

3<sup>rd</sup>)



- R Programming


by Kumaran Ponnambalam, Dedicated to Data Science Education Follow

This course focuses on **R Programming**. It explains the various constructs of the language and provides examples of how to use them.

Please download and install R and R Studio

R installation: <http://cran.r-project.org/bin/windows/base/> in english, into C:\Program Files\R\R-4.0.2

R studio installation: <http://www.rstudio.com/>

The **resource bundle** for this course can be downloaded  ADSR-Resources.zip from <https://www.dropbox.com/s/ayicd007a35j9j0/Resources.zip?dl=0>

**14 Lessons (2h 47m)**

**URL:** <https://www.skillshare.com/classes/Applied-Data-Science-3-R-Programming/135988399>

## 1. About Applied Data Science Series

8:12

### Course goal

- Train students to be full-fledged data science **practitioners** who could execute **end-to-end** data science projects to achieve **business results**
- The course is oriented towards existing software professionals
  - Heavily focused on **programming and solution building**
  - Limited, as-required exposure to math and statistics
  - Overview of ML concepts, with focus on using existing tools to develop solutions

### Achievements

- Understand the **concepts and life cycle of Data Science**
- Develop proficiency to **use R** for all stages of analytics
- Learn **Data Engineering tools and techniques**
- Acquire knowledge of different **machine learning techniques** and know when and how to use them.
- Become a full-fledged Data Science Practitioner who can immediately contribute to real-life Data Science projects

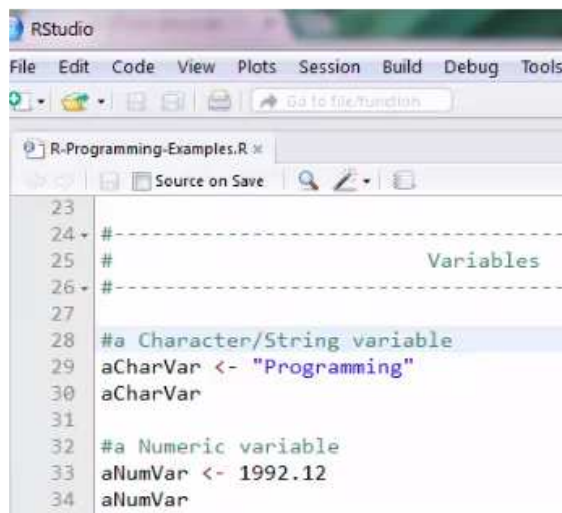
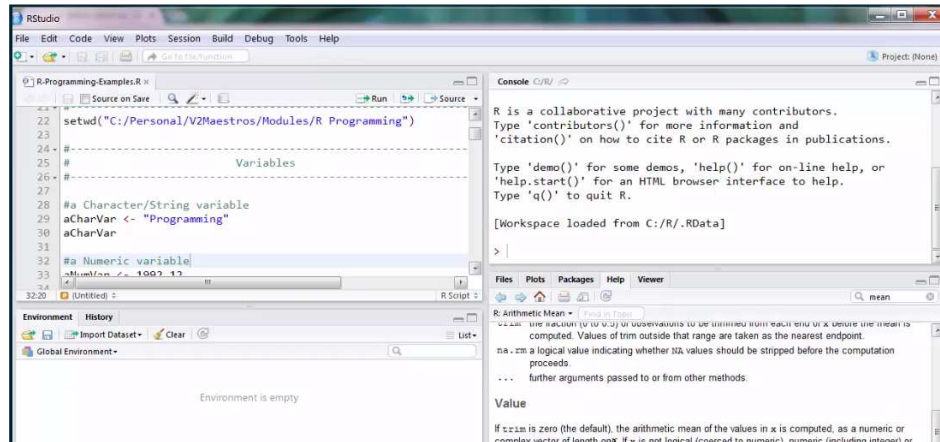
### Course structure

- Concepts of Data Science
- Data Science Life Cycle
- **Statistics for Data Science**
- **R Programming**
  - Examples
- Data Engineering
- Modeling and Predictive Analytics
  - Use cases
- Advanced Topics
- Resource Bundle

## 2. R Studio walkaround

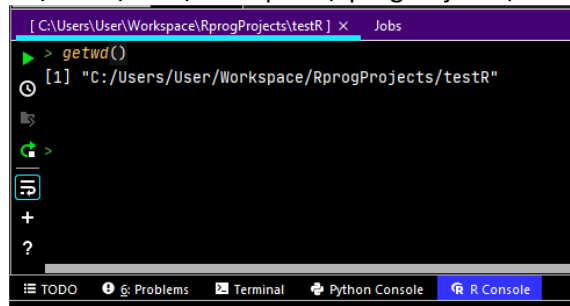
6:40

### RStudio



### Using PyCharm with R Console

C:\Users\User\Workspace\RprogProjects\testR



cmd: getwd()

R Installation : <https://www.w3adda.com/r-tutorial/r-installation> ,  
<https://www.w3schools.in/r/install/>

Tutorials on R : <https://www.w3schools.in/category/r/>

## Introduction

*R* is a language and environment for statistical computing and graphics. *R* provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible...

- Programming experience in another programming language assumed. Hence programming constructs are not explained in detail
- *R* Offers an exhaustive list of functions. Not all covered. It is expected that the student uses other resources to search and master them
  - R Studio help
  - R website
  - The internet

Introduction : <https://www.w3schools.in/r/>

Overview : <https://www.w3schools.in/r/overview/>

## R language basics (committed to Data Science)

### Working directory

- A working directory is always assumed by *R*
- New files are created by default in this directory
- Files to be read are expected by default in this directory
- `getwd()` and `setwd()` are used to manage the it.

