Stochastic simulation and application in finance

Overview

Khóa học sẽ bao gồm các phần sau:

1. Lập trình và xử lý dữ liệu cơ bản trong 
2. Ứng dụng Monte Carlo simulation để giải các bài toán liên quan đến xác suất.
3. Ứng dụng mô phỏng quá trình giá của cổ phiếu và các tài sản tài chính khác (financial assets).
4. Ứng dụng Monte Carlo simulation để tính giá các tài sản phái sinh (derivative assets).

Overview

Sau khóa học này, sinh viên sẽ có khả năng

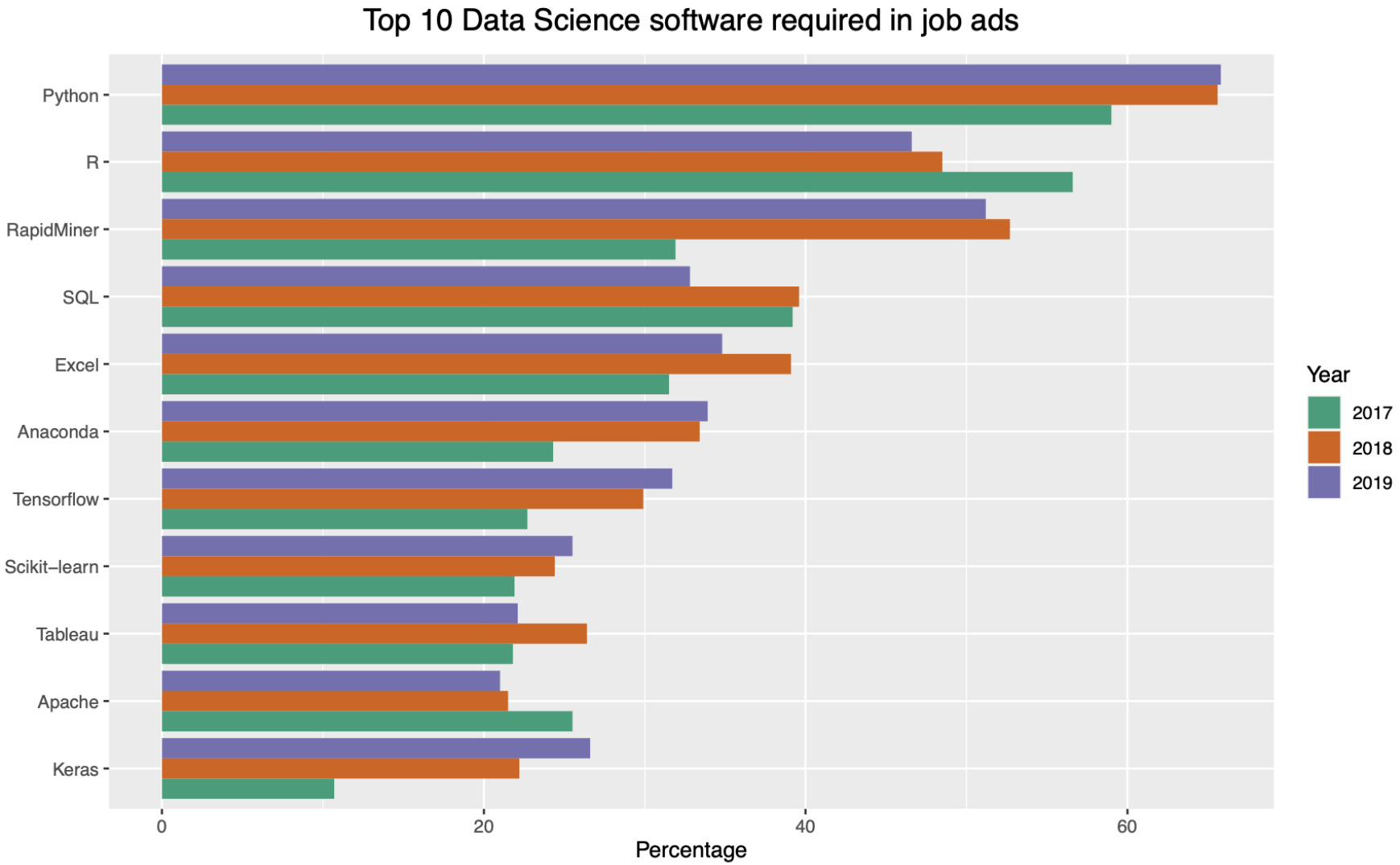
1. Sử dụng  như một ngôn ngữ lập trình.
2. Có kiến thức về Monte Carlo simulation để giải các bài toán liên quan đến xác suất.
3. Ứng dụng mô phỏng ngẫu nhiên để mô tả quá trình giá của các tài sản tài chính:
   - Tính toán an toàn vốn (capital adequacy) tại các doanh nghiệp bảo hiểm
   - Quản lý rủi ro lãi suất (interest rate risk), rủi ro tỷ giá (exchange rate risk)
4. Ứng dụng Monte Carlo simulation để định giá các tài sản phái sinh:
   - Call - Put option on stock
   - Exotic option: American, Asian option ....

