“R” Tutorial on Statistical Manipulation of DATA

by

Faizan Ali

Abdullah Rizvi

Ammar Ateeq

Rahul Mondal

Contents

After this Tutorial, you should be able to

- ★ Have a basic understanding of “R”
- ★ Install the proper & latest package of “R” on your system and run your first script
- ★ Understand the data types used in “R”
- ★ Understand packages used in “R” to manipulate your data for pre-processing and post-processing

R Programming

Intro, why and Packages

What is R?

- R is a programming language developed by Ross Ihaka and Robert Gentleman in 1993.
- R possesses an extensive catalog of statistical and graphical methods.
- It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

What is R used for?

- Statistical inference
- Numerical simulation
- Data analysis
- Data Visualization
- Machine learning algorithm

Why Learn R?

- Open-source Language
- Cross-platform compatible
- Advanced Statistical Language
- Outstanding Graphs
- Big Community!
- Extremely Comprehensive Flexible & Fun!

Installing R

STEP-1 (Install R)

- Open the link (https://www.r-project.org/)
- Goto --> Download R as prescribed

[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 4.0.0 (Arbor Day) prerelease versions** will appear starting Tuesday 2020-03-24. Final release is scheduled for Friday 2020-04-24.
- **R version 3.6.3 (Holding the Windsock)** has been released on 2020-02-29.

Installing R

- Goto --> O-Cloud and click

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics here: main page, windows release, windows old release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

0-Cloud	
https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently s
Algeria	
https://cran.usthb.dz/	University of Science and Technology Houari Boumedi
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	

- Choose your operating system and click it

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Installing R

STEP-2 (Install R-Studio)

- Click on the link(https://rstudio.com/products/rstudio/) or visit R-Studio website
- GoTo --> R-Studio for Desktop as shown below

	• Integrated R help and documentation • Interactive debugger to diagnose and fix errors quickly • Extensive package development tools	• RStudio Professional Drivers
Support	Community forums only	• Priority Email Support • 8 hour response during business hours (ET)
License	AGPL v3	RStudio License Agreement
Pricing	Free	$995/year
	DOWNLOAD RSTUDIO DESKTOP	DOWNLOAD FREE RSTUDIO DESKTOP PRO TRIAL

Installing R

- Select your Operating System and Download the package

OS	Download	Size
Windows 10/8/7	RStudio-1.2.5033.exe	149.83 MB
macOS 10.13+	RStudio-1.2.5033.dmg	126.89 MB
Ubuntu 14/Debian 8	rstudio-1.2.5033-amd64.deb	96.18 MB
Ubuntu 16	rstudio-1.2.5033-amd64.deb	104.14 MB
Ubuntu 18/Debian 10	rstudio-1.2.5033-amd64.deb	105.21 MB
Fedora 19/Red Hat 7	rstudio-1.2.5033-x86_64.rpm	120.23 MB
Fedora 28/Red Hat 8	rstudio-1.2.5033-x86_64.rpm	120.87 MB

- Run the Package(better to run as Administrator for Windows)

Congratulations! You have successfully installed required softwares to work with R

R Packages

- Many useful R functions come in packages
- They increase the power of R by improving existing base R functionalities
- They bundle together code, data, documentation, and tests
- There are thousands of packages available on the **C**omprehensive **R** **A**rchive **N**etwork, or CRAN
- Commonly used R Packages are:
  - Dplyr, tidyr, ggplot2, shiny etc.

R Packages

To install an R package, type following in the command line:

```
install.packages("<the package's name>")
```

R will download the package from CRAN

After it is installed, you can make its contents available by running:

```
library("<the package's name>")
```

You can also get help on them by *help(package =“<the package’s name>”)*

????

DATA TYPES IN R?

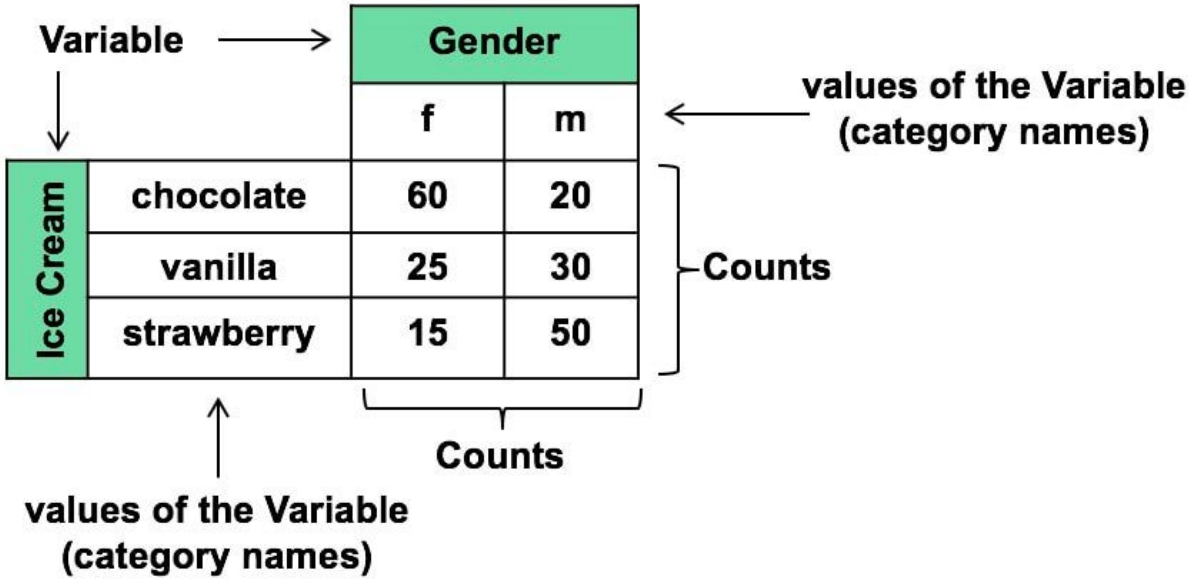
- There are several Data types depending upon the nature of work one has do. Following is a list in which you can store data.
  - **Vector**
  - **Factor**
  - **Array**
  - **Matrix**
  - **Data Frame**
  - **TS**
  - **List**

Vector

- Contains a sequence of items of the same type. This is most basic structure, e.g. x <- 10 or name <- ‘Harry’