### Getting Help

- A number of open source resources available for help with *R* language
- <http://www.r-project.org/> - Manuals
- Within command line - "?"
- *R* forums
  - Nabble
  - Stack Overflow
  - Cross validated
  - R-Help mailing list
- Google search

### Variables

- Creating variables are easy and straightforward
- Assignment operator is "`<-`". "`=`" can also be used
- Variable types
  - Character
  - Numeric
  - Boolean
- *R* is case sensitive. So are variable names
- No explicit variable types declarations. Variable takes the type of the assigned value.

```
var2 <- 3
```

### Strings

- Strings are enclosed in single or double quotes
- Concatenation function is `c()`

```
con_string <- c( "Programming", "Languages" )
```
- Special characters can be included using the backslash "`\`" inside the string
- A number of string manipulation functions available

### Date and Time

- POSIX Date supported as standard
- Convert character to date with `as.Date()`  

```
as.Date('2015-01-01')
```
- A number of date functions available for date formatting, conversion and arithmetic

### Missing Data (NA)

- Missing data is represented as NA (not available) in R
- Most R functions are sensitive to missing values. A parameter "na.rm" can be set to TRUE to tell the function to ignore missing values.
- `is.na()` can be used to test if a variable has missing value

## 4. R02 Vectors and Lists

8:51

### Vectors

- Vectors are a key construct in R.
  - Equivalent of one-dimensional arrays
- Most data handling happens through vectors
- Sequence/List of data elements of the SAME data type
  - Equivalent to arrays
  - Can hold any data type
- Vectors created using c()

```
my_vector <- c( 3, 5, 10)
char_vector <- c( "apples", "oranges")
bool_vector <- c(FALSE, TRUE, FALSE)
```

### Vector indexing

- Members of the vector accessed through index numbers
- Starts with 1
- A sub-set can be accessed through "." like 3:6
- Negative index numbers used to remove members from the output
- Out-of-range indexes results in NA

### Vector operations

- Arithmetic operations can be applied on vectors similar to regular variables.
- When two vectors are added, the equivalent members of the vectors are added and the result will also be a vector.

```
vec1 <- c(1,2,4)
vec2 <- c(4,5,6)
vec1 + vec2
5, 7, 10
```

- When a condition is applied on a vector, it returns another vector of TRUE or FALSE based on each member of the vector

```
my_vector <- c( 3, 5, 10)
my_vector > 6
FALSE, FALSE, TRUE
```



### Named vectors members

- Equivalent of associative arrays
- Helps to track vector members
- Access vector members by name

```
vegetables <- c(5,7,3)
names(vegetables) <- c("carrot", "beans", "broccoli")
vegetables["beans"]
```

### Lists

- Lists are similar to vectors.
- A List can hold elements of **different data types** (A vector can only hold data of one type)
- Equivalent to “structures” in other languages.

```
employee <- list(1001, "Joe Smith", FALSE)
```

## 5. R03 Data Frames and Matrices

14:41

### Data Frames

- The most used data structure in R
- **Represents a table** – with rows and columns
  - One column holds data of the same type ( a vector)
  - Different columns can have different data types. ( a list)
- Operations can be performed on entire rows and columns

	empId	empName	isManager	salary	dept_id
1	1	John	FALSE	1242.11	1
2	2	Mike	TRUE	3490.20	1

### Creating Data Frames

- Data frames can be created in many ways
  - Combining individual vectors
  - Combining other data frames
  - Reading from files
  - Reading from databases
- Data Frames can be analyzed using the following **commands**
  - nrow and ncol
  - summary
  - head
  - str

### Data Frames indexing

- Every row in a data frame has a row id.
- Rows and columns can be accessed through indexes
- Indexes can be **numeric or associative**

```
example_df[ 1, 2 ]
example_df$example_col
example_df[ 1:3, 4:5 ]
```

- Individual rows of a data frame are Lists.
- Individual columns of a data frame are vectors.

*a column = vector , , frame\$col => return a full column of frame (i.e. a vector), a row = list*

## Data Manipulations

- Data Frames can be combined with `rbind()` and `cbind()`
- Columns can be added by using an associative name and contents  

```
example_df$example_col <- c(1,2,3)
```
- Most operations on vectors can be done on individual rows or columns of a data frame
- Data Frames can be merged using a common column with the `merge()` function
- A number of built-in data frame examples are available in R

## Matrices

- Matrices are `two dimensional` arrays similar to data frames
- `Only numeric values` are allowed within a matrix.
- Matrices can be converted to `data frames` and vice-versa
- A number of operations on data frames apply to matrices also.

```
matrix(
  c(2, 4, 3, 1, 5, 7),
  nrow=3,
  ncol=2) → [1,] [2,] [3,]
```

```
[1,] 2 1
[2,] 4 5
[3,] 3 7
```

## Factors

- Factor is a data type used to store `categorical variables`
- Any vector (including columns in data frames) can be converted to and from factors
- Factors permit special grouping operations on data like “tables”
- A number of statistical and machine learning functions require factor data types

## 6. R04 Data and Input Output Ope... 10:30

### Data Transformations

#### Sorting

- Any vector can be sorted using the `sort()` function
- A Data Frame can be sorted using the `order()` function
- Data Frames can be sorted using multiple columns

#### Merging

- Two Data Frames can be merged using a common column – `merge()`
- If column names are same, no specification required.
- Used for foreign key operations (joins)

**Ex.**

empId	empName	isManager	salary	dept_id
1	John	FALSE	1242.11	1
2	Mike	TRUE	3490.20	1
3	Randy	FALSE	2201.87	2

*merged with*

dept_id	dept_name
1	Sales
2	Operations

dept_id	empId	empName	isManager	salary	dept_name
1	1	John	FALSE	1242.11	Sales
1	2	Mike	TRUE	3490.20	Sales
2	3	Randy	FALSE	2201.87	Operations

### Binning

- Continuous data can be converted to categorical data by binning
- Data is converted into pre-defined ranges or bins
- cut() function is used for binning

```
emp_df$sal_range <- cut( emp_df$salary,  
                        c(1,2000,5000 ) )
```

Factor

empId	empName	isManager	salary	dept_id	sal_range
1	John	FALSE	1242.11	1	(1,2e+03]
2	Mike	TRUE	3490.20	1	(2e+03,5e+03]
3	Randy	FALSE	2201.87	2	(2e+03,5e+03]