Getting started with R

In this section, we will discuss the following topics

1. Why we should use R as a tool in data science
2. R as an environment and R as a language
3. How to download and install R to your computer
4. Introduction to RStudio - an Integrated Development Environment (IDE) for R
5. Important characteristics of R
6. How to run your first R program

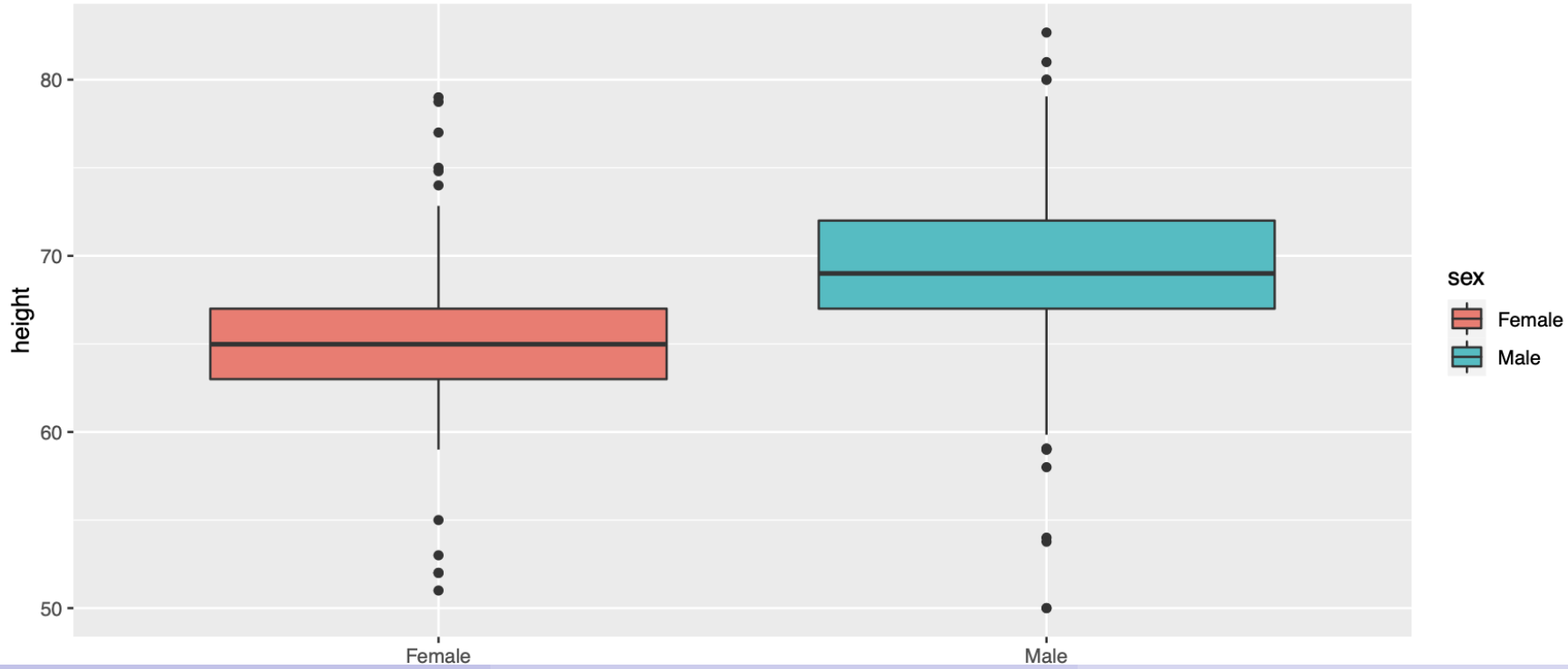
Why use R?


Top 10 Data Science software required in job ads
Python
R
RapidMiner
SQL
Excel
Anaconda
Tensorflow
Scikit–learn
Tableau
Apache
Keras
0
20
40
60
Percentage
Year
2017
2018
2019

What is R?

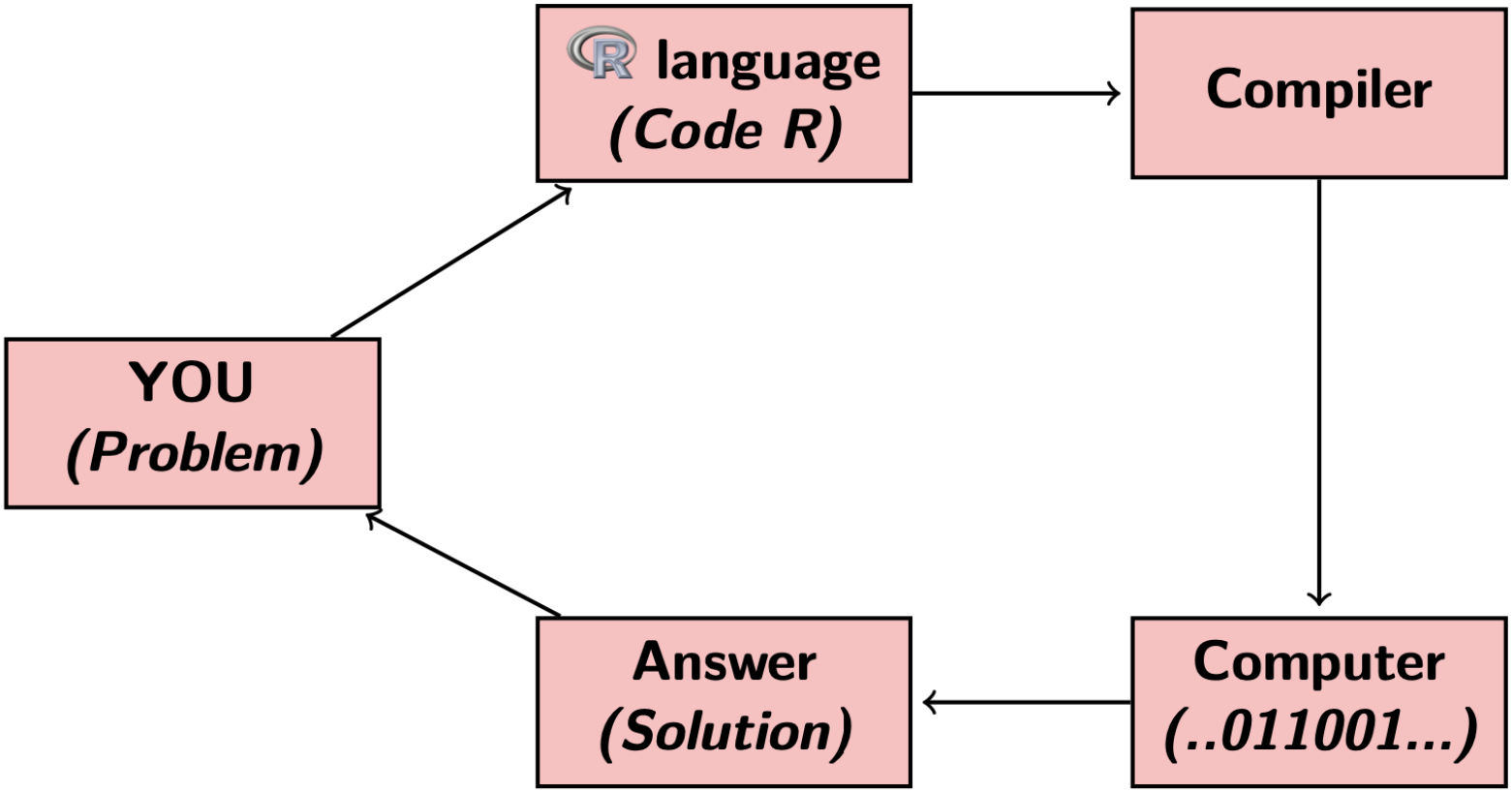
R is an **environment** and also a **language** for performing statistical analysis as well as data mining/data analysis.

