Introduction to R

Research Informatics Core

February 8, 2023

Contents

1 Morning

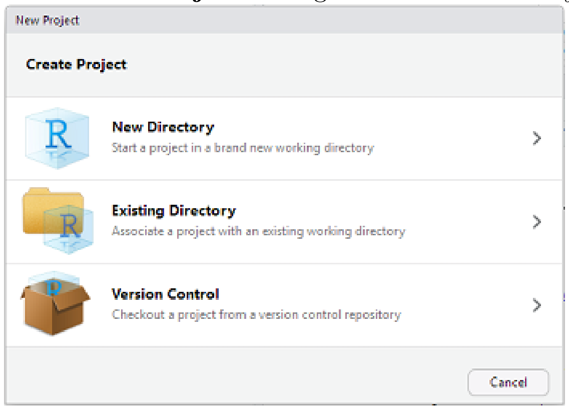
1.1 R Studio and simple R commands

TIPS

- Use Tab to auto complete
- Use up arrow to get previous command

1.1.1 Create a new project in R Studio

1. From *File* menu select *New Project...*
2. From **New Project** Dialog select *New Directory*



New Project

Create Project

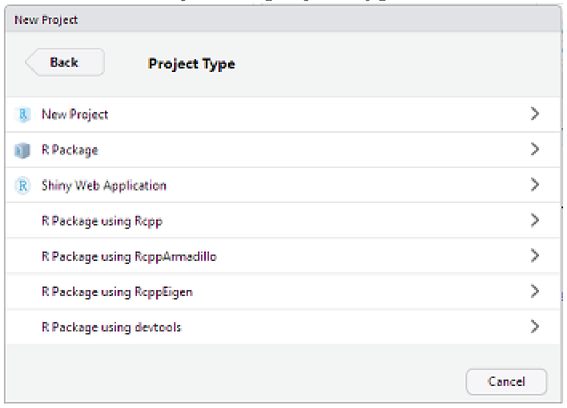
New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

Cancel

3. Select *New Project* as project type.



New Project

Back

Project Type

New Project

R Package

Shiny Web Application

R Package using Rcpp

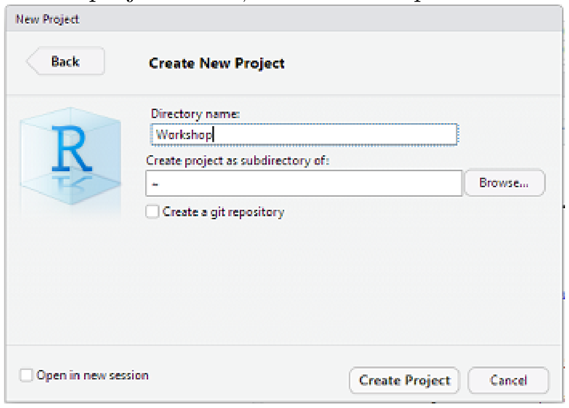
R Package using RcppArmadillo

R Package using RcppEigen

R Package using devtools

Cancel

4. Give a project name, i.e. "Workshop" and click **Create Project** button.



New Project

Back

Create New Project

Directory name:
Workshop

Create project as subdirectory of:
~

Browse...

Create a git repository

Open in new session

Create Project

Cancel

1.1.2 Create a new script.

1. From the *File* menu select *New File* > *R Script*
2. Save file as "my_Rscript.R"

Write the following commands in code editor of R Studio and run them using icon **Run** in R Studio

```
# Performing simple calculations
10+6
```

```
## [1] 16
```

```
10*6
```

```
## [1] 60
```

```
10/6
```

```
## [1] 1.666667
```

```
2**4     # 2 to the power of 4; same as 2^4
```

```
## [1] 16
```

```
1e-3 * 100  # 1e-3 means 1x10^-3
```

```
## [1] 0.1
```

```
# assign a value to a variable
x <- 100
# 1x <- 100 # 1x is an illegal variable name
y <- 1000
xy <- x+y
sqrt.x <- sqrt(x) # sqrt is square root function
sqrt.x
```

```
## [1] 10
```

```
(log10.sqrt.x <- log10(sqrt(x))) # use () to show the result right away
```

