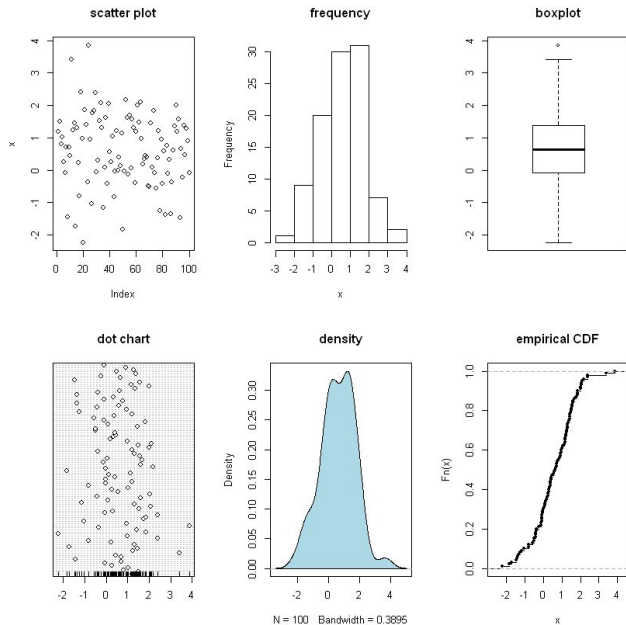


Variable Types

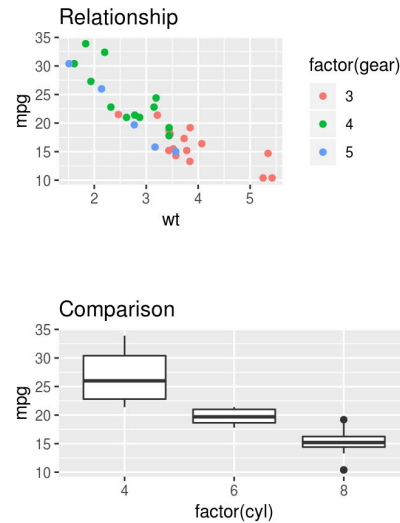


Univariate and Bivariate Graphs

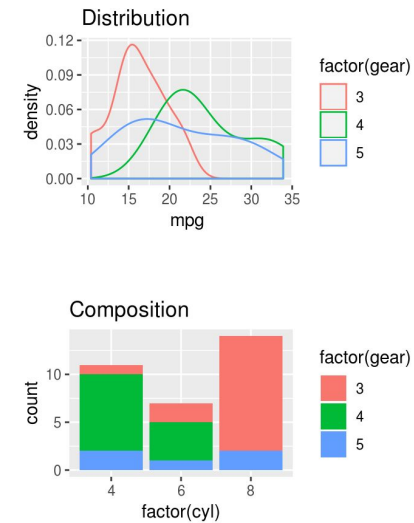
- **Univariate** analysis is when only **one variable** is analyzed.
- **Bivariate** data analysis is when exactly **two variables** are analyzed.
- **Multivariate** analysis is when **more than two variables** get analyzed.



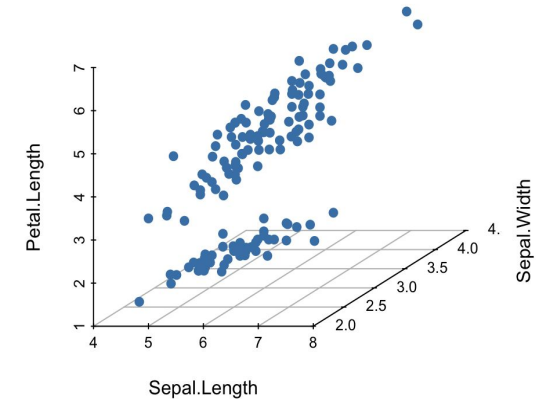
Univariate Graph



Bivariate Graph



Multivariate graph



BASE PLOT AND GG PLOT2

IMPORT DATA INTO R

```
# setup workdir
setwd("/home/ds02/Documents/data_visualization")
getwd()

# 1. Import data
data <- read.csv("data/Heart_Disease_Prediction.csv")
```

```
> head(data)
  index Age Sex Chest.pain.type  BP Cholesterol FBS.over.120 EKG.results Max.HR
1     0  70  1         4 130         322           0           2      109
2     1  67  0         3 115         564           0           2      160
3     2  57  1         2 124         261           0           0      141
4     3  64  1         4 128         263           0           0      105
5     4  74  0         2 120         269           0           2      121
6     5  65  1         4 120         177           0           0      140
  Exercise.angina ST.depression Slope.of.ST Number.of.vessels.fluro Thallium
1              0          2.4           2              3              3
2              0          1.6           2              0              7
3              0          0.3           1              0              7
4              1          0.2           2              1              7
5              1          0.2           1              1              3
6              0          0.4           1              0              7
  Heart.Disease
1      Presence
2      Absence
3      Presence
4      Absence
5      Absence
6      Absence
```


Introduce data in R

This dataset contains **270 case studies of individuals** classified as either having or not having **heart disease** based on results from cardiac catheterizations - the gold standard in heart health assessment.

```
# 'data.frame':  270 obs. of  15 variables:
#  $ index      : int  0 1 2 3 4 5 6 7 8 9 ...
#  $ Age        : int  70 67 57 64 74 65 56 59 60 63 ...
#  $ Sex        : int  1 0 1 1 0 1 1 1 1 0 ...
#  $ Chest.pain.type : int  4 3 2 4 2 4 3 4 4 4 ...
#  $ BP         : int  130 115 124 128 120 120 130 110 140 150 ...
#  $ Cholesterol : int  322 564 261 263 269 177 256 239 293 407 ...
#  $ FBS.over.120 : int  0 0 0 0 0 0 1 0 0 0 ...
#  $ EKG.results  : int  2 2 0 0 2 0 2 2 2 2 ...
#  $ Max.HR       : int  109 160 141 105 121 140 142 142 170 154 ...
#  $ Exercise.angina : int  0 0 0 1 1 0 1 1 0 0 ...
#  $ ST.depression : num  2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2 4 ...
#  $ Slope.of.ST   : int  2 2 1 2 1 1 2 2 2 2 ...
#  $ Number.of.vessels.fluro: int  3 0 0 1 1 0 1 1 2 3 ...
#  $ Thallium      : int  3 7 7 7 3 7 6 7 7 7 ...
#  $ Heart.Disease : chr  "Presence" "Absence" "Presence" "Absence" ...
```