- As an environment: you can use R to run, develop, enable your or other's applications.



What is R?

- As a computer languague, you can use R to communication with a computer as Pascal, C, C++,...

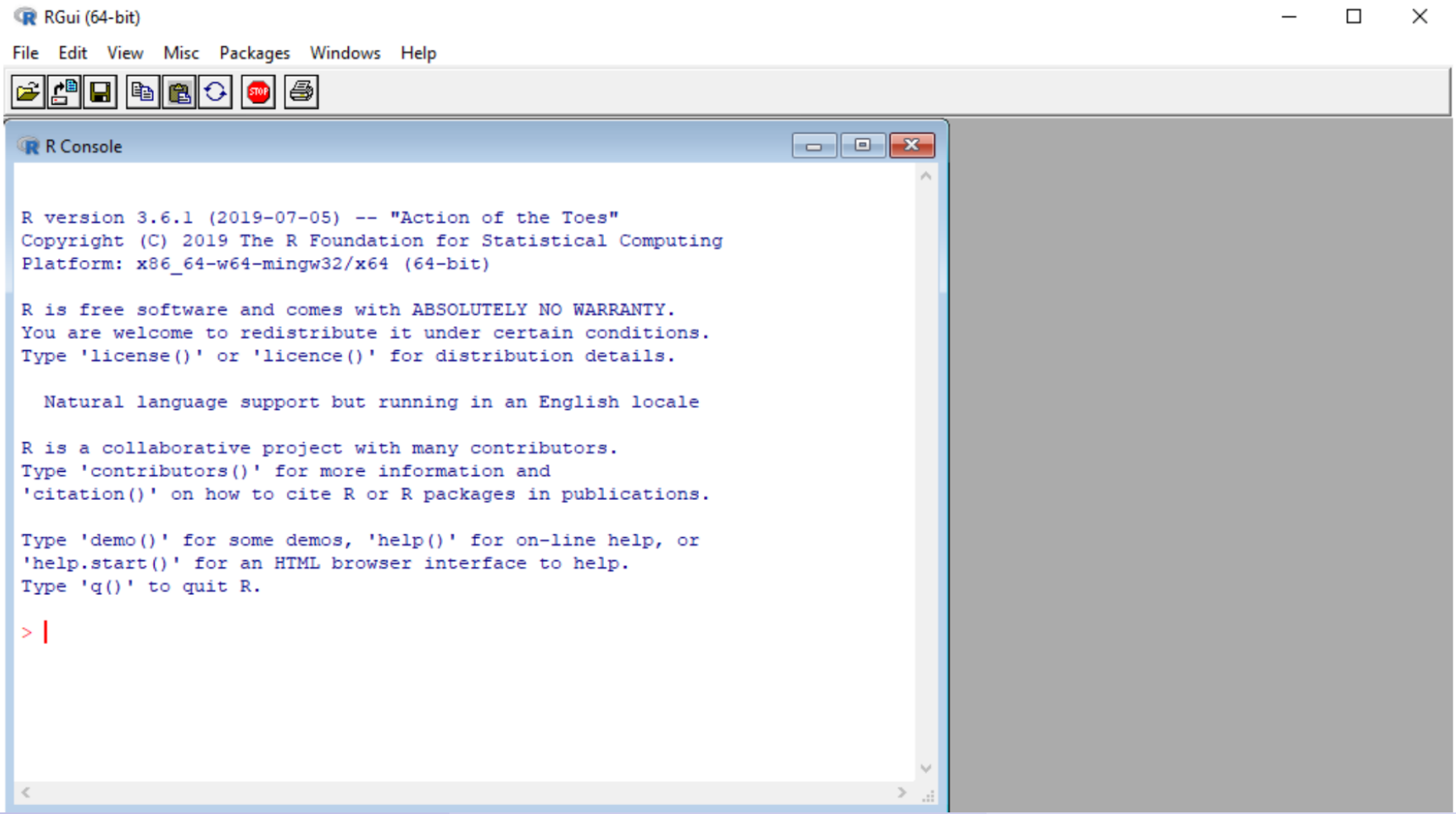


How to install R?