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

```
ls() # list all variables in the memory
```

```
## [1] "log10.sqrt.x" "sqrt.x"       "x"            "xy"           "y"
```

```
# string (character) variable
x <- "hello world"
x = "hello world"   # "=" can also be used for assignment
x
```

```
## [1] "hello world"
```

```
# logical variable
y <- TRUE
y <- T # the same as TRUE
y
```

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

```
z <- F
z
```

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

```r
y & z # logical operation: TRUE and FALSE => FALSE
```

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

```r
y | z # logical operation: TRUE or FALSE => TRUE
```

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

1.1.3 Getting help

```r
?sqrt # help information for function sqrt
```

1.2 Vector, factor, and list

1.2.1 Vector

```r
# create vectors
Num.Vec <- c(3.3, 2.2, 4.4, 1.1)
Num.Vec[2:4]
```

```
## [1] 2.2 4.4 1.1
```

```r
Char.Vec1 <- c("cow", "dog")
Char.Vec1
```

```
## [1] "cow" "dog"
```

```r
Char.Vec2 <- as.character(Num.Vec)
Log.Vec1 <- c(TRUE, TRUE, F, F)
Log.Vec2 <- c(rep(T, 5), rep(F,5))

# check the type of vectors
typeof(Num.Vec)
```

```
## [1] "double"
```

```r
typeof(Char.Vec1)
```

```
## [1] "character"
```

```r
typeof(Log.Vec1)
```

```
## [1] "logical"
```

```r
is.numeric(Num.Vec)
```

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

```r
is.character(Num.Vec)
```

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

```r
is.character(Char.Vec1)
```

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

```r
is.logical(Log.Vec2)
```

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

```r
# Order the vector
# order of indices in Num.Vec: which index has the smallest value
order(Num.Vec)
```

```
## [1] 4 2 1 3
```

```r
# reorder the vector based on that order
Ordered.Num.Vec <- Num.Vec[order(Num.Vec)]
Ordered.Num.Vec
```

```
## [1] 1.1 2.2 3.3 4.4
```

```r
x <- c(1, 2, "dog", T) # what type of vector is created?
x
```

```
## [1] "1"    "2"    "dog"  "TRUE"
```

```
typeof(x)
```

```
## [1] "character"
```

1.2.2 Factor

```
# Create a factor
wk <- factor(c(1:5,2,2,5))
wk
```

```
## [1] 1 2 3 4 5 2 2 5
## Levels: 1 2 3 4 5
```

```
wk[1]
```

```
## [1] 1
## Levels: 1 2 3 4 5
```

```
wk[1] <- "M"  # Why doesn't this work?
wk[1] <- 1 # Fix the value
levels(wk) <- c("M","T","W","Th", "F")
wk
```

```
## [1] M  T  W  Th F  T  T  F
## Levels: M T W Th F
```

```
wk[1]
```

```
## [1] M
## Levels: M T W Th F
```

```
# Convert to vectors
as.character(wk)
```

```
## [1] "M"  "T"  "W"  "Th" "F"  "T"  "T"  "F"
```

```
as.numeric(wk) # be careful!
```

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

```
# Can we compare the elements of a factor? NO
wk[1] < wk[3]
```

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

```
# Create an ordered factor
wk <- factor(c(1:5,2,2,5),labels=c("M","T","W","Th", "F"), ordered=T)
wk
```

```
## [1] M  T  W  Th F  T  T  F
## Levels: M < T < W < Th < F
```

```
wk <- factor(wk, ordered = T)
wk
```

```
## [1] M  T  W  Th F  T  T  F
## Levels: M < T < W < Th < F
```

```
wk[1]
```

```
## [1] M
## Levels: M < T < W < Th < F
```

```
# Yes, we can compare now, because it is an ordered factor
wk[1] < wk[3]
```