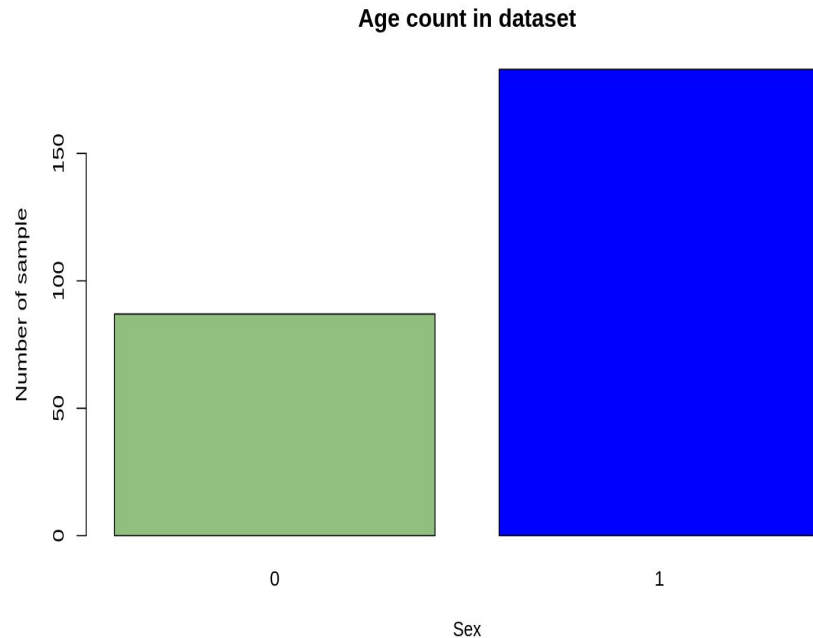
DESCRIBE DATASET

Column name	Description
Age	The age of the patient. (Numeric)
Sex	The gender of the patient. (Categorical)
Chest pain type	The type of chest pain experienced by the patient. (Categorical)
BP	The blood pressure level of the patient. (Numeric)
Cholesterol	The cholesterol level of the patient. (Numeric)
FBS over 120	The fasting blood sugar test results over 120 mg/dl. (Numeric)
EKG results	The electrocardiogram results of the patient. (Categorical)
Max HR	The maximum heart rate levels achieved during exercise testing. (Numeric)
Exercise angina	The angina experienced during exercise testing. (Categorical)
ST depression	The ST depression on an Electrocardiogram. (Numeric)
Slope of ST	The slope of ST segment electrocardiogram readings. (Categorical)
Number of vessels fluro	The amount vessels seen in Fluoroscopy images. (Numeric)
Thallium	The Thallium Stress test findings. (Categorical)
Heart Disease	Whether or not the patient has been diagnosed with Heart Disease. (Categorical)

Barplot

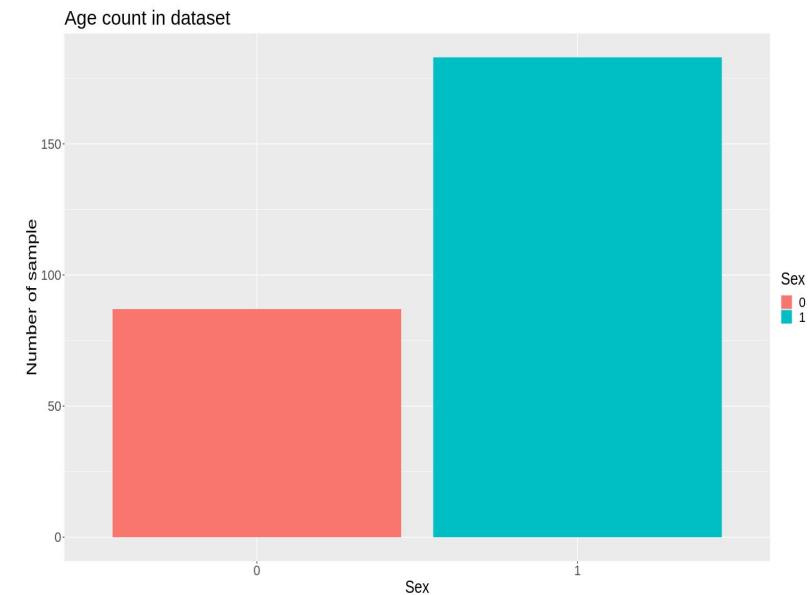
Basic Plot

```
barplot(table(data$Sex),  
  ylab = "Number of sample", # add y label  
  xlab = "Sex", # add x label  
  main = "Age count in dataset", # add title for graph  
  col = c("#90bf80", "blue"), # add color  
  fill = c("pink")  
)
```



ggplot2 Plot

```
ggplot(data, aes(as.factor(Sex), fill = as.factor(Sex))) +  
  geom_bar() + # select a type of plot  
  labs(x = "Sex", y = "Number of sample", title = "Age count in dataset") + # add title, x  
  axis's name, y axis's name  
  guides(fill = guide_legend(title = "Sex")) +  
  theme(text = element_text(size = 20))
```



Reshape2::melt,dcast

```
long_df <- melt(w_data, id = "team")
```

Wide Format

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Long Format

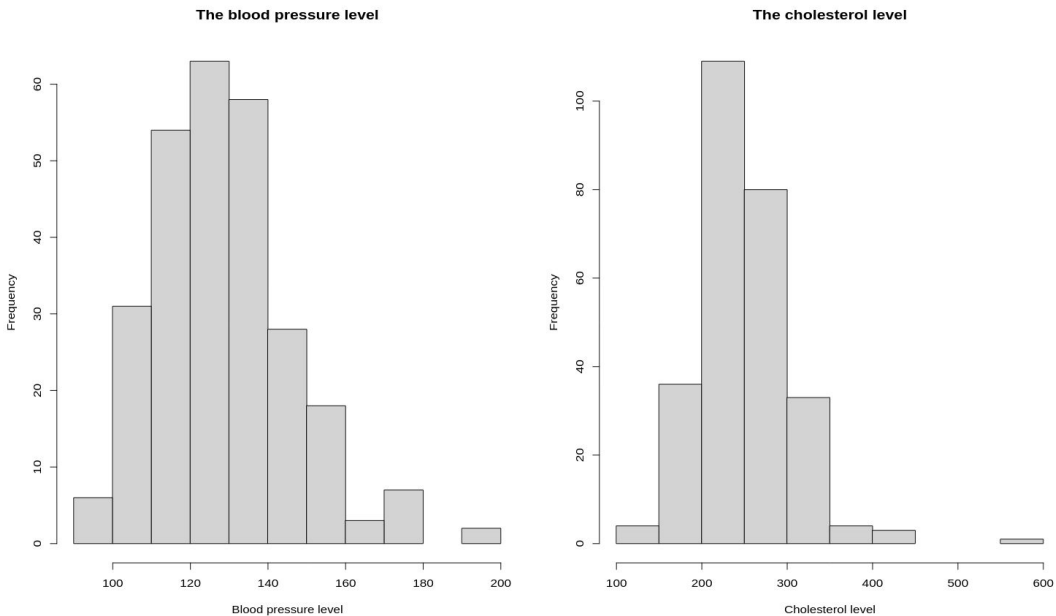
Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

```
rw_data <- dcast(long_df, team ~ variable)
```