- You can download R at https://cran.r-project.org.
- There are links to download R for Windowns, Mac and Linux.
- Windows users should click the link Download R for Windows.
- As of this silde date, R is at version 4.0.0.
- The choice between using 32-bit and using 64-bit depends on whether the computer supports 64-bit
- Installing R is like installing any other program.
- For reference, see https://www.youtube.com/watch?v=NZxSA80lF1I

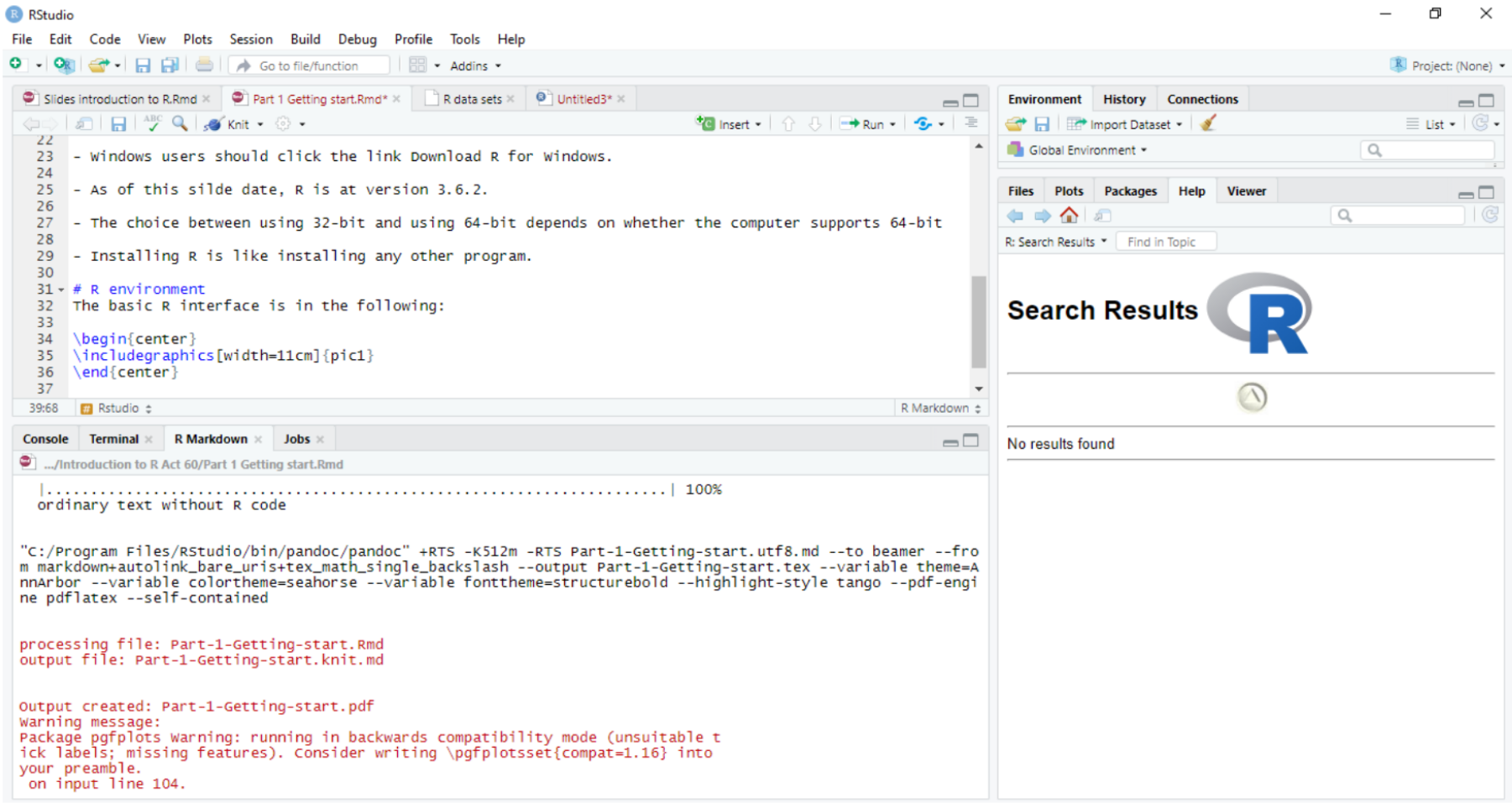
as an environment

The basic R interface is as the following:



Rstudio

Rstudio is the best Integrated Development Environment (IDE) for



is a language?

Example 1: You want to solve equation $x^5 + 3x^4 + x - 3 = 0$.

This is how you "talk" to :

```
# Asking R to create function f
f<-function(x){
  f<-x^5 + 3*x^4 + x - 3
}

# Asking R to find a root on interval (0,3)
answer<-uniroot(f,c(0,3))$root
```

and will answer:

```
## [1] 0.8625006
```

is a language?

Example 2: You want to calculate $\int_0^1 x^2 \times e^x \, dx$. This is how you talk to :

```
f<-function(x){ # asking R to create function f
  f<-x^2*exp(x)
}
n<-10000
answer<-0
# using numerical approximation
for (i in 1:n){
  answer<-answer + 1/n*f(i/n)
}
```

and will answer:

```
## [1] 0.7184177
```

Characteristics of R

- Can perform analysis as well as write program.
- Highly extensible - share packages.
  - R has base packages (base, stats, ect ...).
  - R can be extended by Comprehensive R Archive Network (CRAN) packages such as MASS.
  - A full list of R packages is avaiable at http://cran.r-project.org/web/packages
- Publicly avaiable set of real world data.

Install a package in R

To install a package in R, we run the following code (Your computer must be connected to the internet)

```
install.packages("package name")
# make sure that your computer is connected to the internet
```

To enable a package to use, we type:

```
library(package name)
```

For example, to install a package named "tidyverse":

```
install.packages("tidyverse") # It takes several minutes
library(tidyverse)
```