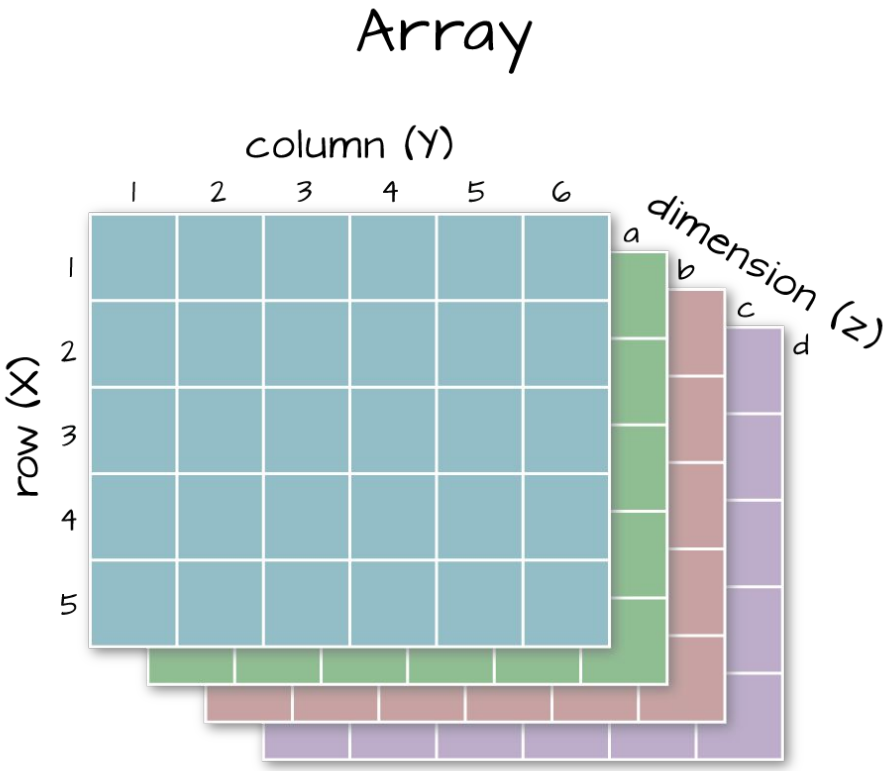
Factor

- A Factor is a Categorical Variable.
- A Categorical variable represents types of attributes. e.g. Gender: Male, Female or Flavors of ice cream: Chocolate, Vanilla, Strawberry.



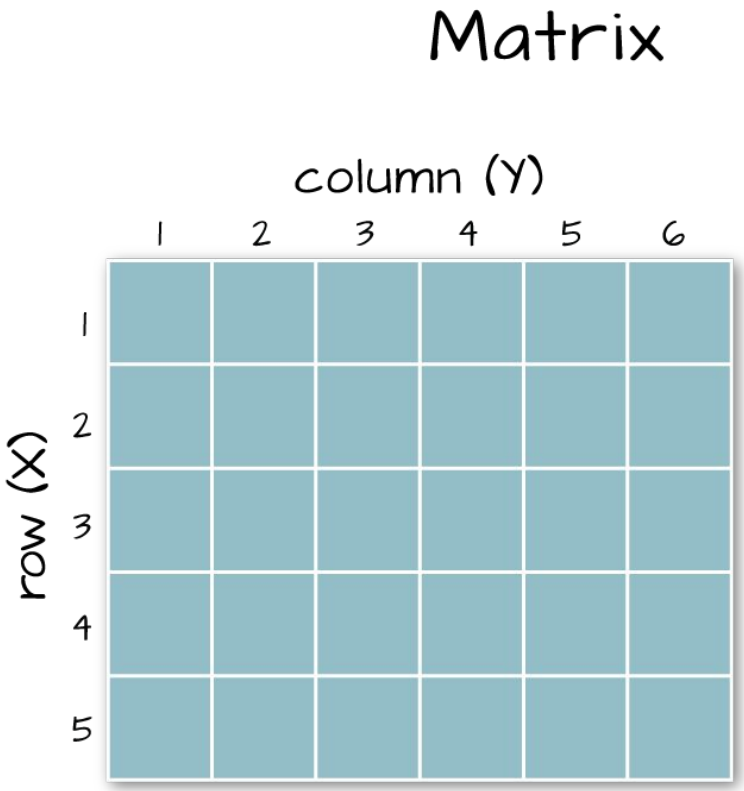
Array

- An Array is a table with ‘k’ dimensions. Usually used to store data in a table format. If we create an array of dimensions (2,3,4) then 4 rectangular matrices will be created, each with 2 rows and 3 columns.



Matrix

- A Matrix is an Array, but having specifically two dimensions, e.g. ‘k=2’.