Histograms

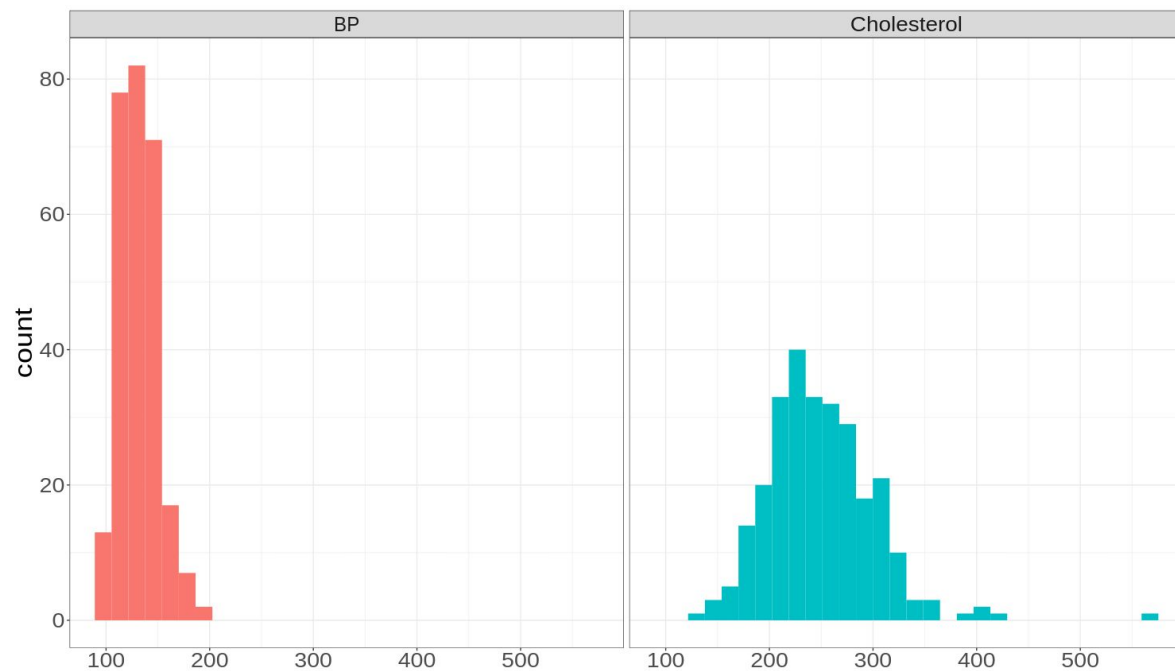
Basic Plot

```
par(mfrow = c(1, 2))
hist(data$BP,
     main = "The blood pressure level",
     xlab = "Blood pressure level"
)
hist(data$Cholesterol,
     main = "The cholesterol level",
     xlab = "Cholesterol level"
)
```



ggplot2 Plot

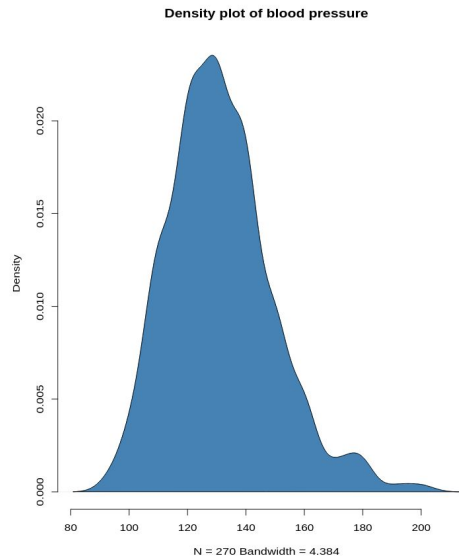
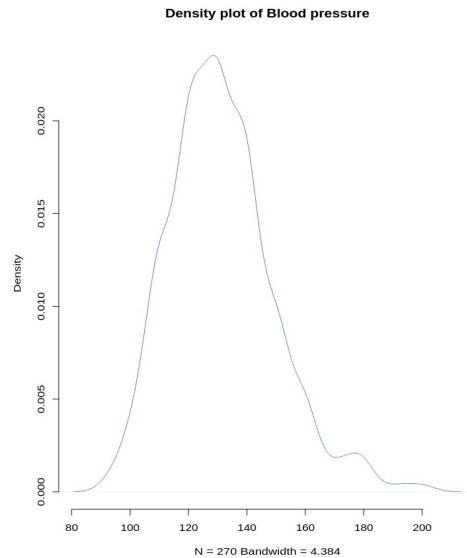
```
ggplot(mdata_sub, aes(value, fill = variable)) +
  theme_bw() +
  geom_histogram() +
  facet_grid(~variable) +
  theme(legend.position = "NULL") +
  labs(x = "") +
  theme(text = element_text(size = 25))
```