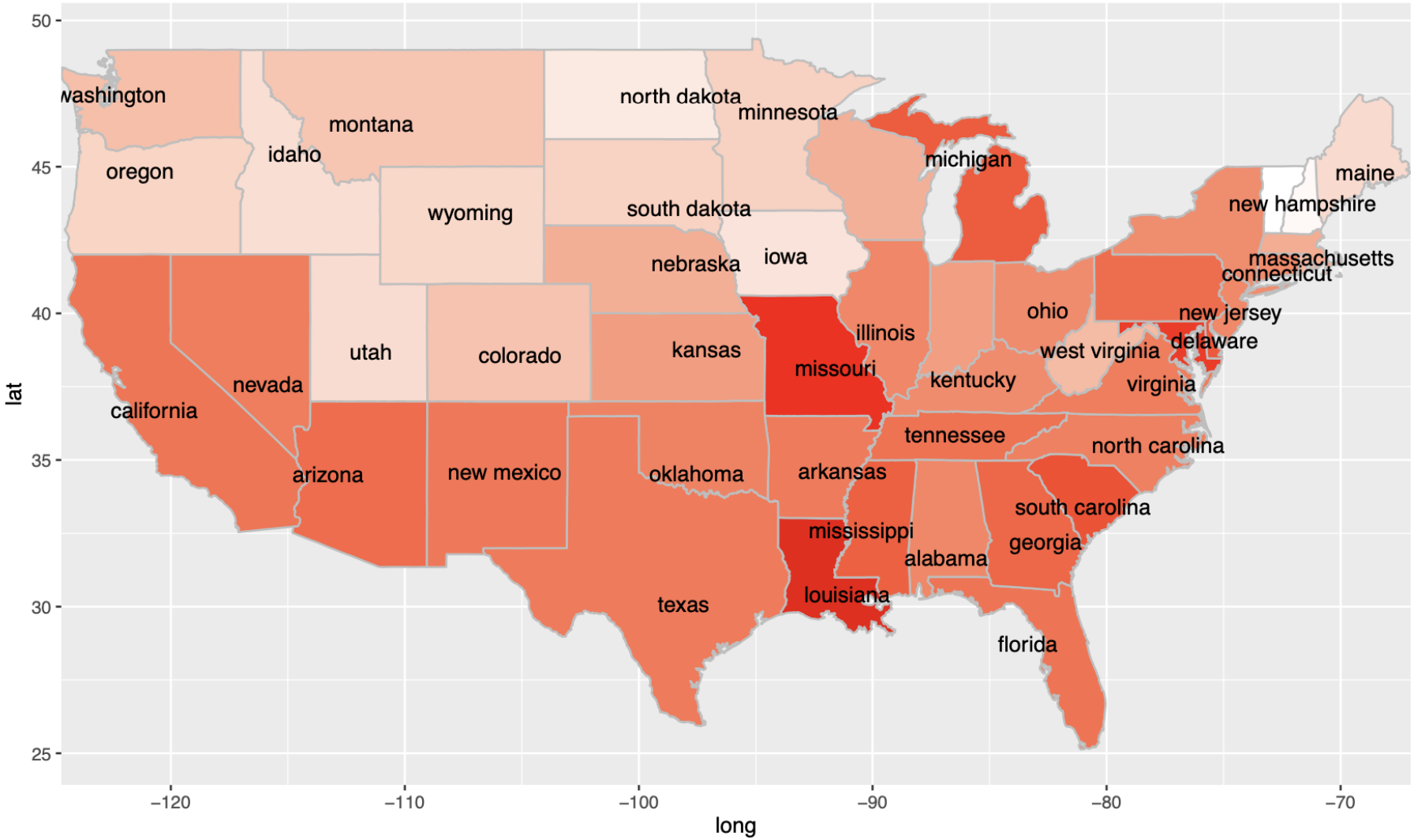
For example, to install a package named "dslabs":

```
install.packages("dslabs") # It takes several minutes
library(dslabs)
```

Loading an existing package in R

```
library(maps) #maps package provide area boundaries
dat<-map_data("state")
dat1<-murders%>%mutate(rate=total/population*10^5)
dat1$state<-tolower(dat1$state)
dat<-dat%>%mutate(rate=dat1$rate[match(dat$region,
                                       dat1$state)])
dat<-dat%>%group_by(region)%>%mutate(xm=mean(long))%>%
  mutate(ym=mean(lat))%>%ungroup()
p<-ggplot(dat,aes(long,lat,group=group,fill=rate))+
  geom_polygon(color="grey",show.legend=FALSE)+
  scale_x_continuous(expand=c(0,0))+
  scale_fill_gradientn(colors = c(rgb(1,1,1),
                       rgb(1,0,0),rgb(0.8,0,0)),trans = "sqrt'
p+geom_text(aes(x=xm,y=ym,label = region),
            check_overlap = TRUE)
```

Using package ggplot2 to visualize murders dataset



Real public datasets available in R

To show all avaiable datasets in R and attached libraries: *data()*.

To get a datasets for learning/researching:

```
data(airquality) # call data name airquality to R console
head(airquality) # show the head of data airquality
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

Working directory

When you want to save/load your works, including (your codes, workspaces or datasets) to/from a location on your computer, you must set an appropriate working directory.

To know the current working directory:

```
getwd()
```

To change the working directory:

```
setwd(PATH)
```

where PATH is the path (directory) to the folder where you saved your works or your datasets.

```
setwd("C:/Users/AD/Desktop/Tex file/Thu latex/Introduction to
MyData<-read.csv("CafeF.HSX.Upto20102020.csv")
# using "/" instead of "\" as in Window
```

Basics of R - basic math operators

Let's type on the console window:

```
options(tinytex.verbose = TRUE)
```

```
1 + 0.001
```

```
## [1] 1.001
```

```
2*pi - 3
```

```
## [1] 3.283185
```

```
exp(1)-exp(-1)
```

```
## [1] 2.350402
```

```
log(3.2) # natural logarithm
```

```
## [1] 1.163151
```

```
log(32,2) # the binary logarithm of 32
```

Basics of R - basic math operators

Calculate the following formulas:

- (a) $\frac{1}{4^{-6}}$
- (b) $\sqrt[4]{22}$
- (c) 4^{3+2}
- (d) $\frac{4+7}{12-2}$
- (e) $(12-5)^{4/3}$
- (f) $\frac{2^3}{3^2}$
- (g) $ln\left(\frac{2+3}{4-1}\right)$.

Basics of R - variable assignment

We can use "<-" or "=" for assigning value to a variable