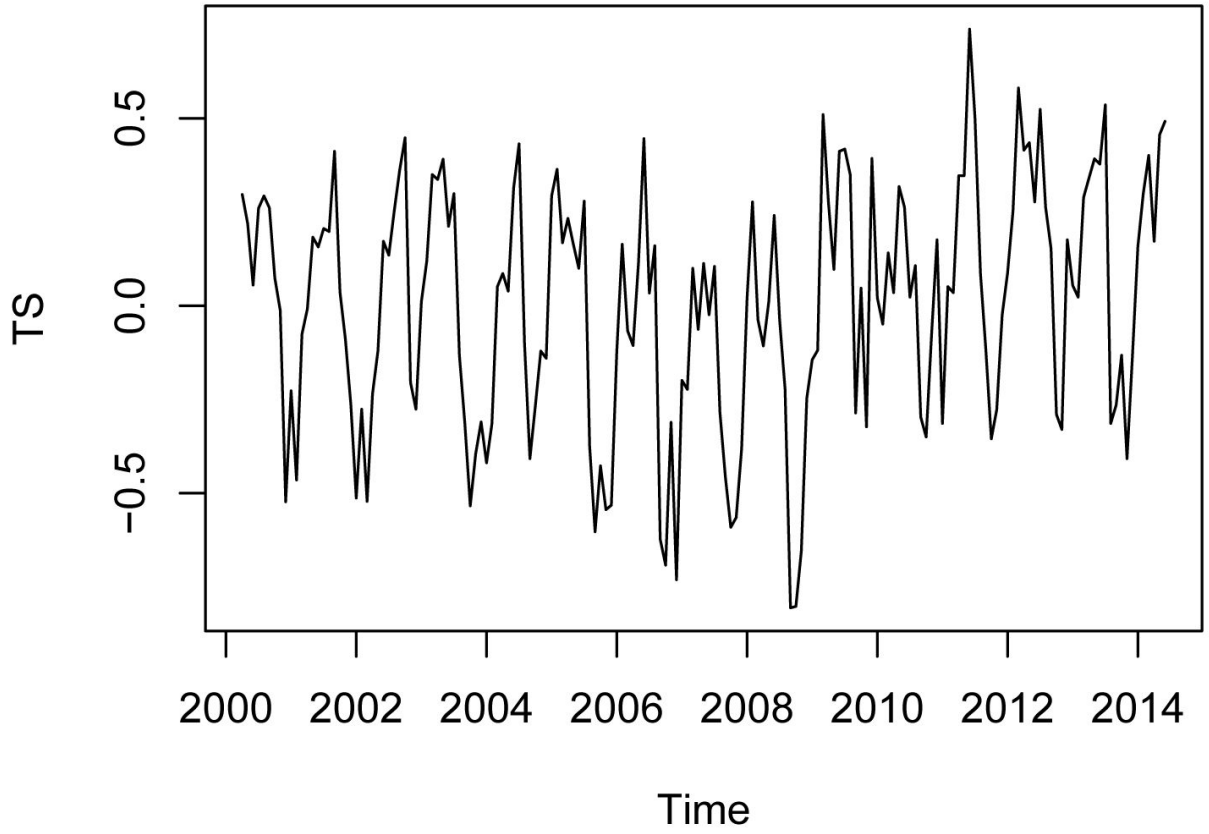


Data Frame

- A data frame is a table composed with one or several vectors and/or factors all of the same length but possibly of different modes. A data frame may contain multiple arrays.

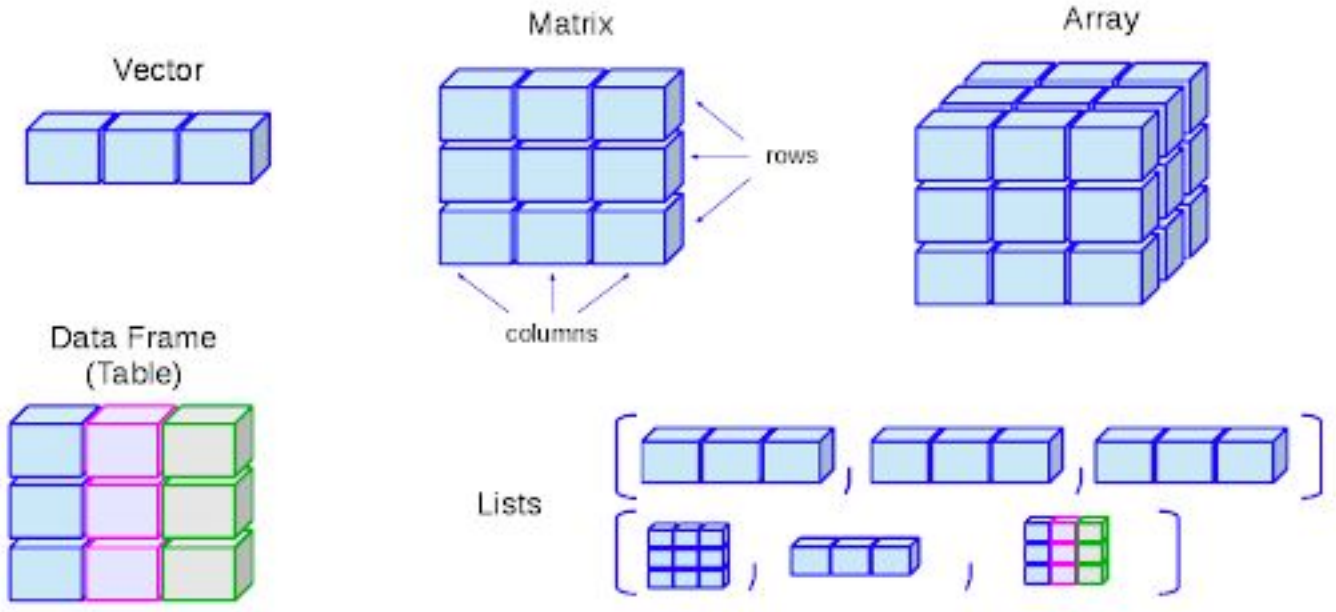
TS – Time Series

- A TS is a Time Series dataset, additionally it contains attributes like frequency and dates.



List

- A list is the ultimate data type to store every element, including vectors, factors, data frames and even list itself.



Data Types and Their Modes

object	modes	several modes possible in the same object?
vector	numeric, character, complex *or* logical	No
factor	numeric *or* character	No
array	numeric, character, complex *or* logical	No
matrix	numeric, character, complex *or* logical	No
data frame	numeric, character, complex *or* logical	Yes
ts	numeric, character, complex *or* logical	No
list	numeric, character, complex, logical, function, expression, . . .	Yes

READING DATA IN R

INPUT

- The input format is quite easy. We just have to know the path of the file.

- df <- read.csv("C:/Desktop/airquality.csv")

- “header = True” in read.csv is a logical value, indicating whether the file contains the names of the variables in the first line.

OUTPUT

- write.csv(Your DataFrame,"Path where you'd like to export the Data Frame \\File Name.csv", row.names = FALSE)

- **Formally**: write.csv(df,"C:\\Users\\Ron\\Desktop\\MyData.csv", row.names = FALSE)

- row.names is used to name the first column of the dataset. It defines the name of rows.

QUICK TASK

- Download the data from here: (https://drive.google.com/open?id=1I1pKFcObgkGoHtm6twRYoqC4-aJLMUts)
- Input the dataset via R command.
- Output the dataset via R command.

????