Density plot

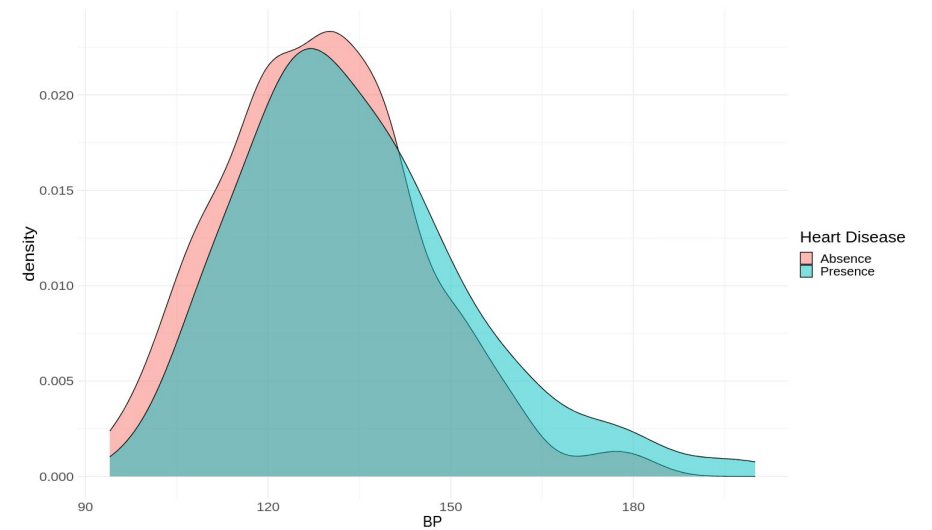
Basic Plot

```
### Compute the density data
dens <- density(data$BP)
par(mfrow = c(1, 2))
### plot density
plot(dens, frame = FALSE, col = "steelblue",
     main = "Density plot of Blood pressure"
)
### Fill the density plot using polygon()
plot(dens, frame = FALSE, col = "steelblue",
     main = "Density plot of blood pressure"
)
polygon(dens, col = "steelblue")
```

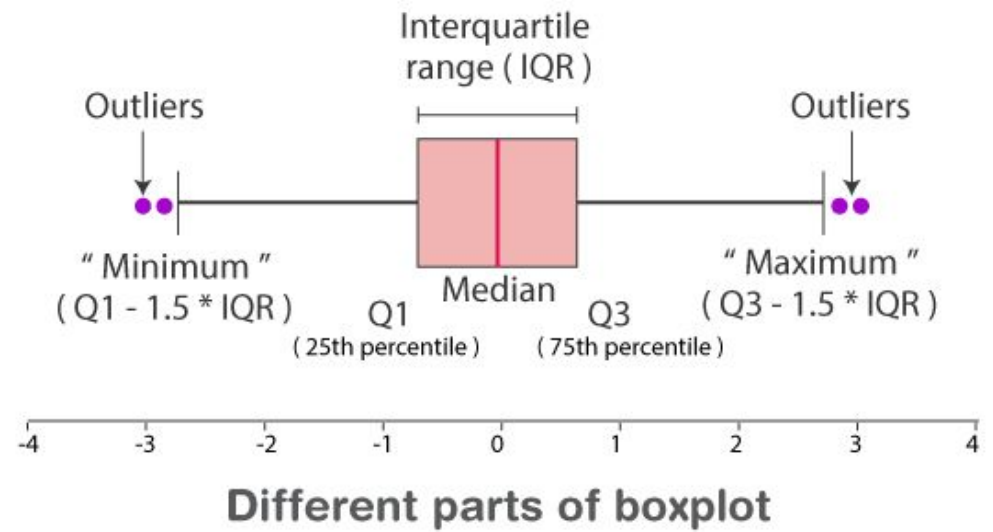
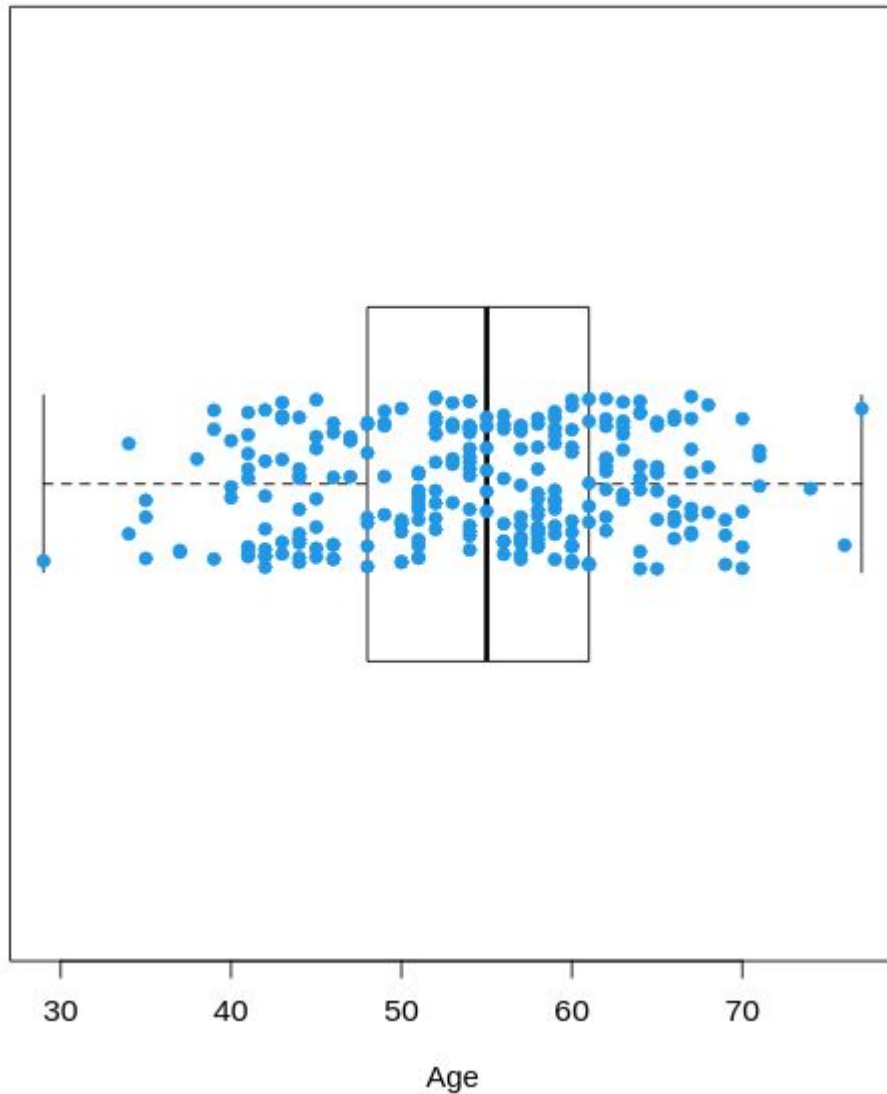


ggplot2 Plot

```
ggplot(data, aes(BP, fill = Heart.Disease)) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  theme(text = element_text(size = 20)) +
  guides(fill = guide_legend(title = "Heart Disease"))
```