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

1.2.3 List

```
# Create a list
myList <- list(c(100,10), T, 10.5, "apple")
myList[[1]]
```

```
## [1] 100  10
```

```
myList[[3]]
```

```
## [1] 10.5
```

```
# myList[[1]] is the vector
myList[[1]][2] <- 1000
myList
```

```
## [[1]]
## [1]  100 1000
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] 10.5
##
## [[4]]
## [1] "apple"
```

```
unlist(myList) # convert to a character vector
```

```
## [1] "100"   "1000"  "TRUE"  "10.5"  "apple"
```

1.3 Matrix

```
# create a 3x2 matrix
B <- matrix(c(1:6), nrow=3, ncol=2)
B
```

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

```
# element-wise addition
C <- B + B
C
```

```
##      [,1] [,2]
## [1,]    2    8
## [2,]    4   10
## [3,]    6   12
```

```
# bind by rows
rbind(B, C)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
## [4,]    2    8
## [5,]    4   10
## [6,]    6   12
```

```
# bind by columns
D <- cbind(B, c(7,8,9), C)
D
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    4    7    2    8
## [2,]    2    5    8    4   10
## [3,]    3    6    9    6   12
```

```
# dimensions of D, i.e. how many rows and how many columns
dim(D)
```

```
## [1] 3 5
```

```
D[1, 1:5]
```

```
## [1] 1 4 7 2 8
```

```
D[1:2, ]
```

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

```
D[, 1:3]
```

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

```r
# transpose the matrix B
t(B)
```

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

```r
# convert the matrix B to a vector by column
as.vector(B)
```

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

```r
# convert the matrix B to a vector by row

as.vector(t(B))
```

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

1.4 Data frame

1. Create a data.frame

```
n <- c(2, 3, 5)
s <- c("aa", "bb", "cc")
b <- c(TRUE, FALSE, TRUE)
df <- data.frame(n, s, b)       # df is a data frame
df
```

```
##   n  s     b
## 1 2 aa  TRUE
## 2 3 bb FALSE
## 3 5 cc  TRUE
```

2. Access the element of a data.frame

```
df[1,1]
```

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

```
df[1:2, ]
```

```
##   n  s     b
## 1 2 aa  TRUE
## 2 3 bb FALSE
```

3. Use a built-in data frame

```
?mtcars
dim(mtcars)
```

```
## [1] 32 11
```

```
head(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
```

4. Select an item (cell) from a data frame. By index and by row and column names

```
mtcars[1, 1]
```

```
## [1] 21
```

```
mtcars["Mazda RX4", "mpg"]
```

```
## [1] 21
```

5. Select rows from a data frame By index and by row names.

```
mtcars[c(1, 3), ]
```

```
##             mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21.0   6  160 110 3.90 2.62 16.46  0  1    4    4
## Datsun 710 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1
```

```
mtcars[c("Mazda RX4", "Datsun 710"), ]
```

```
##              mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21.0   6  160 110 3.90 2.62 16.46  0  1    4    4
## Datsun 710 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1
```

6. Select rows based on criteria

```
mtcars[mtcars$mpg<=15, ]
```

```
##                      mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
```

7. Select a column from a data frame

```
mtcars[[1]] # as a vector
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
## [16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
## [31] 15.0 21.4
```

```
mtcars[1] #as a data frame
```

```
##                      mpg
## Mazda RX4           21.0
## Mazda RX4 Wag       21.0
## Datsun 710          22.8
## Hornet 4 Drive      21.4
## Hornet Sportabout   18.7
## Valiant             18.1
## Duster 360          14.3
## Merc 240D           24.4
## Merc 230            22.8
## Merc 280            19.2
## Merc 280C           17.8
## Merc 450SE          16.4
## Merc 450SL          17.3
## Merc 450SLC         15.2
## Cadillac Fleetwood  10.4
## Lincoln Continental 10.4
## Chrysler Imperial   14.7
## Fiat 128            32.4
## Honda Civic         30.4
## Toyota Corolla      33.9
## Toyota Corona       21.5
## Dodge Challenger    15.5
## AMC Javelin         15.2
## Camaro Z28          13.3
## Pontiac Firebird    19.2
## Fiat X1-9           27.3
## Porsche 914-2       26.0
## Lotus Europa        30.4
## Ford Pantera L      15.8
## Ferrari Dino        19.7
## Maserati Bora       15.0
## Volvo 142E          21.4
```