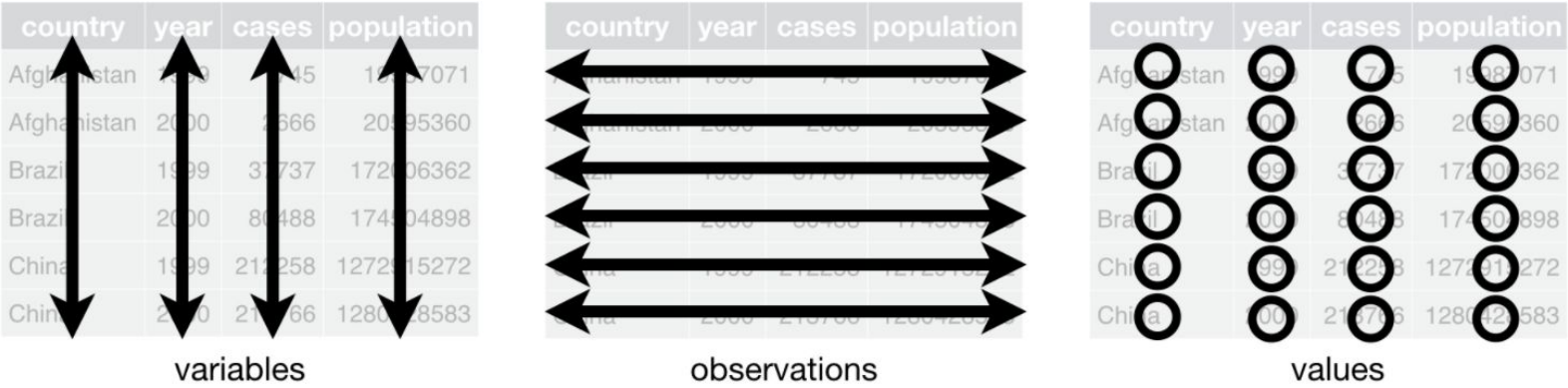
Handle Tidy Data

What is Tidy Data?

A data set is called **tidy** when:

- each column represents a variable
- and each row represents an observation

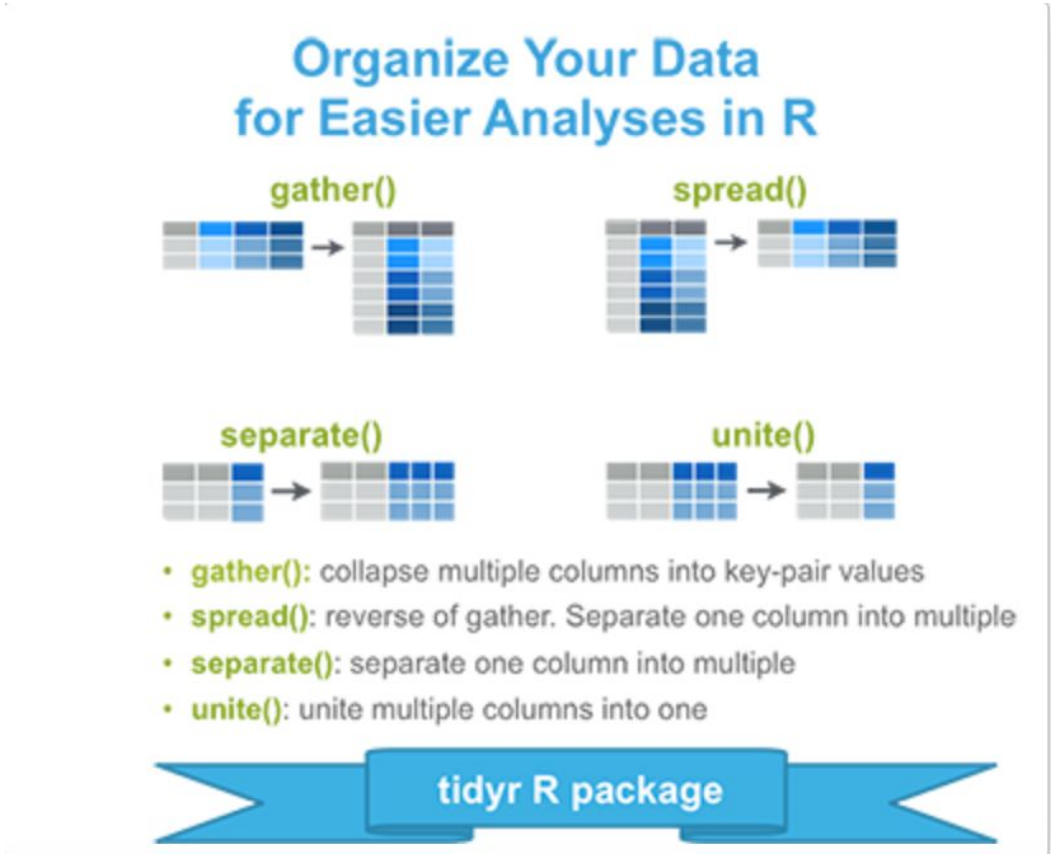
- The opposite of **tidy** is **messy data**, which corresponds to any other arrangement of the data.



Package to Manipulate Tidy-Data? (Tidyr)

- **Organize** (or **reshape**) your data in order to make the analysis easier. This process is called **tidying your data.**

1. gather(): gather (collapse) columns into rows
2. spread(): spread rows into columns.
3. separate(): separate one column into multiple
4. unite(): unite multiple columns into one



Hands-on Tidyr:

```
# Installing
install.packages("tidyr")
# Loading
library("tidyr")
# Load-Dataset
my-data <- USArrests[c(1, 10, 20, 30),  ]    # [c(1,10,20,30), ] means specific rows and all columns
```

```
> my_data
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Georgia      17.4     211       60 25.8
Maryland     11.3     300       67 27.8
New Jersey    7.4     159       89 18.8
```

Row names are states, so let's use the function cbind() to add a column named "state"

In the data. This will make the data tidy and the analysis easier.

```
My_data <- cbind(state = rownames(my_data), my_data)
my_data
```

States as a columns

```
> my_data
                state Murder Assault UrbanPop Rape
Alabama       Alabama   13.2     236       58 21.2
Georgia       Georgia   17.4     211       60 25.8
Maryland     Maryland   11.3     300       67 27.8
New Jersey New Jersey    7.4     159       89 18.8
```

gather(): collapse columns into rows
[Replacement of MELT(reshape2)]

Simplified format:

gather(data, key, value, ...)

data: A data frame

key: Names of key

value: Value columns to create in output

…: Specification of columns to gather.

Allowed values are: variable names

1. if you want to select all variables between a and e, use a:e
2. if you want to exclude a column name y use -y
3. for more options, see: dplyr::select()

Usage of gather():

Move to tidy.R file

https://drive.google.com/file/d/1N0AZZtf7IaZKCRm2Vuj0u8vcLeQ1tlim/view?usp=sharing

spread(): spread two columns into multiple columns [Replacement of CAST(reshape2)]

Features:

1. The function spread() does the reverse of gather().
2. It takes two columns (key and value) and Spreads into multiple columns. It produces a "wide" data format from a "long" one.