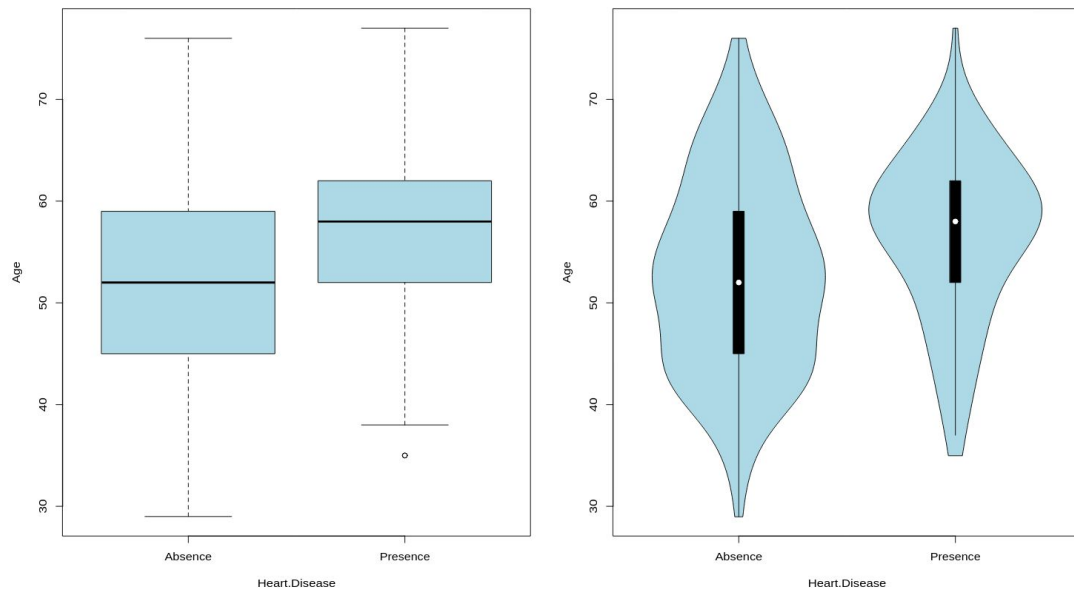
Box and violin plots



Box and violin plots

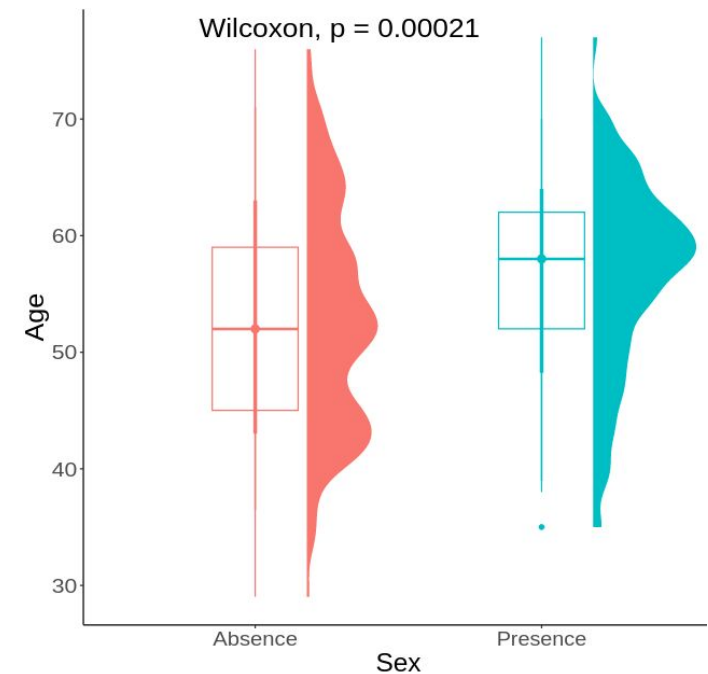
Basic Plot

```
par(mfrow = c(1, 2))  
boxplot(Age ~ Heart.Disease, data = data, col = "lightblue")  
vioplot(Age ~ Heart.Disease, data = data, col = "lightblue")  
dev.off()
```



ggplot2 Plot

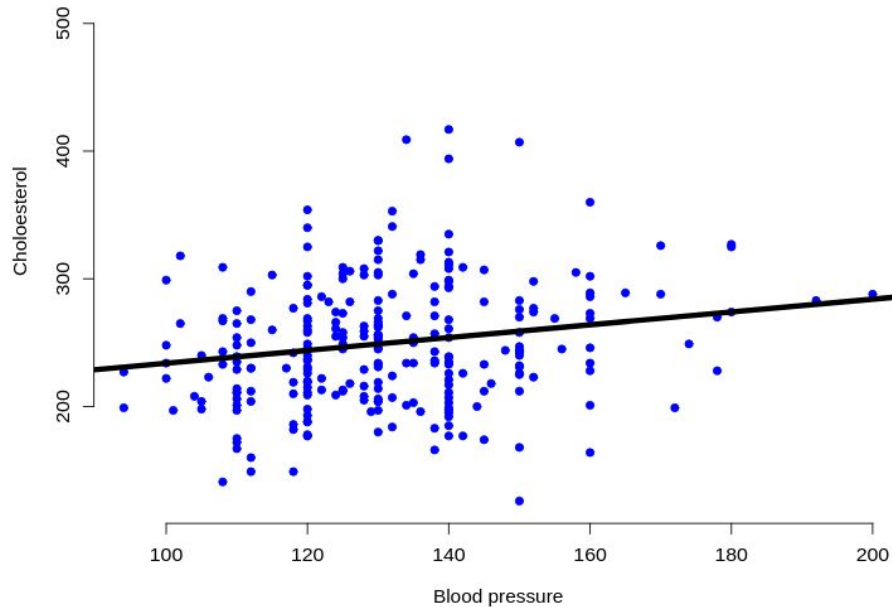
```
ggplot(data, aes(x = Heart.Disease, y = Age, col = Heart.Disease)) +  
  geom_boxplot(width = 0.3) +  
  stat_compare_means(size = 7) +  
  theme_classic() +  
  labs(x = "Sex") +  
  theme(legend.position = "NULL") +  
  theme(text = element_text(size = 20)) +  
  ggdist::stat_halfeye(aes(fill = Heart.Disease),  
    adjust = 0.5, justification = -0.5, width = 0.4)
```