### Input/Output operations

#### Console operations

- scan() function can be used to read input from console (and from files too)
- print() function – used to print data to console
- cat() – also writes to screen
- dir() / list.files() prints directory listing

#### File operations

- read.table() function reads any file into a data frame
  - Can control file format like newline, separator, header etc.
- read.csv() function reads a csv file into a data frame
  - Pre-defined format
- write.table() and write.csv() are equivalent write operations

#### Database operations

- Similar to Files, Data can be read from and written to RDBMS databases
- Libraries exist to access any kind of DBs
- Data can also be accessed from Hadoop using Rhadoop

#### Web downloads

- Files can be downloaded directly from the web using download.file()
- Raw web pages can be scraped using getURL()

## 7. R05 Programming and Packages

12:41

### R Programming

#### Operators

- Arithmetic
  - +, -, \*, /
  - Modulus - %%
- Logical
  - <, <=, >, >=, ==, !=
  - |, &
- Decimal places adjust based on operands and result

## Control structures

- If-else

```
if (cond) expr  
if (cond) expr1 else expr2
```

- For

```
for (var in seq) expr
```

- While

```
while (cond) expr
```

- Switch

- Ifelse

```
ifelse(test, yes, no)
```

## User Defined Function

- Functions can be built and saved as function definitions in memory
- Functions take input parameters and write output
- Functions can access global level variables

```
computeSum <- function(x,y) {  
  x+y  
}  
computeSum(3,5)
```

## Packages (libraries)

- Packages are the life line of R
- Packages provide a set of related functionality
- They are implemented in native R and sometimes in C and FORTRAN
- Extensive set of packages offering a wide variety of options
- The online CRAN (Comprehensive R archive network) repository hosts downloadable packages. <http://cran.r-project.org/web/packages/>
- Anyone can build a package and host it for others to use.

<https://cran.r-project.org/web/packages/>

## Installing Packages

- Packages can be installed inside R command shell using `install.packages()` command
- Dependent packages are also automatically installed.
- Once installed, packages are available for use in that R installation
- Packages need to be loaded before use with the `library()` command.
- R Studio maintains packages and can also upgrade them.

## Apply Functions

- A number of R programs require looping through a data frame and performing an operation on each record.
- The “apply” class of functions provide a short-cut through which that entire operation can be performed in one function call.
- The function to apply for each row can be user defined function too.
- Variants of the “apply” functions include `apply`, `by`, `eapply`, `lapply`, `mapply`, `rapply`, `tapply`

```
mat <- matrix(c(1:20), nrow = 10, ncol = 2)  
apply(mat, 1, mean)  
apply(mat, 2, mean)  
1=>row, 2=>col
```



## 8. R07 Statistics in R

3:01

### Statistics in R

- R was originally **built for statisticians**, so there is no dearth of statistical functions and packages
  - Descriptive Statistics
  - Correlations
  - T-tests
  - Regression
  - ANOVA and its variants
  - Power Analysis

### Descriptive Statistics

- `mean()`, `sd()`, `var()`, `min()`, `max()`, `median()`, `range()`, `quantile()`
- `summary()` on a numeric vector provides **quartiles**
- **psych** package provides additional statistical functions

### Correlation

- Correlation between columns in a data frame can be found using the **`cor()` function**.
- Different types of correlation methods supported – pearson, spearman, kendall
- **psych** package has additional correlation functions.

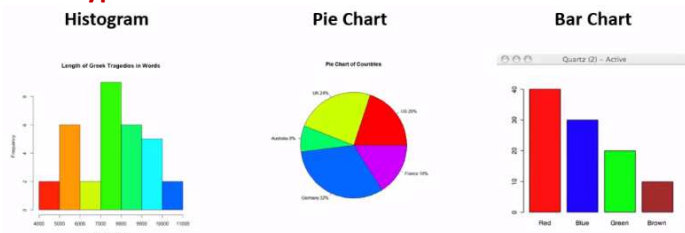
## 9. R08 Graphics in R

6:51

### Graphics in R

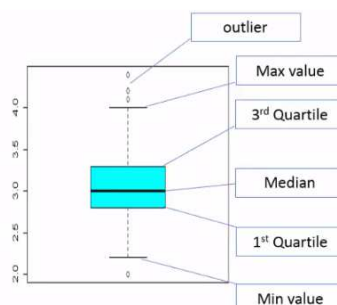
- A picture is worth a thousand words.
- With large amounts of data, graphical representations are the best way to understand trends
- Graphics play a huge role in all stages of data science
  - Identifying outliers during cleansing
  - Spotting good predictors during exploratory data analysis
  - Explaining results to project owners

### Chart types



### Box and Whisker Plot

- Shows the 4 Quartiles for any set of values
- Values that are beyond 1.5 times the IQR from the 1<sup>st</sup>/3<sup>rd</sup> quartiles are shown as outliers
- Shows how skewed the distribution of values are



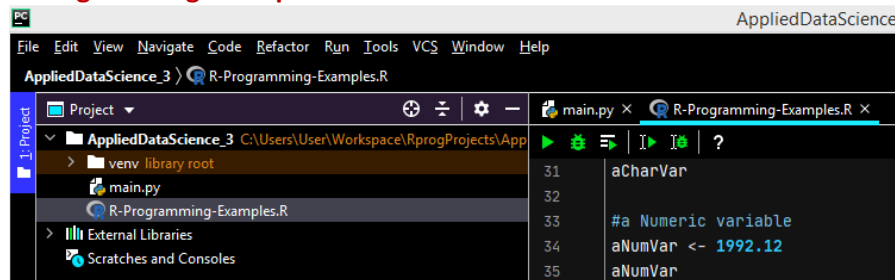
## Plotting systems

- Three systems
  - Base Plotting System
  - Lattice System
  - Grammar of Graphics (GG)
- Extensive set of functionality for type and control
- Use help files to explore the charting options available