Simplified format:

gather(data, key, value, ...)

data: A data frame

key: The (unquoted) name of the column whose values will be used as column headings

value: The (unquoted) names of the column whose values will populate the cells

Usage of spread()

Move to tidy.R file

unite(): Unite multiple columns into one

The function unite() takes multiple columns and paste them together into one.

Simplified format:

```
unite(data, col, ..., sep = "_")
```

data: A data frame

col: The new (unquoted) name of column to add.

sep: Separator to use between values

separate(): separate one column into multiple

The function sperate() is the reverse of unite(). It takes values inside a single character column and separates them into multiple columns.

Simplified format:

```
separate(data, col, into, sep = "[^[:alnum:]]+")
```

data: A data frame

col: Unquoted column names

into: Character vector specifying the names of new variables to be created.

sep: Separator between columns:

Usage of unite() & separate()

Move to tidy.R file

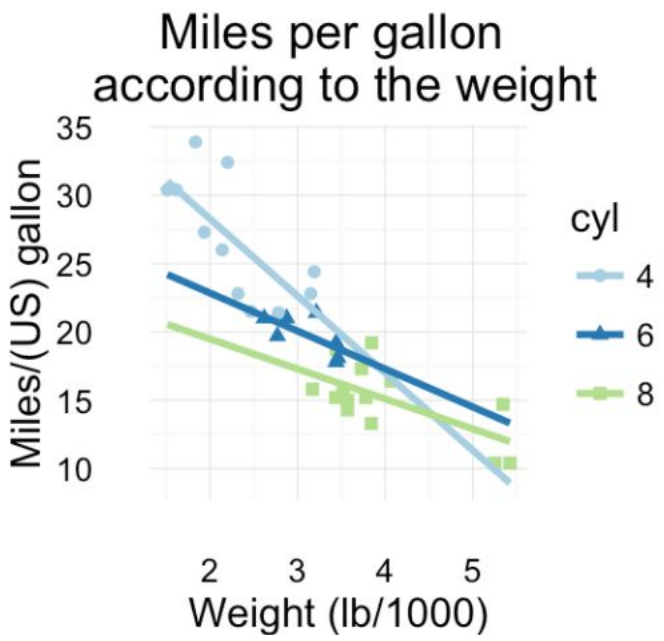
Practice Task

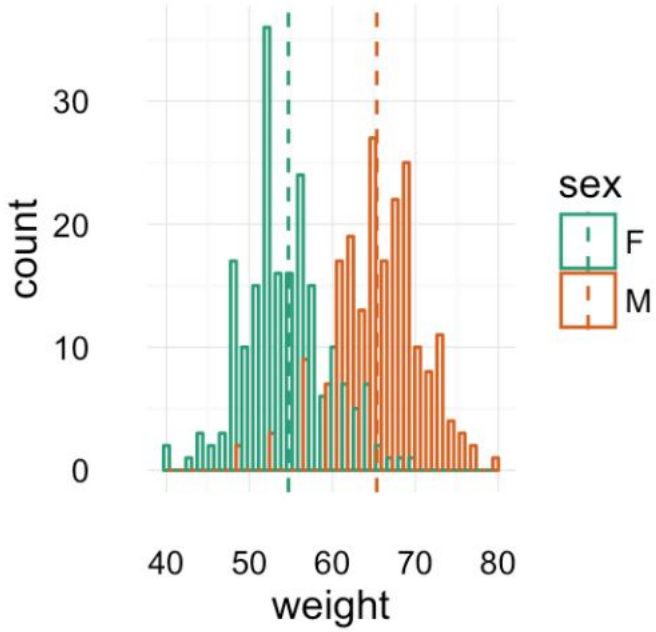
Visualize Your Data

- **ggplot2** is a powerful and a flexible **R package**, used for producing elegant graphics.
- The concept behind ggplot2 divides plot into three different fundamental parts: **Plot = data + Aesthetics + Geometry**.

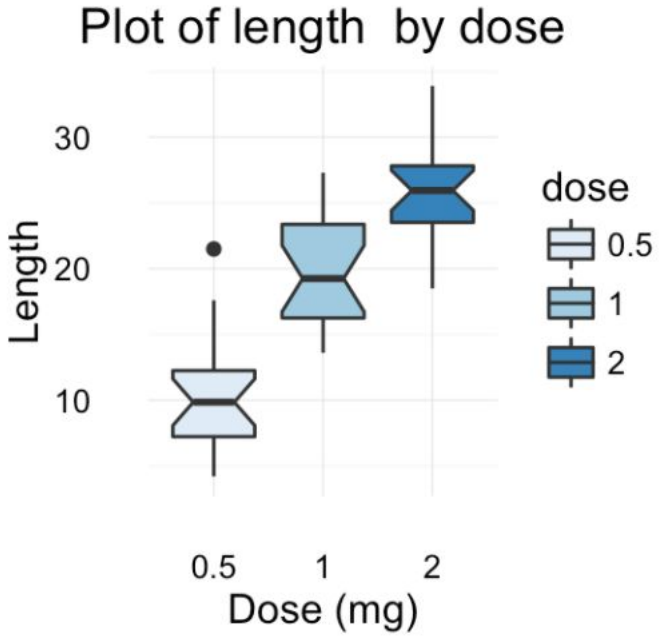
The principal components of every plot can be defined as follow:

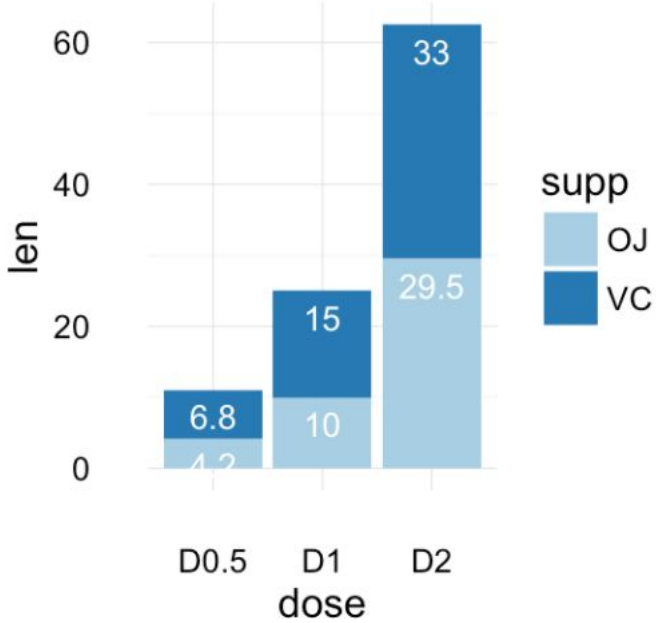
1. **data** is a data frame
2. **Aesthetics** is used to indicate x and y variables. It can also be used to control the **color**, the **size** or the **shape** of points, the height of bars, etc…..
3. **Geometry** defines the type of graphics (**histogram**, **box plot**, **line plot**, **density plot**, **dot plot**, ….)