Scatter Plots add text

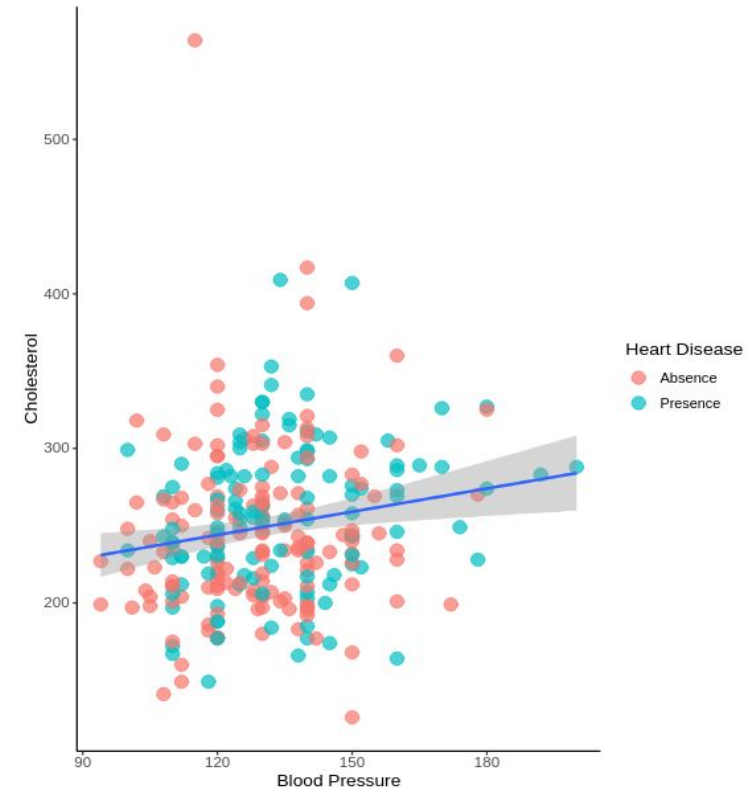
Basic Plot

```
x <- data$BP
y <- data$Cholesterol
plot(x, y,
     xlab = "Blood pressure",
     ylab = "Cholesterol",
     pch = 19, frame = FALSE, col = "blue")
abline(lm(y ~ x, data = data),
      col = "black", lwd = 5, lty = 1)
```



ggplot2 Plot

```
ggplot(data, aes(x = BP, y = Cholesterol)) +
  geom_point(aes(col = Heart.Disease), size = 4, alpha = 0.7) +
  geom_smooth(method = "lm") +
  theme(text = element_text(size = 15)) +
  theme_classic() +
  labs(x = "Blood Pressure", col = "Heart Disease")
```



Save the plot

Method 1

1. Open a pdf file, can add the path

```
pdf("method1.pdf")
```

2. Create a plot

```
ggplot(data, aes(x=names, y=value, fill=names)) + geom_boxplot() + theme(legend.position="NONE")
```

3. Close the pdf file

```
dev.off()
```

Method 2

ggsave

```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
```

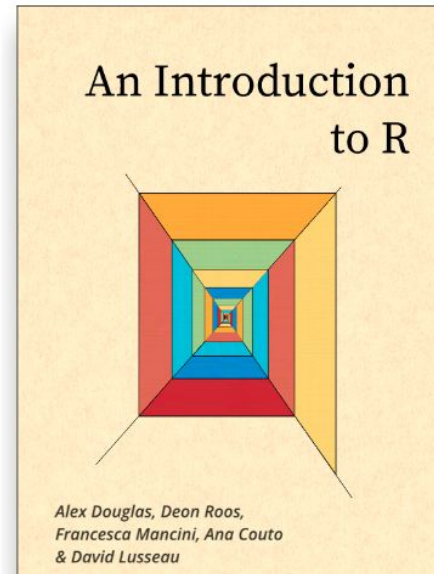
```
ggsave("method2.pdf", width = 6, height = 4)
```

An Introduction to R

Alex Douglas, Deon Roos, Francesca Mancini, Ana Couto & David Lusseau

April 14, 2023

Preface



<https://intro2r.com/>

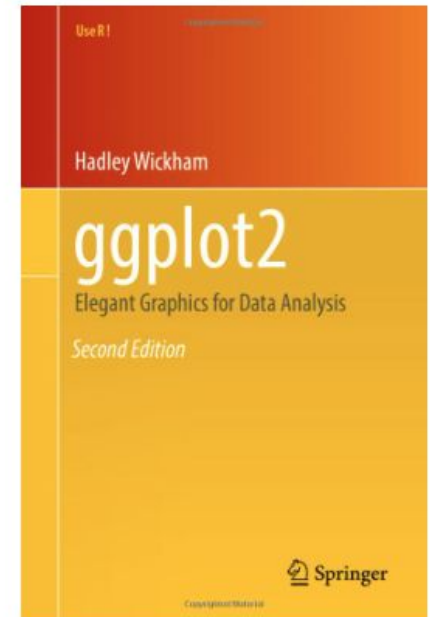
ggplot2: Elegant Graphics for Data Analysis (3e)

Welcome

This is the on-line version of work-in-progress **3rd edition** of “ggplot2: elegant graphics for data analysis” published by Springer. You can learn what’s changed from the 2nd edition in the [Preface](#).

While this book gives some details on the basics of ggplot2, its primary focus is explaining the Grammar of Graphics that ggplot2 uses, and describing the full details. It is not a [cookbook](#), and won’t necessarily help you create any specific graphic that you need. But it will help you understand the details of the underlying theory, giving you the power to tailor any plot specifically to your needs.

The book is written by [Hadley Wickham](#), [Danielle Navarro](#), and [Thomas Lin Pedersen](#).



<https://ggplot2-book.org/>

Cheetsheet

Cheat Sheet for R

Cheat Sheet for ggplot2



R for Data Science

Getting started with R and Cheat Sheet

Learn R online at [www.DataCamp.com](https://www.datacamp.com)

> How to use this cheat sheet

It is one of the most popular programming languages in data science and is widely used across various industries and academics. Thanks to its open source, easy to learn, and capable of handling complex data and statistical modelling, it has become the preferred programming language for many data scientists today.

This Cheat sheet can be considered of getting started with it. It is a handy, high-level reference for a quick start with R, for those desirous of Cheat Sheets. Hence, the highlighted cheat sheets below.