## 10. R Examples 01

16:18

### R-Programming-Examples.R



browsing the program in details... run etc :

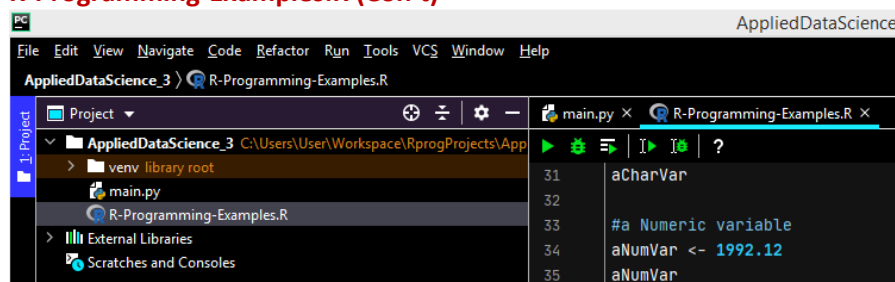
Variables  
Arithmetic  
Strings  
Date and Time  
Vectors  
Lists

<- , setwd(), getwd(), class(), typeof(), ? in order to get help, boolean (TRUE, FALSE), as.numeric(), as.character(), ls(), c(), paste(), paste0(), cat(), Sys.time(), length(), sum(), list(), list\$elt

## 11. R Examples 02

15:05

### R-Programming-Examples.R (Con't)



browsing the program in details... run etc

Data Frames

stringsAsFactors = TRUE vs. FALSE (factor vs. chr)

<- c(), class(), nrow(), ncol(), str(), summary(), names(), head(), tail(), sum(), <- data.frame(...)

```
170 emp_df[1,3] # access row 1 col 3
171 emp_df[1:2,1:3] # access range
172 emp_df$salary # access full salary column
173 emp_df$empName[2] # access 2nd element in a column
```

```

177 emp_df$salary > 2000
178 emp_df [ emp_df$salary < 2500, 1] # filter on salary
179
180 emp_df$dept_id <- c(1,1,2) #add a new column dept_id
181 emp_df
182 emp_df[ emp_df$dept_id == 2 , (1:5)] # filter on dept_id

```

filtering, c(), ...

builtin data frame

iris (<https://www.rdocumentation.org/packages/datasets/versions/3.6.2/topics/iris>)

str(iris)

```

> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...

```

Matrices

```
<- matrix(c(2,4,6,3,5,8), ncol=3,nrow=2)
```

t() to transpose a matrix

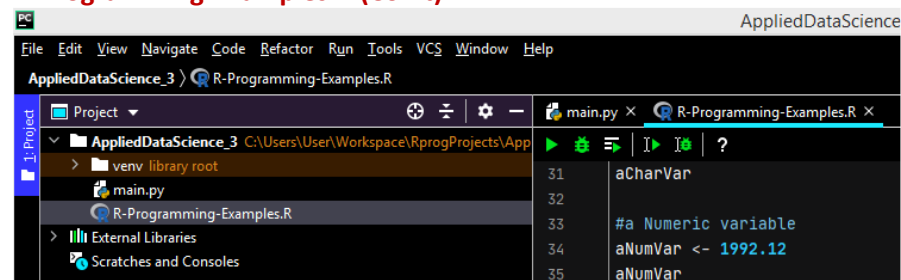
Factors

as.factor(), table(), rbind()

## 12. R Examples 03

17:17

### R-Programming-Examples.R (Con't)



browsing the program in details... run etc

Sorting

sort(vector), data\_frame[order(column)], ...

Merging

merge() join via a column

Binning

cut() used to create bin, aggregate() i.e. group by operation

Input/Output Operations

scan() to read from input, file ...

print(), dir(), list.files(), read.csv(), write.csv() ...

```
employee.csv
1  "", "empId", "empName", "isManager", "salary"
2  "1", 1, "John", FALSE, 1242.11
3  "2", 2, "Mike", TRUE, 3490.2
4  "3", 3, "Randy", FALSE, 2201.87
```

=>

```
emp_df <- read.csv("employee.csv")
emp_df$sal_range <- cut( emp_df$salary, c(1,2000.0,5000.0 ))
write.csv(emp_df,"employee_added.csv")
```

=>

Disque local (C:) > Utilisateurs > User > Workspace > RprogProjects > AppliedDataScience\_3

Nom	Modifié le	Type	Taille
.idea	08-10-20 17:51	Dossier de fichiers	
.RDataFiles	08-10-20 17:48	Dossier de fichiers	
venv	07-10-20 15:34	Dossier de fichiers	
employee.csv	28-01-15 02:16	Fichier CSV	1 Ko
employee_added.csv	08-10-20 17:50	Fichier CSV	1 Ko
R-Programming-Examples.R	08-10-20 17:45	Fichier R	13 Ko

=>

```
employee_added.csv
1  "", "x", "empId", "empName", "isManager", "salary", "sal_range"
2  "1", 1, 1, "John", FALSE, 1242.11, "(1,2e+03]"
3  "2", 2, 2, "Mike", TRUE, 3490.2, "(2e+03,5e+03]"
4  "3", 3, 3, "Randy", FALSE, 2201.87, "(2e+03,5e+03]"
```

## Control Structures

*for (i in 1:nrow(iris)) { ... }, if () else if () else ()*

## Functions

```
290- computeSum <- function(x,y) {
291-   print( paste("Received ", x, y))
292-   x+y
293- }
294- computeSum(4,6)
```

## Packages

The screenshot shows the CRAN (Comprehensive R Archive Network) website. It features the R logo, navigation links like 'CRAN Mirrors', 'What's new?', and 'Task Views'. The main content area is titled 'Contributed Packages' and states that there are 6316 available packages. It provides links to 'Table of available packages, sorted by date of publication' and 'Table of available packages, sorted by name'. At the bottom, there is a section for 'Installation of Packages'.

```
install.packages(pkgs = "RCurl")
library(RCurl)
```