```
x<-3.25 # variable name is x
x
```

```
## [1] 3.25
```

```
x^2+exp(x)-3
```

```
## [1] 33.35284
```

```
y=2
x^y
```

```
## [1] 10.5625
```

Variable names can contain any combination of alphanumeric characters along with periods (.) and underscores (_). However, they cannot start with a number or an underscore.

Basics of R - removing variables

```
123x<-3.25
# check if there is an error
```

```
_x<-3.25
# check if there is an error
```

To remove a variable, we use function *rm()*

```
x<-3.25
x
```

```
## [1] 3.25
```

```
rm(x)
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

Basics of R - numerical variables

- There are 4 main types of variables: **numeric**, **character (string)**, **Date/Time** and **logical**.
- **Numeric** is similar to float or double in other languages.

```
x<-5
is.numeric(x)
```

```
## [1] TRUE
```

- An **integer** variable is also a numerical variable.

```
x<-5
is.integer(x)
```

```
## [1] FALSE
```

- To assign an integer value to variable x:

```
x<-5L
```

Basics of R - numerical variables

Operator	Description
$+$	Addition
$-$	Subtraction
$*$	Multiplication
$/$	Division
$\hat{}$	Exponent
$exp()$	Natural Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

Basics of R - numerical variables

```
6.5 %/% 2
```

```
## [1] 3
```

```
6.5 %% 2
```

```
## [1] 0.5
```

```
((-1)/0)*(1/0)
```

```
## [1] -Inf
```

```
log(-2)
```

```
## [1] NaN
```

```
log(-2)*(1/0)
```

```
## [1] NaN
```

Basics of R - logical variables

- Logical variables can be either **TRUE** or **FALSE**.
- Numerically, **TRUE** is the same as 1 and **FALSE** is the same as 0.
- Logical variables are results from relational operations

```
x<-TRUE
x
```

```
## [1] TRUE
```

```
x*2
```

```
## [1] 2
```

```
y<-FALSE
x + y
```

```
## [1] 1
```

Basics of R - logical variables

Operator	Description
$<$	Less than
$>$	Greater than
$<=$	Less than or equal to
$>=$	Greater than or equal to
$==$	Equal to
$!=$	Not equal to
!	Not
&	And
|	Or

Basics of R - logical variables

```
x<-(1>=2)
x
```

```
## [1] FALSE
```

```
y<-(x==0)
y
```

```
## [1] TRUE
```

```
x+y
```

```
## [1] 1
```

```
x&y
```

```
## [1] FALSE
```

```
x|y
```

Basics of R - character variables

- In any language, **character variables** should be use with care.

```
x<-"Ice cream"
is.character(x)
```

```
## [1] TRUE
```

- Capital letter "I" and normal letter "i" are different.

```
x == "ice cream" # capital letter
```

```
## [1] FALSE
```

- Funtion *nchar(x)* count the characters in string *x*

```
nchar(x) # number of characters
```

```
## [1] 9
```

Basics of R - character variables

- Function *paste()* concatenate number of strings into a single string

```
x<-paste("We", "are", "the champion", sep = " ")
x
```

```
## [1] "We are the champion"
```

- Function *substr()* extract a substrings in a character vector.

```
substr("abcdefgh",3,4)
```

```
## [1] "cd"
```

- Function *sub()* and *gsub()* perform replacement of the first and all matches, respectively

```
sub("a","1234","abcdefga")
```

```
## [1] "1234bcdefga"
```

Basics of R - Date/time variables.

- Dealing with dates and times can be difficult in any language.
- Date variables store a date as number of days since January 1, 1970 while POSIXct stores a time as number of seconds since 07:00:00 Jan 1, 1970.

```
date1<-as.Date("2019-12-05")
as.numeric(date1) # number of days since Jan 01, 1970
```

```
## [1] 18235
```

```
time1<-as.POSIXct("1970-01-01 07:00:01")
as.numeric(time1) # number of seconds since 7AM Jan 01, 1970
```

```
## [1] -3599
```

```
as.numeric(substr(time1,12,13)) # extract hour from date2
```

```
## [1] 7
```

Basics of R - Date/time variables.

Base  does not well support to date/time variables, we use packages **lubridate** instead

```
library(lubridate)
today()
```

```
## [1] "2022-12-27"
```

```
now()# Universal Time Coordinated (UTC)
```

```
## [1] "2022-12-27 21:21:16 +07"
```

```
mdy("December 31st, 2021")
```

```
## [1] "2021-12-31"
```

```
dmy("31-Dec-2021")
```

```
## [1] "2021-12-31"
```

Basics of R - vertors (1)

- A vector is a collection of elements, all of the **same type**.
- Vectors play a crucial and helpful role **R**. **R** is also call a **vectorized** language.
- There are many ways to create a vector in R

```
x<-c(2,3,5,7,11,13) # 1st way to create a vector, "c" means co
x
```

```
## [1]  2  3  5  7 11 13
```

```
length(x)
```

```
## [1] 6
```

```
s<-c("R", "VBA", "Python","C++")
s
```

```
## [1] "R"      "VBA"    "Python" "C++"
```

Basics of R - vertors (2)

To access elements of a vector, we uses **[]**

```
x<-5:10 # 2nd way to create a vector
x[5]
```

```
## [1] 9
```

```
x[2:4]
```

```
## [1] 6 7 8
```

Third way to create a vector

```
x<-seq(1,2,0.2) # from 1 to 2 with jump = 0.3
length(x)
```

```
## [1] 6
```

```
x<-seq(1,2,length=11) # from 1 to 2 with length 11
x
```

Basics of R - vertors (3)

```
x<-1:5
x*2
```

```
## [1]  2  4  6  8 10
```

```
x^2
```

```
## [1]  1  4  9 16 25
```

Example 1: Calculate $\sum_{n=1}^{100} \frac{1}{n^2}$. *We DO NOT need a loop for, indeed,*

```
n<-100
x<-1:n
sum(1/x^2)
```

```
## [1] 1.634984
```

Basics of R - vertors (4)