> Accessing help

Accessing help files and [documentation](#)

New R users help documentation for the `foo` function

```
??foo
```

Full version R manuals the documentation for the `foo` package

```
??foo
```

Full version R manuals the documentation for the `foo` package

Full version R manuals the documentation for the `foo` package

Information about objects

`str()` returns the structure and information of a given object

`class()` returns the class of a given object

> Using packages

R packages are collections of functions and code developed by the R community. They increase the power of R by improving existing base functionalities, or by adding new ones.

Install packages ("libraries")

```
install.packages("library")
```

Load packages into your current working directory

```
library(library)
```

Load packages into your current working directory

```
require(library)
```

> The working directory

The working directory is the path that R will use as the starting point for relative file paths. That is, the default location for opening and saving files. An example of a working directory looks like `"C:/Users/John" / "Documents"`.

Print the working directory

```
getwd()
```

Change the working directory to a desired file path

```
setwd("C:/Users/John" / "Documents")
```

> Operators

R has multiple operators that allow you to perform a variety of tasks. Arithmetic operators allow you perform arithmetic tasks such as addition and multiplication. Relational operators are used to compare between values. Logical operators are used for Boolean operations.

Arithmetic Operators	Relational Operators	Logical Operators
1. <code>+</code> Addition	1. <code><</code> Less than	1. <code>&&</code> AND
2. <code>-</code> Subtraction	2. <code><=</code> Less than or equal to	2. <code>&&&</code> OR
3. <code>*</code> Multiplication	3. <code>></code> Greater than	3. <code>&&&&</code> XOR
4. <code>/</code> Division	4. <code>>=</code> Greater than or equal to	4. <code>!&&</code> NOT
5. <code>%</code> Modulus	5. <code>is.na()</code> Is NA	5. <code>is.null()</code> Is null
6. <code>^</code> Exponentiation	6. <code>is.nan()</code> Is NaN	6. <code>is.infinite()</code> Is infinite
7. <code>~</code> Negation	7. <code>is.nan()</code> Is NaN	7. <code>is.infinite()</code> Is infinite
8. <code>abs()</code> Absolute value	8. <code>is.nan()</code> Is NaN	8. <code>is.infinite()</code> Is infinite
9. <code>sqrt()</code> Square root	9. <code>is.nan()</code> Is NaN	9. <code>is.infinite()</code> Is infinite
10. <code>log()</code> Natural log	10. <code>is.nan()</code> Is NaN	10. <code>is.infinite()</code> Is infinite
11. <code>log10()</code> Base 10 log	11. <code>is.nan()</code> Is NaN	11. <code>is.infinite()</code> Is infinite
12. <code>exp()</code> Exponential	12. <code>is.nan()</code> Is NaN	12. <code>is.infinite()</code> Is infinite
13. <code>expm1()</code> Exponential minus one	13. <code>is.nan()</code> Is NaN	13. <code>is.infinite()</code> Is infinite
14. <code>log2()</code> Base 2 log	14. <code>is.nan()</code> Is NaN	14. <code>is.infinite()</code> Is infinite
15. <code>log1p()</code> Log one plus	15. <code>is.nan()</code> Is NaN	15. <code>is.infinite()</code> Is infinite
16. <code>loglik()</code> Log likelihood	16. <code>is.nan()</code> Is NaN	16. <code>is.infinite()</code> Is infinite
17. <code>logdet()</code> Log determinant	17. <code>is.nan()</code> Is NaN	17. <code>is.infinite()</code> Is infinite
18. <code>loggamma()</code> Log gamma	18. <code>is.nan()</code> Is NaN	18. <code>is.infinite()</code> Is infinite
19. <code>logbeta()</code> Log beta	19. <code>is.nan()</code> Is NaN	19. <code>is.infinite()</code> Is infinite
20. <code>logpoisson()</code> Log Poisson	20. <code>is.nan()</code> Is NaN	20. <code>is.infinite()</code> Is infinite
21. <code>logbinom()</code> Log Binomial	21. <code>is.nan()</code> Is NaN	21. <code>is.infinite()</code> Is infinite
22. <code>logmultinom()</code> Log Multinomial	22. <code>is.nan()</code> Is NaN	22. <code>is.infinite()</code> Is infinite
23. <code>loggamma()</code> Log gamma	23. <code>is.nan()</code> Is NaN	23. <code>is.infinite()</code> Is infinite
24. <code>logbeta()</code> Log beta	24. <code>is.nan()</code> Is NaN	24. <code>is.infinite()</code> Is infinite
25. <code>logpoisson()</code> Log Poisson	25. <code>is.nan()</code> Is NaN	25. <code>is.infinite()</code> Is infinite
26. <code>logbinom()</code> Log Binomial	26. <code>is.nan()</code> Is NaN	26. <code>is.infinite()</code> Is infinite
27. <code>logmultinom()</code> Log Multinomial	27. <code>is.nan()</code> Is NaN	27. <code>is.infinite()</code> Is infinite
28. <code>loggamma()</code> Log gamma	28. <code>is.nan()</code> Is NaN	28. <code>is.infinite()</code> Is infinite
29. <code>logbeta()</code> Log beta	29. <code>is.nan()</code> Is NaN	29. <code>is.infinite()</code> Is infinite
30. <code>logpoisson()</code> Log Poisson	30. <code>is.nan()</code> Is NaN	30. <code>is.infinite()</code> Is infinite
31. <code>logbinom()</code> Log Binomial	31. <code>is.nan()</code> Is NaN	31. <code>is.infinite()</code> Is infinite
32. <code>logmultinom()</code> Log Multinomial	32. <code>is.nan()</code> Is NaN	32. <code>is.infinite()</code> Is infinite
33. <code>loggamma()</code> Log gamma	33. <code>is.nan()</code> Is NaN	33. <code>is.infinite()</code> Is infinite
34. <code>logbeta()</code> Log beta	34. <code>is.nan()</code> Is NaN	34. <code>is.infinite()</code> Is infinite
35. <code>logpoisson()</code> Log Poisson	35. <code>is.nan()</code> Is NaN	35. <code>is.infinite()</code> Is infinite
36. <code>logbinom()</code> Log Binomial	36. <code>is.nan()</code> Is NaN	36. <code>is.infinite()</code> Is infinite
37. <code>logmultinom()</code> Log Multinomial	37. <code>is.nan()</code> Is NaN	37. <code>is.infinite()</code> Is infinite
38. <code>loggamma()</code> Log gamma	38. <code>is.nan()</code> Is NaN	38. <code>is.infinite()</code> Is infinite
39. <code>logbeta()</code> Log beta	39. <code>is.nan()</code> Is NaN	39. <code>is.infinite()</code> Is infinite
40. <code>logpoisson()</code> Log Poisson	40. <code>is.nan()</code> Is NaN	40. <code>is.infinite()</code> Is infinite
41. <code>logbinom()</code> Log Binomial	41. <code>is.nan()</code> Is NaN	41. <code>is.infinite()</code> Is infinite
42. <code>logmultinom()</code> Log Multinomial	42. <code>is.nan()</code> Is NaN	42. <code>is.infinite()</code> Is infinite
43. <code>loggamma()</code> Log gamma	43. <code>is.nan()</code> Is NaN	43. <code>is.infinite()</code> Is infinite
44. <code>logbeta()</code> Log beta	44. <code>is.nan()</code> Is NaN	44. <code>is.infinite()</code> Is infinite
45. <code>logpoisson()</code> Log Poisson	45. <code>is.nan()</code> Is NaN	45. <code>is.infinite()</code> Is infinite
46. <code>logbinom()</code> Log Binomial	46. <code>is.nan()</code> Is NaN	46. <code>is.infinite()</code> Is infinite
47. <code>logmultinom()</code> Log Multinomial	47. <code>is.nan()</code> Is NaN	47. <code>is.infinite()</code> Is infinite
48. <code>loggamma()</code> Log gamma	48. <code>is.nan()</code> Is NaN	48. <code>is.infinite()</code> Is infinite
49. <code>logbeta()</code> Log beta	49. <code>is.nan()</code> Is NaN	49. <code>is.infinite()</code> Is infinite
50. <code>log</code>		