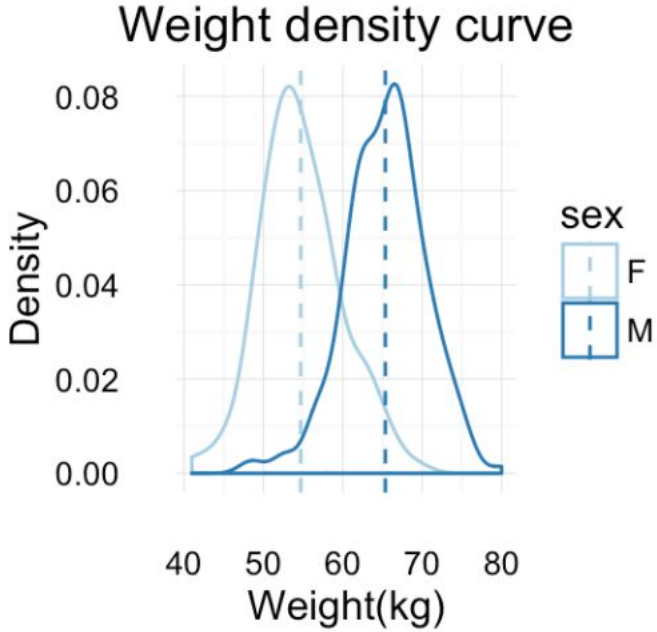

Miles per gallon
according to the weight
Miles/(US) gallon
35
30
25
20
15
10
2
3
4
5
Weight (lb/1000)
cyl
4
6
8


count
30
20
10
0
40
50
60
70
80
weight
sex
F
M


Plot of length by dose
Length
30
20
10
0.5
1
2
Dose (mg)
dose
0.5
1
2


Plot of length by dose
Length
30
20
10
0.5
1
2
Dose (mg)
dose
0.5
1
2


len
60
40
20
0
33
29.5
15
10
6.8
D0.5
D1
D2
dose
supp
OJ
VC


Weight density curve
Density
0.08
0.06
0.04
0.02
0.00
40
50
60
70
80
Weight(kg)
sex
F
M

Scatter-Plots using ggplot2()

- **Basic scatter plots**
- Simple scatter plots are created using the R code. The color, the size and the shape of points can be changed using the function **geom_point()** as follow :

```
geom_point(size, color, shape)
```

Install Package ggplot2() and data-set

Package:

install.packages(‘ggplot2’)

Library(ggplot2)

Data-Set:

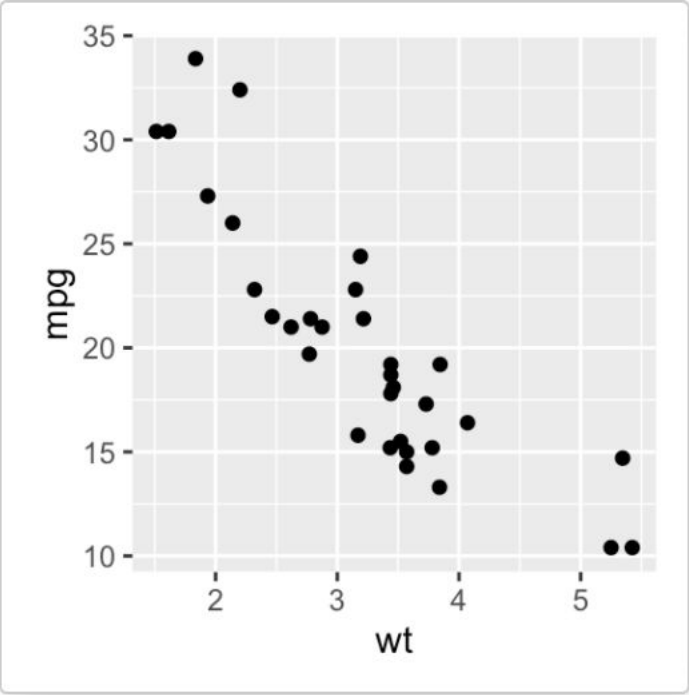
mtcars

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
>
```

- **library**(ggplot2)

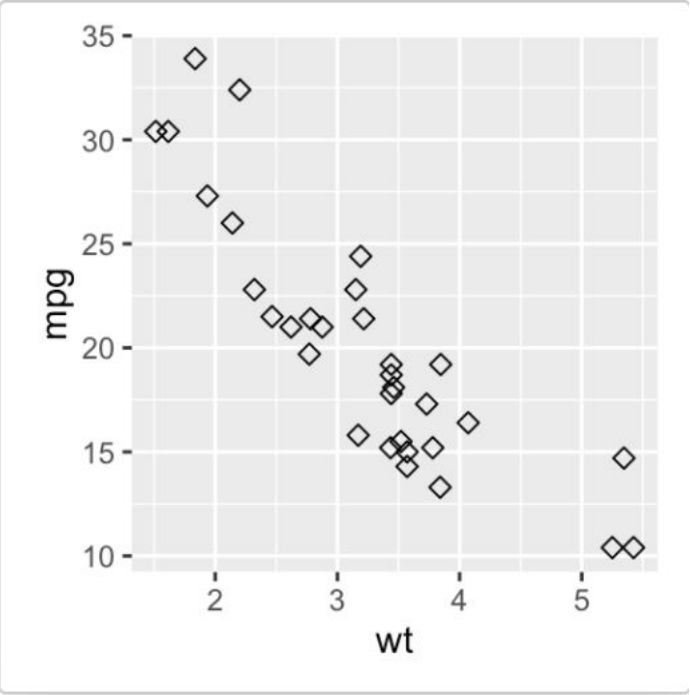
Basic scatter plot:

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



Change the point size, and shape:

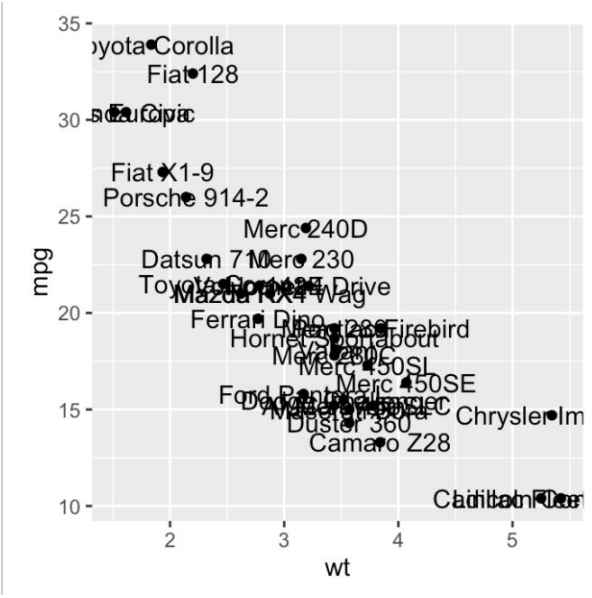
```
ggplot(mtcars, aes(x=wt, y=mpg))
+ geom_point(size=2, shape=23)
```



Label points in the scatter plot

The function **geom_text()** can be used:

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

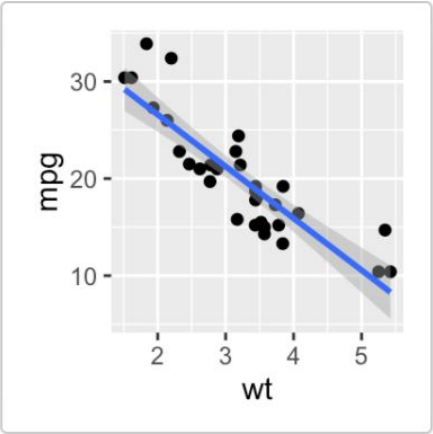


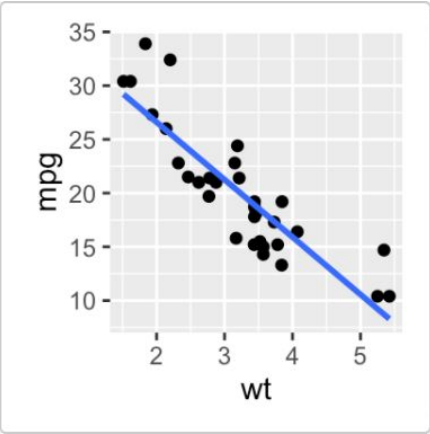
Add regression lines

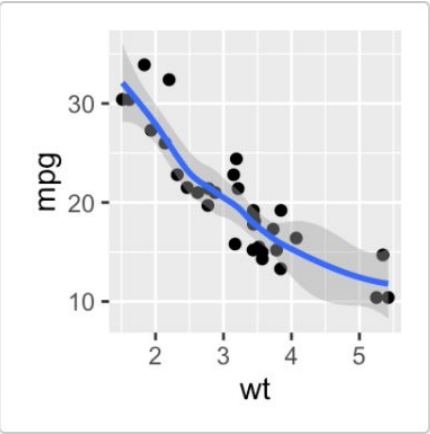
The functions below can be used to add regression lines to a scatter plot :

- **geom_smooth()** and **stat_smooth()**

```
.# Add the regression line:
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()+ geom_smooth(method=lm)
# Remove the confidence interval:
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()+ geom_smooth(method=lm, se=FALSE)
# Loess method:
ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()+ geom_smooth()
```







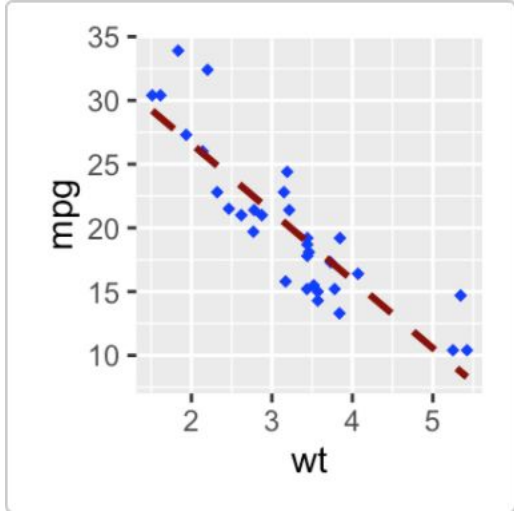
Change the appearance of points and lines

This section describes how to change :

1. the color and the shape of points
2. the line type and color of the regression line
3. the fill color of the confidence interval(se)

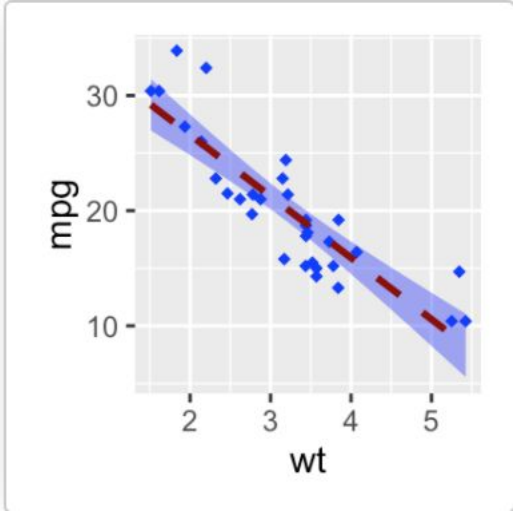
Change colors and shape of point and line

```
ggplot(mtcars, aes(x=wt, y=mpg)) +
geom_point(shape=18, color="blue")+
geom_smooth(method=lm, se=FALSE,
linetype="dashed", color="darkred")
```



Change the confidence interval fill color