8. Sort a data frame by a column.

```
ordered.mtcars <- mtcars [order(mtcars$mpg),]
head(ordered.mtcars)
```

```
##                      mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
```

1.5 Save history and R data

```
savehistory(file = "my.Rhistory")

# Save all objects in the current R session using save.image() function.
# Where it will be saved to?
save.image(file = "all.RData")
```

```
#Save a particular set of objects using save() function
ls() # list the objects in current R session
```

```
##  [1] "b"               "B"               "C"               "Char.Vec1"
##  [5] "Char.Vec2"       "D"               "df"              "Log.Vec1"
##  [9] "Log.Vec2"        "log10.sqrt.x"    "myList"          "n"
## [13] "Num.Vec"         "ordered.mtcars"  "Ordered.Num.Vec" "s"
## [17] "sqrt.x"          "wk"              "x"               "xy"
## [21] "y"               "z"
```

```
save(ordered.mtcars, x , file = "my_data.RData")

# Quit the session
q()

# Restart RStudio
loadhistory(file = "my.Rhistory")
history()
# Use up arrow to see if you fetch the commands

#Load R data file
load(file = "my_data.RData")
```

1.6 Write a user-defined function

Given a numeric vector, please write a function to get the maximum value of the vector. This is just for our practice, as the max function already exists in R. Use indentation (tab) to make codes more human readable.

```r
get.max <- function(x)
{
    max <- x[1]
    for (i in x)
    {
        if (i > max)
        {
            max <- i
        }
    }
    return(max)
}

a <- c(23.3, 1, 3, 55, 6)
get.max(a)
```

```
## [1] 55
```

2 Afternoon

2.1 Read and write a file

1. Read the file https://https://uic-ric.github.io/workshop-data/R/birth_weight.txt into a data frame
2. Sort the data frame by mother's age, and write to a new file

- Data frame columns
  - bwt: baby's birth weight in ounce (low bwt: < 88 ounces)
  - smoke: 0 - mother is not a smoker, 1 - smoker
  - parity: 0 - child first born, 1 - otherwise
  - gestation: length of pregnancy in days
  - age: mother's age in years
  - height: mother's height in inches
  - weight: mother's pregnancy weight in pounds

```
bw_data  <-
read.table ("https://uic-ric.github.io/workshop-data/R/birth_weight.txt",
header=T)

head(bw_data)
```

```
##   bwt gestation parity age height weight smoke
## 1 120       284      0  27     62    100     0
## 2 113       282      0  33     64    135     0
## 3 128       279      0  28     64    115     1
## 4 108       282      0  23     67    125     1
## 5 136       286      0  25     62     93     0
## 6 138       244      0  33     62    178     0
```

```
data.ordered.by.age <- bw_data[order(bw_data$age), ]
head(data.ordered.by.age)
```

```
##     bwt gestation parity age height weight smoke
## 470 114       283      1  15     64    117     1
## 400 120       271      1  17     64    142     1
## 429 123       323      1  17     64    140     0
## 537 141       284      1  17     64    105     0
## 561 144       289      1  17     69    130     1
## 812 124       284      1  17     62    112     0
```

```
write.table(data.ordered.by.age, file="birth_weight_ordered_by_age.txt", sep="\t",
   quote=F, row.names=F)
```