Example 2: Calculate $\sum_{n=0}^{100} \frac{1}{n!}$.

```
n<-100
x<-0:n
sum(1/factorial(x))
```

```
## [1] 2.718282
```

Example 3: Calculate $\int_0^2 e^x \, dx$ *and compare the result to* $e^2 - 1$. Note that $\int_a^b f(x) \, dx$ is approximated by

$$\int_a^b f(x) \, dx \sim \sum_{i=1}^{n} \frac{b-a}{n} \times f\left(a + \frac{i(b-a)}{n}\right) \quad \textit{with } n \textit{ is large}$$

Basics of R - vertors (5)

```
n<-10000
x<-seq(0+(2-0)/n,2,(2-0)/n)
(2-0)/n*sum(exp(x))
```

```
## [1] 6.389695
```

```
exp(2)-1 # Analytical solution
```

```
## [1] 6.389056
```

Basics of R - vertors (5)

```
n<-10000
x<-seq(0+(2-0)/n,2,(2-0)/n)
(2-0)/n*sum(exp(x))
```

```
## [1] 6.389695
```

```
exp(2)-1 # Analytical solution
```

```
## [1] 6.389056
```

Example 4: The standard normal random variable has distribution function $\mathbb{P}(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$. *Calculate* $\mathbb{P}(-0.5 < X \leq 1)$.

Basics of R - vertors (5)

```
n<-10000
x<-seq(0+(2-0)/n,2,(2-0)/n)
(2-0)/n*sum(exp(x))
```

```
## [1] 6.389695
```

```
exp(2)-1 # Analytical solution
```

```
## [1] 6.389056
```

Example 4: The standard normal random variable has distribution function $\mathbb{P}(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$. *Calculate* $\mathbb{P}(-0.5 < X \leq 1)$. *Answer*

Basics of R - vertors (6)

- Logical operations on vectors result in a logical vector:

```
x<-1:7
x
```

```
## [1] 1 2 3 4 5 6 7
```

```
x<=3
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
sum(x<=3)
```

```
## [1] 3
```

```
any(x==1)
```

```
## [1] TRUE
```

```
all(x==1)
```

Basics of R - vertors (7)

Example 5: **AirPassengers** is a dataset (in vector) in **R**

- Is there any month that the average number of passengers is more than 650 ?
- What is the proportion of months that the average number of passengers are less than 300 ?

Basics of R - vertors (7)

Example 5: **AirPassengers** is a dataset (in vector) in **R**

- Is there any month that the average number of passengers is more than 650 ?
- What is the proportion of months that the average number of passengers are less than 300 ?

Answer

```
any(AirPassengers>650)
```

```
## [1] FALSE
```

```
sum(AirPassengers<300)/length(AirPassengers)
```

```
## [1] 0.5694444
```

Basics of R - operations between vectors

- Combine vectors

```
x<-1:3
y<-4:8
z<-9:10
c(x,y,z)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
x<-seq(1,2,0.5)
y<-c(3,4)
z<-seq(5,6,length=3)
c(x,y,z)
```

```
## [1] 1.0 1.5 2.0 3.0 4.0 5.0 5.5 6.0
```

Basics of R - operations between vectors

```
x<-"I am an actuary, "
y<-"what is your talent ?"
c(x,y)
```

```
## [1] "I am an actuary, "      "what is your talent ?"
```

```
z<-paste(x,y)
z
```

```
## [1] "I am an actuary,  what is your talent ?"
```

Basics of R - operations between vectors

- Subtract vector

```
x<-c(2,3,5,7,11,13,17,19,23,29)
x %% 2 == 1
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
y<-(x %% 2 == 1)
x[y]
```

```
## [1]  3  5  7 11 13 17 19 23 29
```

```
y<-c(10,1,4)
x[y]
```

```
## [1] 29  2  7
```

Basics of R - operations between vectors

- *Example 6 Taking the numbers of passengers in months of June out of AirPassengers dataset*

Basics of R - operations between vectors

- *Example 6 Taking the numbers of passengers in months of June out of AirPassengers dataset*

```
y<-0:11*12+6
AirPassengers[y]
```

```
##  [1] 135 149 178 218 243 264 315 374 422 435 472 535
```

Which month has the lowest average number of passengers ?

Basics of R - operations between vectors

- *Example 6 Taking the numbers of passengers in months of June out of AirPassengers dataset*

```
y<-0:11*12+6
AirPassengers[y]
```