[illegible]

> Getting started with Data Frames in R

A data frame has the attributes of a data set and so columns and the observations as rows.

It often creates the data frame `z`, such as on the right
`z <- data.frame(x = 1:5, y = c("a", "b", "c", "d", "e"), z = 10:14)`

`z`

often selects all rows of the second column
`z[,2]`

a	10
b	11
c	12
d	13
e	14

often selects all values of the third row
`z[3,]`

`z`

often selects the third column of the second row
`z[2,3]`

a	10	12
b	11	13
c	12	14
d	13	15
e	14	16

often selects the column `x`
`z[,x]`

`z`

often selects the column `x`
`z[,x]`

a	10
b	11
c	12
d	13
e	14

> Manipulating Data Frames in R

(SQL allows us to easily and quickly manipulate data frames. In fact, the following functions, you should install and load first" using `install.packages("dplyr")`

filter a number of rows
 vector or data frame arguments
 the columns that to columns

`filter`

`z`

often selects to a new position
`release(z[, "age"] < 40)`
`z`

`z`

`z`

filter a number of values
 vector or data frame arguments
 and conditions that to rows

`filter`

`z`

often selects the value of a column from data to use
`arrange(z[, desc(x)])`

`z`

`z`

often uses that most logical criteria
`filter(x, x < 2)`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

often row with duplicate values
`distinct(z[, 2])`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

often row by position
`slice(z[, 10:20])`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

often rows with the largest value
`slice_max(z[, x, nrow = 5])`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

often selects a value
 by row or column
`pull(z[, x])`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

`z`

Data Visualization with ggplot2: : CHEAT SHEET

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.

data (geom) + coordinate system = plot

data: $x=F, y=A$
coordinate system: $x=F, y=A$
plot: $x=F, y=A$

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

data (geom) + coordinate system = plot

data: $x=F, y=A$
coordinate system: $x=F, y=A$
plot: $x=F, y=A$

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +
  <REQUIRED>
  <AESTHETIC MAPPINGS> (mapping = aes(<MAPPINGS>))
  <STAT> (stat = <POSITION>)
  <SCALE FUNCTIONS> +
  <THEME FUNCTION>
```

Test required, sensible defaults supplied

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

Aesthetic mappings **data** **geom**

plot() $x = \text{ct}, y = \text{hwy}$, data = mpg, geom = "point". Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5"x5" file named "plot.png" in working directory. Matches file type to file extension.

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

a = geom_blank() (Useful for expanding limits)

b = geom_curve(aes(yend = lat + 1, xend=long+1, curvature=2)) x, aend, y, yend, alpha, angle, color, curvature, linetype, size

c = geom_path(linetype="butt", linjoin="round", linetrend=1) x, y, alpha, color, group, linetype, size

d = geom_polygon(aes(group = group)) x, y, alpha, color, fill, group, linetype, size

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

f = geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) x, y, 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(intercept = lat))
b = geom_vline(aes(intercept=long))
b = geom_segment(aes(yend=lat+1, xend=long+1))
b = geom_spoke(aes(angle = 1:155, 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, weight

c2 = 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(hwy))
```

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

TWO VARIABLES

continuous x, continuous y

```
e <- ggplot(mpg, aes(ct, hwy))
```

e = geom_label(aes(label = ct), nudge_x = 1, nudge_y = 1, check.overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

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

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 = "b"), x, y, alpha, color, linetype, size, weight

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

e = geom_text(aes(label = ct), nudge_x = 1, nudge_y = 1, check.overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, linetype, size, vjust

discrete x, continuous y

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

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

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

discrete x, discrete y

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

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

THREE VARIABLES

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

h = geom_contour(aes(z = z)) x, y, z, alpha, color, group, linetype, size, weight

h = geom_bin2d(binwidth = c(0.25, 0.05)) x, y, alpha, color, fill, linetype, size, weight

h = geom_density2d() x, y, alpha, color, group, linetype, size

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

continuous bivariate distribution

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

h = geom_bin2d(binwidth = c(0.25, 0.05)) x, y, alpha, color, fill, linetype, size, weight

h = geom_density2d() 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

```
d <- data.frame(murder = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(d, aes(fit, fit_min = fit-se, fit_max = fit+se))
```

j = geom_crossbar(aes(x = 2, y = ymax, ymin, alpha, color, fill, group, linetype, size)

j = geom_errorbar() x, y, ymax, ymin, alpha, color, fill, group, linetype, size, width (also geom_errorbarh)

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

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

maps

```
map_data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
k <- ggplot(map_data, state)
l <- ggplot(data, aes(fill = murder))
```

k = geom_map(map = map, id = state, map = map) + expand_limits(lim = map\$long, y = map\$lat), map_id, alpha, color, fill, linetype, size

l = geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)

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



Thank you

Homework