*library(...) used to use a library that is installed*

**package 'RCurl' successfully unpacked and MD5 sums checked**

## Apply Functions

```
mat <- matrix(c(1:20), nrow = 10, ncol = 2)
apply(mat, 1, mean)
apply(mat, 2, mean) =>
[8,] 8 10
[9,] 9 19
[10,] 10 20
> apply(mat, 1, mean)
[1] 6 7 8 9 10 11 12 13 14 15
> apply(mat, 2, mean)
[1] 5.5 15.5
```



## 13. R Examples 04

17:29

Statistics in R

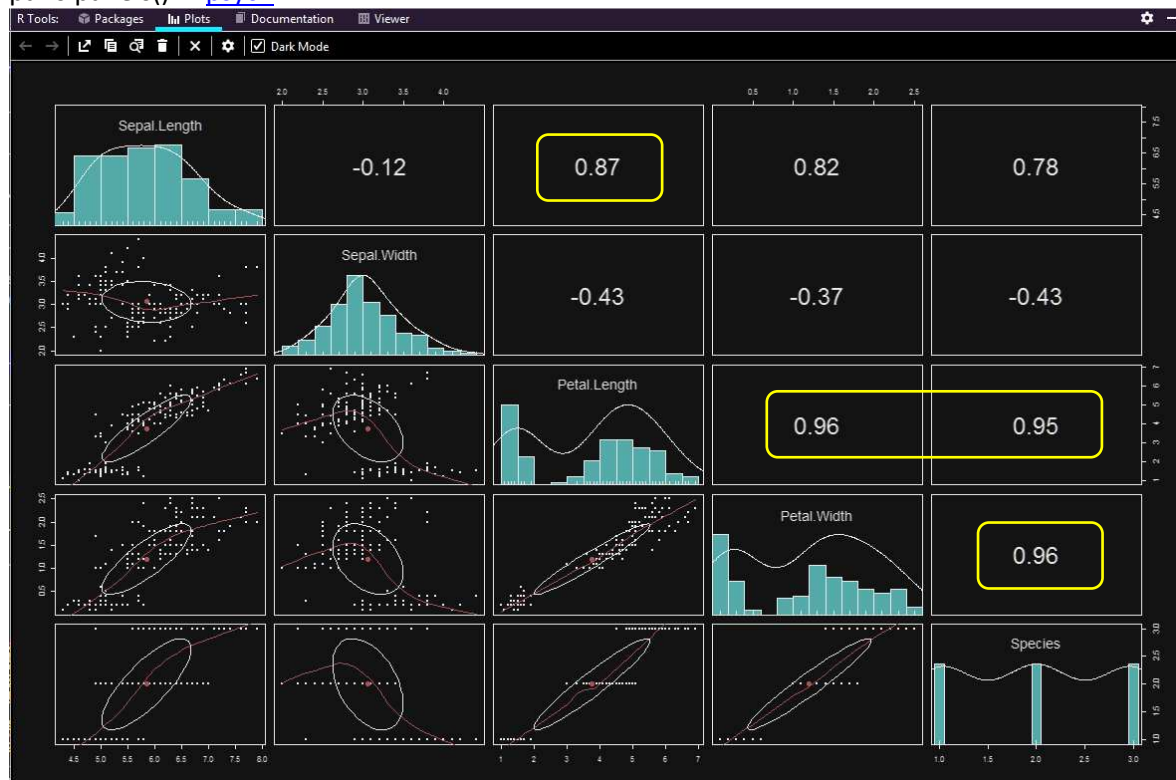
mean(), range(), aggregate(), cor()

	Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	setosa	5.006	3.428	1.462	0.246
2	versicolor	5.936	2.770	4.260	1.326
3	virginica	6.588	2.974	5.552	2.026

install.packages("psych"), library(psych)

<input checked="" type="checkbox"/>	psych	Procedures for Psychological, Psycho...	2.0.9	ⓧ
<input type="checkbox"/>	RCurl	General Network (HTTP/FTP/...) Client ...	1.98-1.2	ⓧ

pairs.panels() in [psych](#)



Observations : les 3 espèces sont bien différenciées avec la longueur et largeur des pétales (et la longueur du sépal qui est corrélé à 87% avec la longueur du pétal)

**Linear model** : lm()

<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm>

<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/formula>

**Residuals:**

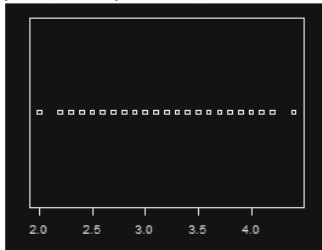
Min	1Q	Median	3Q	Max
-0.59215	-0.15368	0.01268	0.11089	0.55077

**Coefficients:**

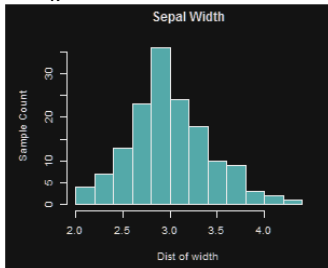
Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	1.18650	0.20484	5.792	4.15e-08 ***
Sepal.Length	-0.11191	0.05765	-1.941	0.0542 .