```
ggplot(mtcars, aes(x=wt, y=mpg)) +
geom_point(shape=18, color="blue")+
geom_smooth(method=lm, linetype="dashed",
color="darkred", fill="blue")
```



Scatter plots with multiple groups

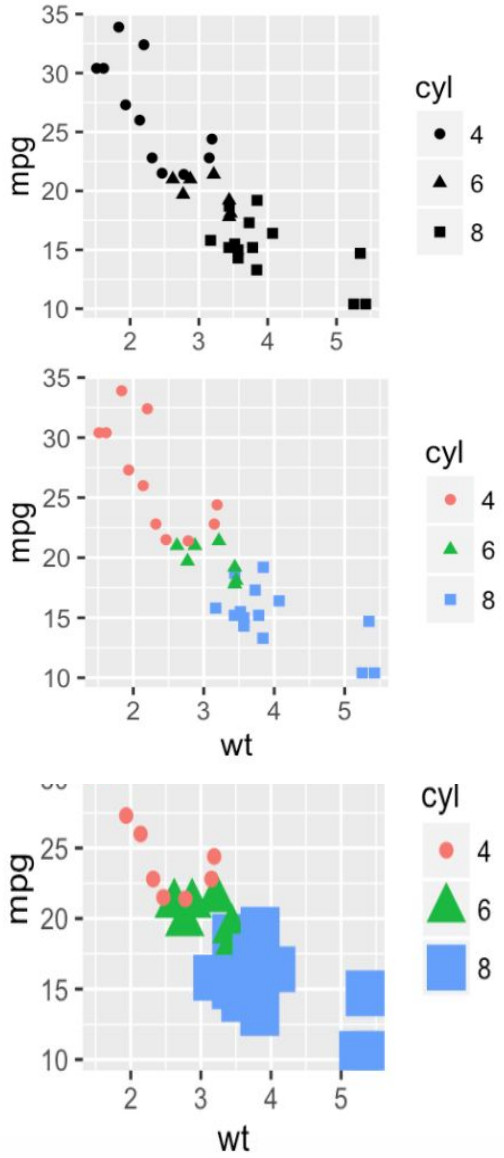
Change the point color/shape/size automatically:

- In the R code below, point shapes, colors and sizes are controlled by the levels of the factor variable *cyl* :

```
# Change point shapes by the levels of cyl
ggplot(mtcars, aes(x=wt, y=mpg, shape=cyl)) + geom_point()

# Change point shapes and colors
ggplot(mtcars, aes(x=wt, y=mpg, shape=cyl, color=cyl)) + geom_point()

# Change point shapes, colors and sizes
ggplot(mtcars, aes(x=wt, y=mpg, shape=cyl, color=cyl, size=cyl)) + geom_point()
```



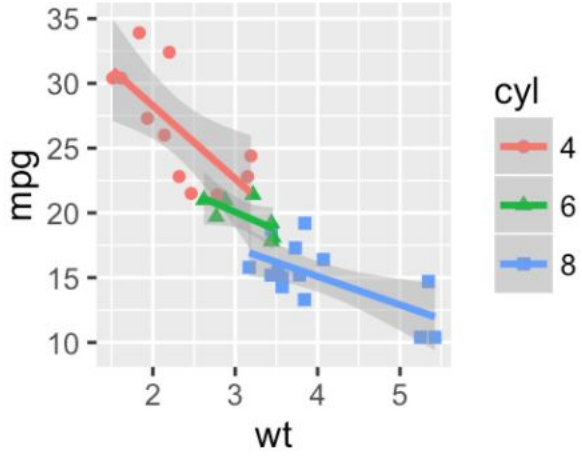
Add regression lines

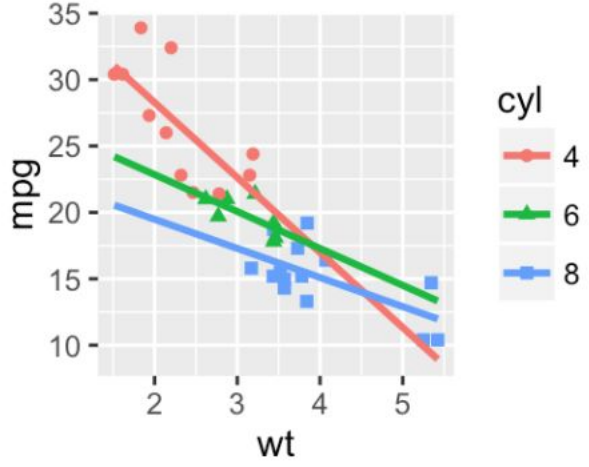
Add regression lines

- ggplot(mtcars, aes(x=wt, y=mpg, color=cyl, shape=cyl)) + geom_point() + geom_smooth(method=lm)

Remove confidence intervals and Extend the regression lines

- ggplot(mtcars, aes(x=wt, y=mpg, color=cyl, shape=cyl)) + geom_point() + geom_smooth(method=lm, se=FALSE, fullrange=TRUE)

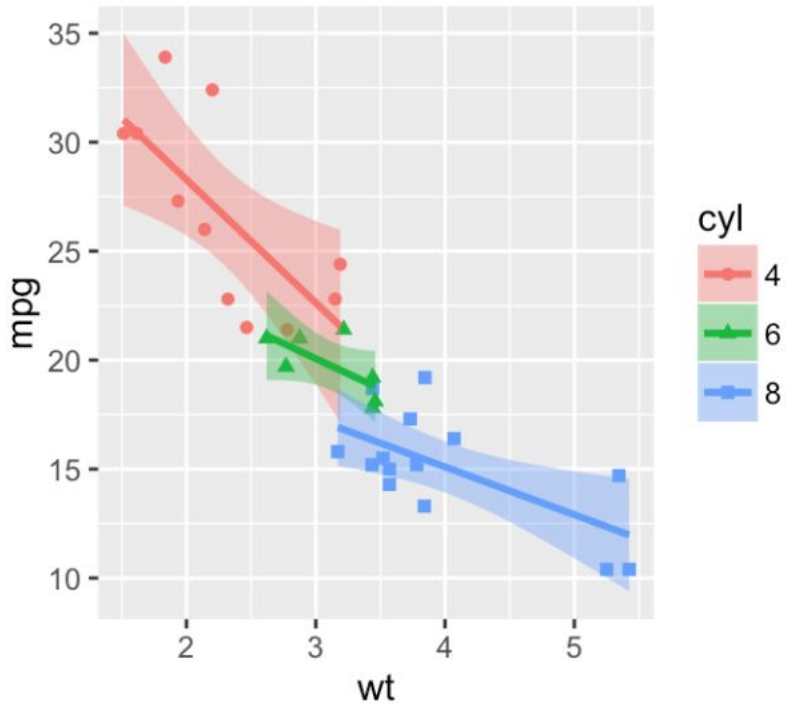




#The fill color of confidence bands can be changed as follow :

- ggplot(mtcars, aes(x=wt, y=mpg, color=cyl, shape=cyl))

+ geom_point() + geom_smooth(method=lm, aes(fill=cyl))



????

Practice Task