2.2 Histogram, boxplot, and scatter plot

If you closed R, execute the following

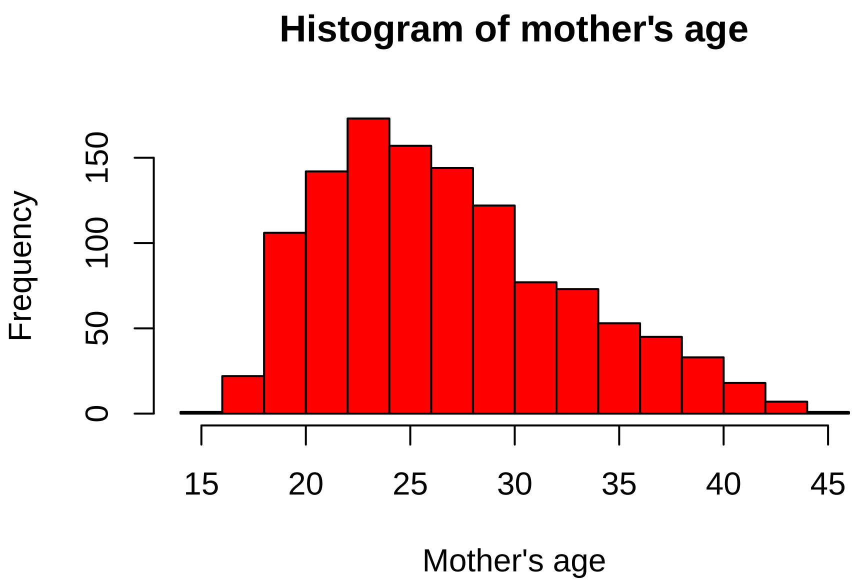
```
bw_data  <-
read.table ("https://uic-ric.github.io/workshop-data/R/birth_weight.txt",
header=T)
```

```
head(bw_data)
```

```
##   bwt gestation parity age height weight smoke
## 1 120       284      0  27     62    100     0
## 2 113       282      0  33     64    135     0
## 3 128       279      0  28     64    115     1
## 4 108       282      0  23     67    125     1
## 5 136       286      0  25     62     93     0
## 6 138       244      0  33     62    178     0
```

2.2.1 Histogram

```
hist(bw_data$age, main="Histogram of mother's age", xlab="Mother's age", col="red")
```

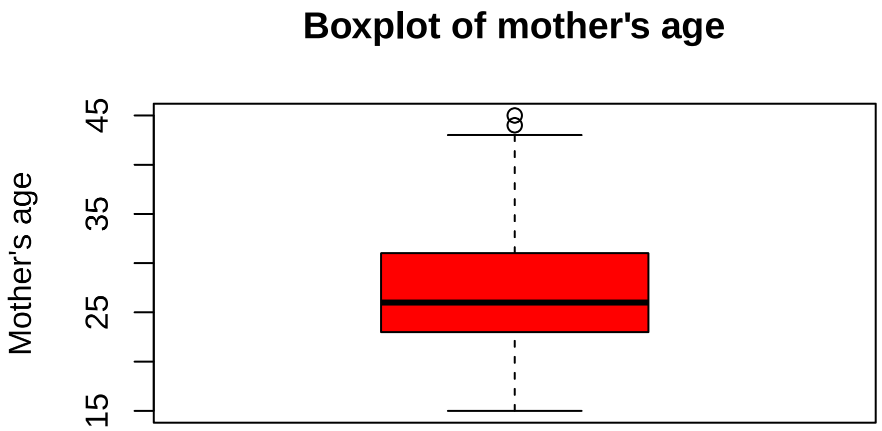


```
# save plot to a file
pdf(file="hist.age.pdf")
hist(bw_data$age, main="Histogram of mother's age", ylab="Mother's age", col="red")
dev.off()
```

```
## pdf
##   2
```