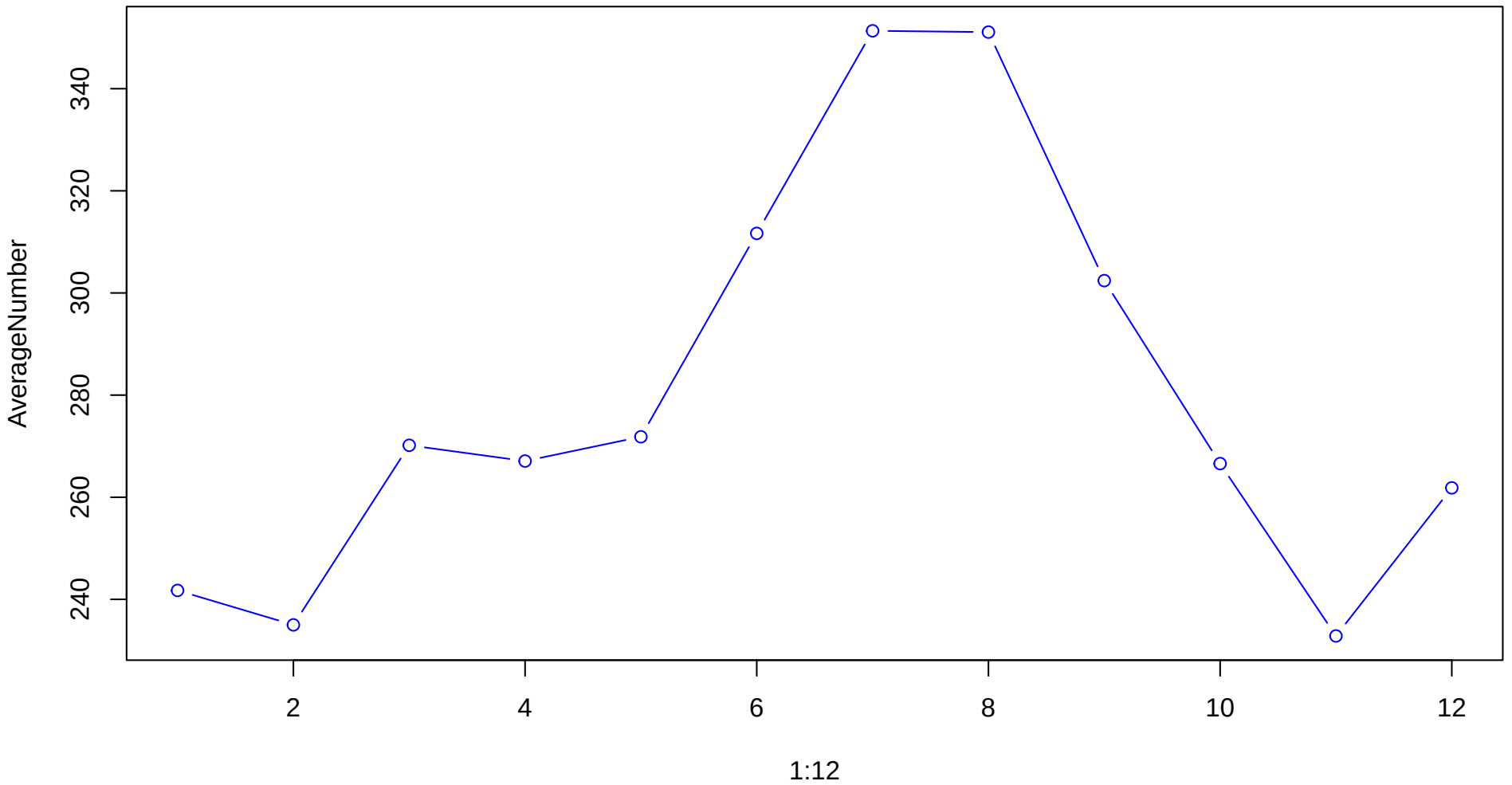
```
##  [1] 135 149 178 218 243 264 315 374 422 435 472 535
```

Which month has the lowest average number of passengers ?

```
AverageNumber<-rep(0,12)
for (i in 1:12){y<-0:11*12+i
AverageNumber[i]<-mean(AirPassengers[y])}
```

Basics of R - operations between vectors

```
plot(x=1:12,y=AverageNumber,col="blue",type="b")
```



Basics of R - operations between vectors

- Operation between vectors with **the same length**: Let $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$, then $\mathbf{x}\Delta\mathbf{y}$ will result in $\mathbf{z}$ where $z[i] = x[i]\Delta y[i]$ for all operation Δ.

```
x<-1:7
y<-7:1
x*y
```

```
## [1]  7 12 15 16 15 12  7
```

```
x<y
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
round(x/y,2)
```

```
## [1] 0.14 0.33 0.60 1.00 1.67 3.00 7.00
```

Basics of R - missing data

- Missing data plays a critical role in both statistics and computing.
- R uses**NA** to denote missing data.

```
x<-c(1,2,3,NA,5,NA)
is.na(x)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE  TRUE
```

```
x*2
```

```
## [1]  2  4  6 NA 10 NA
```

```
x>3
```

```
## [1] FALSE FALSE FALSE    NA  TRUE    NA
```

```
y<-c(NA,2,NA,4,5,6)
x*y
```

```
## [1] NA  4 NA NA 25 NA
```

Basics of R - often used functions with vectors

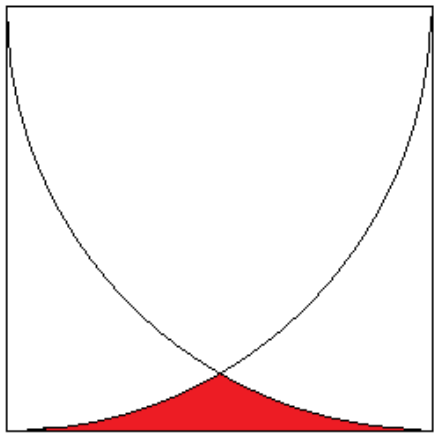
length(x)	:	number of elements in vector **x**
sum(x)	:	sum of all elements in vector **x**
mean(x)	:	average of all elements in vector **x**
sd(x)	:	standard deviation of all elements in vecto
prod(x)	:	product of all elements in vector **x**
min(x)	:	minimum value of all elements in vector **x**
max(x)	:	maximum value of all elements in vector **x**
sort(x,decreasing = TRUE)	:	sorts vector **x** in decreasing order
sort(x,decreasing = FALSE)	:	sorts vector **x** in increasing order

Basics of R - exercises

- Ex1: Calculate from *AirPassengers* dataset, **the proportion** of months with average number of passengers is
  - Less than 100?
  - More than 300 but less than 400 ?
  - More than 400 but less than 450 or more than 500 but less than 550 ?
- Ex 2: Given $\mathbf{x} = (31, 2, 11, 37, 5, 13, 17, 29, 23, 19, 3, 7)$. Assigning to vector **y** values in vector **x** but in reserve order.
- Ex 3: Let **x** = Nile (measurements of the annual flow of the river Nile, in 10^2 km^2, from 1871 to 1970).
  - What is the smallest and the largest values of annual flow?
  - In which year the annual flow is the smallest, is the largest?
  - In which year the annual flow is the 20^{th} largest?

Basics of R - exercises

- Ex 4: Calculate $\int_0^3 x^{1.5}\, e^{-0.3x}\, dx$
- Ex 5: Calculale A:



- Ex 6: Let **x** = presidents
  - How many **NA** values are in **x**?
  - Which years there are no observation?