## Base Plotting System

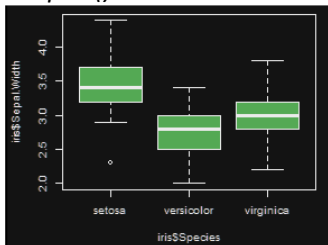
`par()`, `stripchart()`



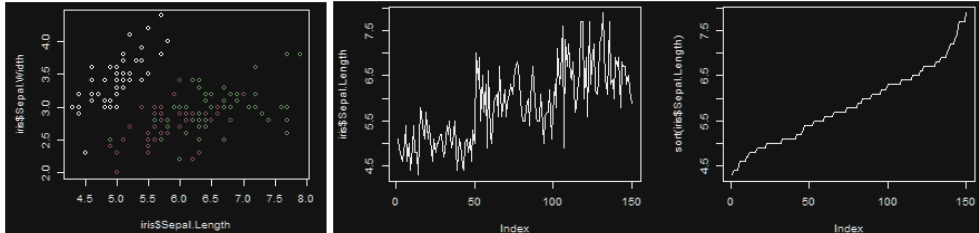
`hist()`



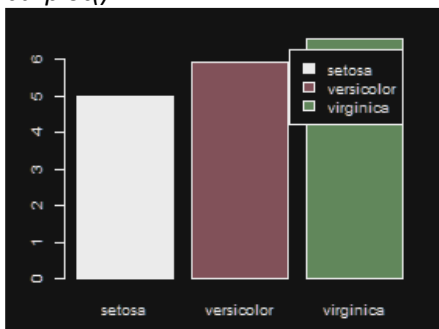
`boxplot()`



`plot()`



`barplot()`



## 14. R Examples 05

17:22

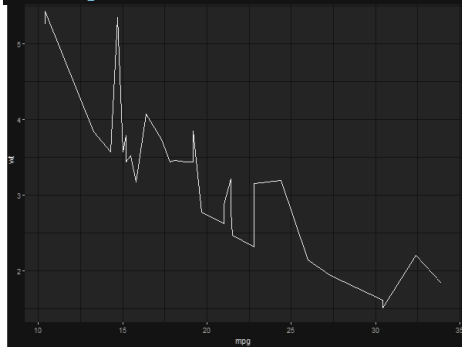
```
ggplot
```

```
data(mtcars), summary(), head()
```

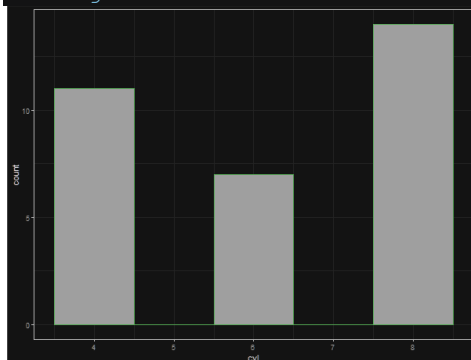
<https://rpubs.com/BillB/217355#:~:text=Source%20Data,for%20then%2Dcurrent%20car%20models.>

```
library(ggplot2), ggplot() + ...
```