2.2.2 Boxplot

```
boxplot(bw_data$age, main="Boxplot of mother's age", ylab="Mother's age", col="red")
```

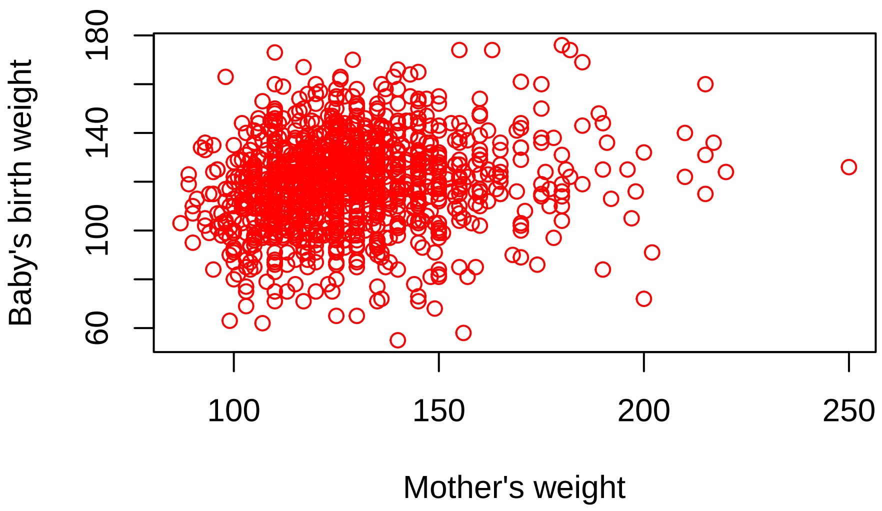


```
# save plot to a file
pdf(file="boxplot.age.pdf")
boxplot(bw_data$age, main="Boxplot of mother's age", ylab="Mother's age", col="red")
dev.off()
```

```
## pdf
##   2
```

2.2.3 Scatterplot

```
plot(bw_data$weight, bw_data$bwt, xlab="Mother's weight", ylab="Baby's birth weight",
   col="red")
```

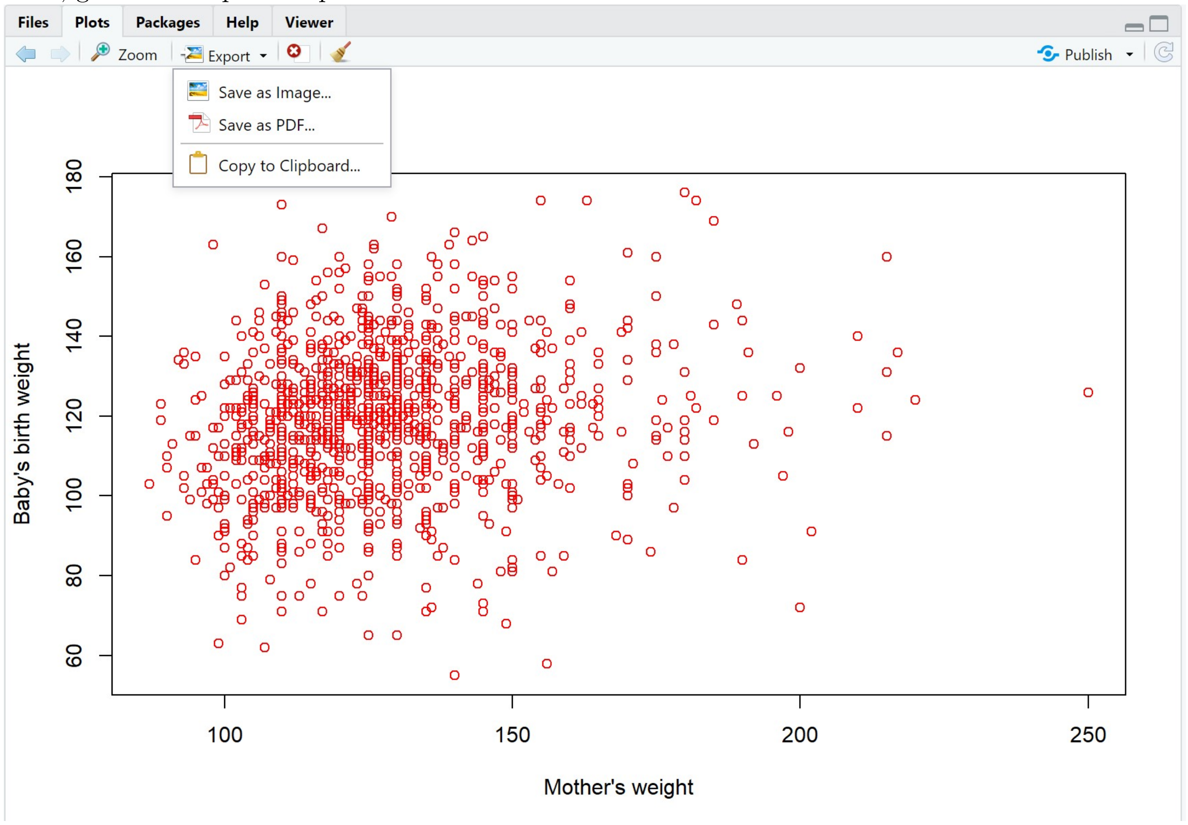


```
# save plot to a file
pdf(file="scatter.plot.weight.pdf")
plot(bw_data$weight, bw_data$bwt, xlab="Mother's weight", ylab="Baby's birth weight",
   col="red")
dev.off()
```