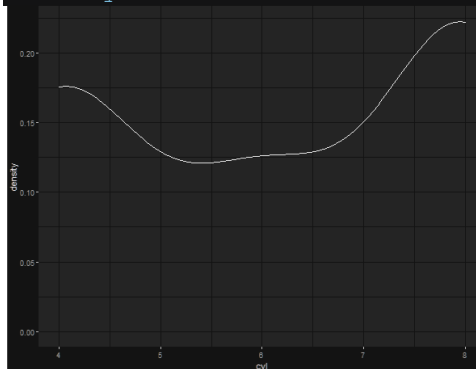
```
line plot
```



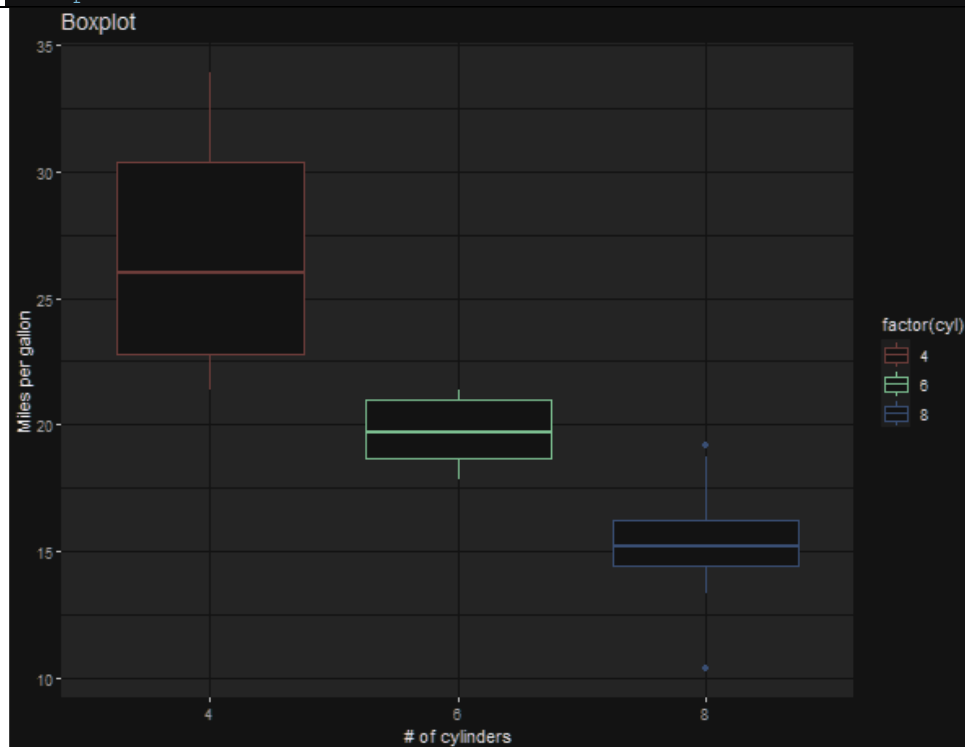
```
histogram
```



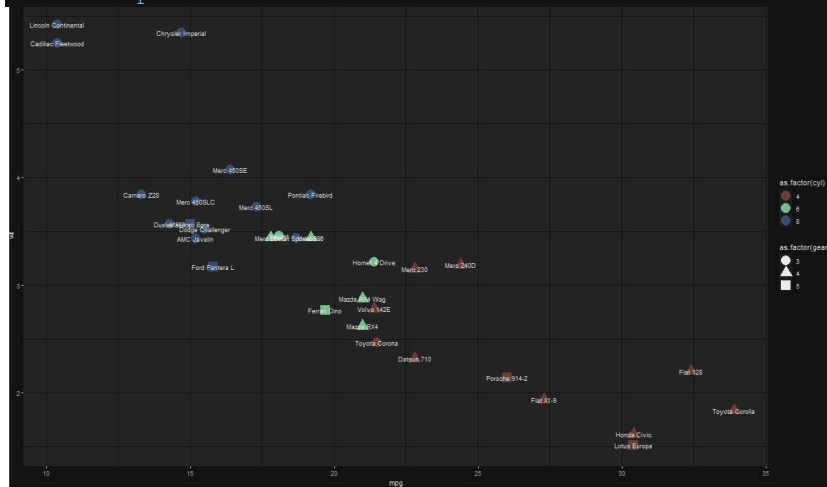
```
density
```



```
box plot
```

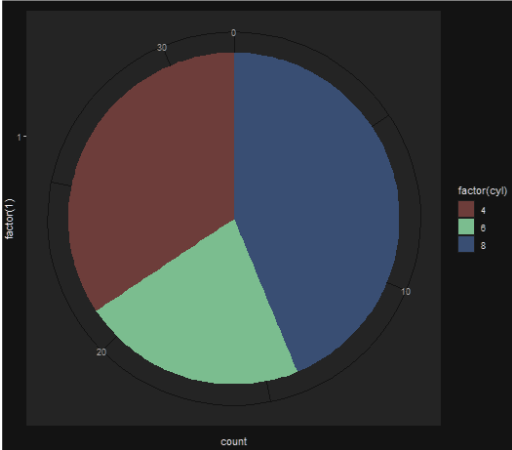


```
scatter plot
```

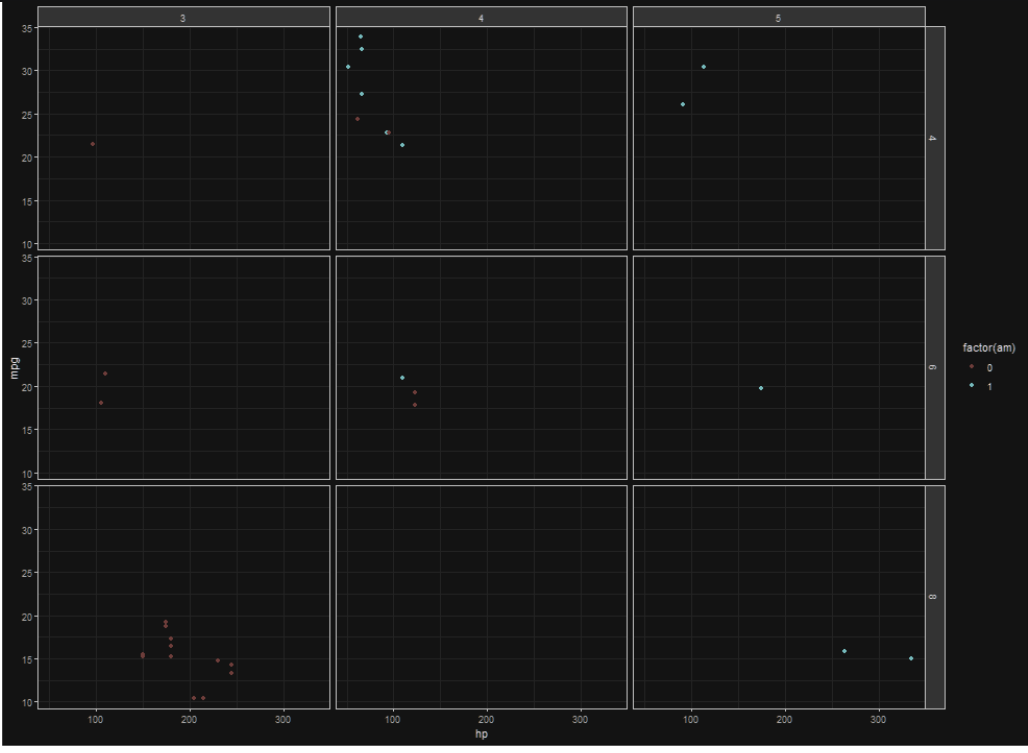


```
pie chart
```

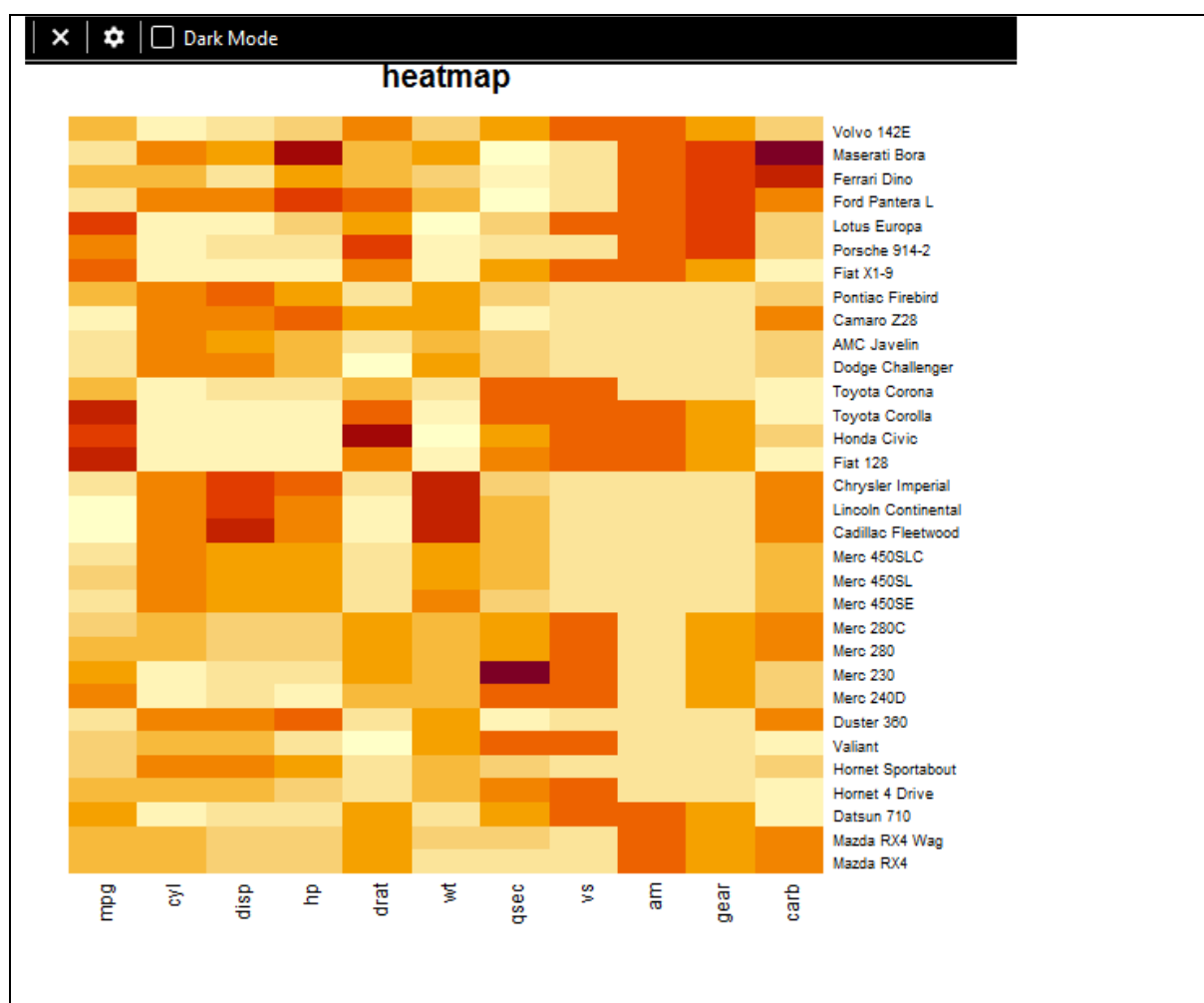




Faceting



Heat Maps



## Time Series

Disque local (C:) > Utilisateurs > User > Workspace > RprogProjects > AppliedDataScience\_3

Nom	Modifié le	Type	Taille
.idea	12-10-20 12:43	Dossier de fichiers	
.RDataFiles	12-10-20 10:55	Dossier de fichiers	
venv	07-10-20 15:34	Dossier de fichiers	
employee.csv	28-01-15 02:16	Fichier CSV	1 Ko
employee_added.csv	08-10-20 17:50	Fichier CSV	1 Ko
R-Programming-Examples.R	12-10-20 12:40	Fichier R	15 Ko
timeseries.csv	20-11-14 03:14	Fichier CSV	8 Ko

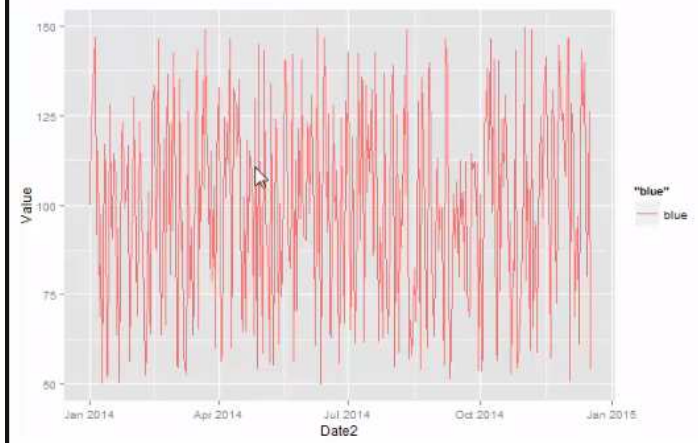
=>

[ C:\Users\User\Workspace\RprogProjects\AppliedDataScience\_3 ] ×

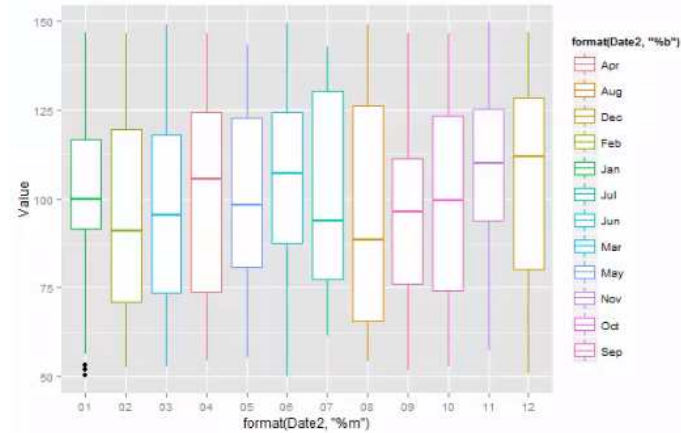
	Date	Value
1	1/1/2014	100.00000
2	1/2/2014	124.71940
3	1/3/2014	125.44312
4	1/4/2014	140.54405
5	1/5/2014	146.70681
6	1/6/2014	91.96421

aes() <https://ggplot2.tidyverse.org/reference/aes.html>

```
geom_line()
```



```
geom_boxplot()
```



Plotting maps

```
library(ggmap), qmap()
```

!!! <https://cran.r-project.org/web/packages/ggmap/readme/README.html> !!!

<https://lucidmanager.org/data-science/geocoding-with-ggmap/>

<https://cran.r-project.org/web/packages/ggmap/index.html>



```
library(tmaptools)
```

[https://wiki.openstreetmap.org/wiki/OSM\\_Scientific\\_Tools](https://wiki.openstreetmap.org/wiki/OSM_Scientific_Tools)

<https://cran.r-project.org/web/packages/OpenStreetMap/index.html>

<https://stackoverflow.com/questions/52704695/is-ggmap-broken-basic-qmap-produces-arguments-imply-differing-number-of-rows/52710855#52710855>

<https://stackoverflow.com/questions/53533662/r-free-maps-similar-to-ggmap>