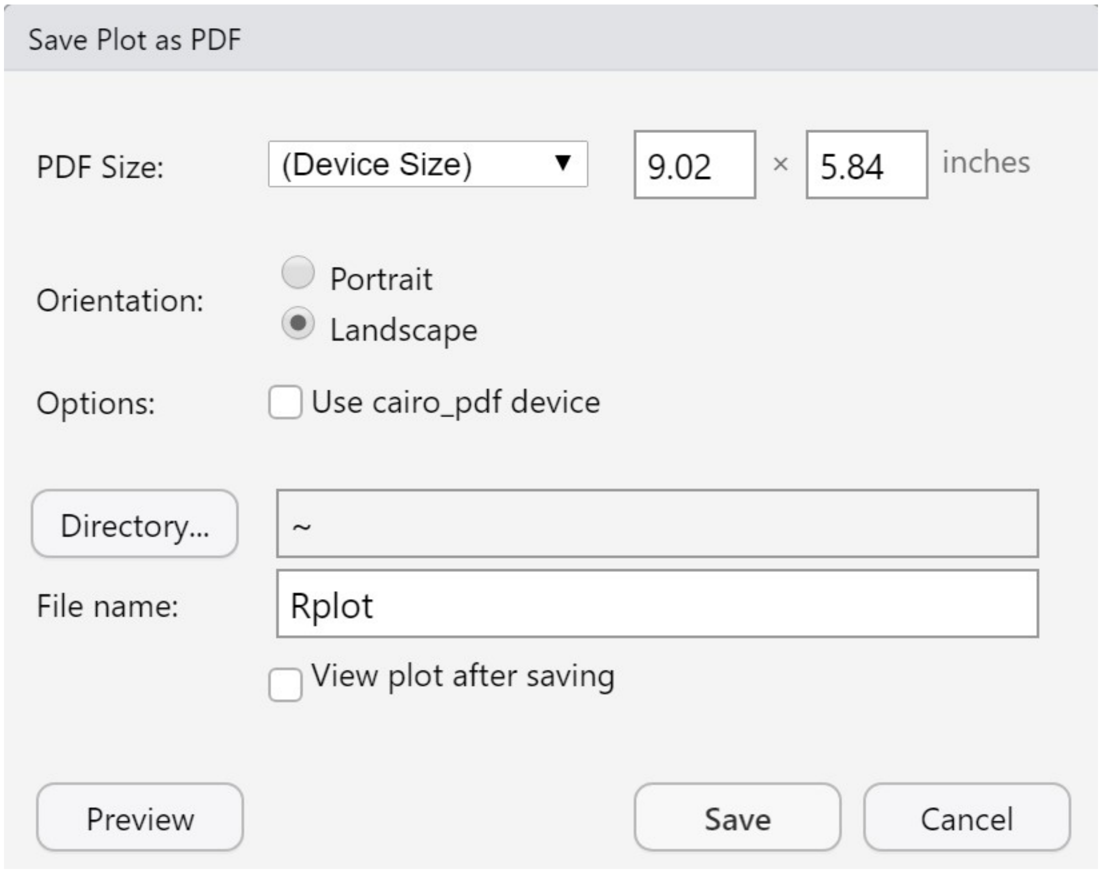
```
## pdf
##   2
```

2.2.4 Using R Studio plot window to save and review plots

1. Another way to save a plot into a PDF file is to use the window in the Plots tab of the lower right pane. First, go to the Export drop down menu



2. Choose Save as PDF, then specify the directory and file name and click Save


Save Plot as PDF

PDF Size: (Device Size) 9.02 × 5.84 inches

Orientation: ( ) Portrait [✓ Landscape]

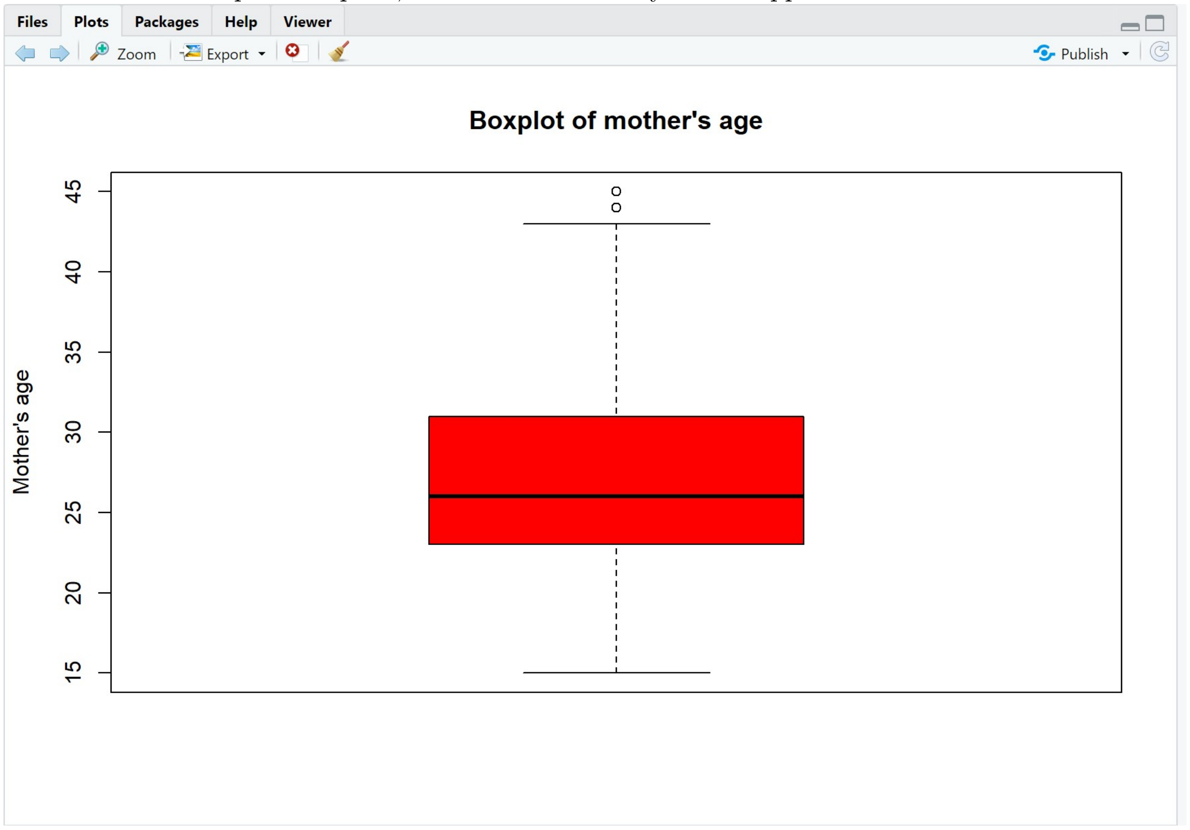
Options: ☐ Use cairo_pdf device

Directory... ~

File name: Rplot

☐ View plot after saving

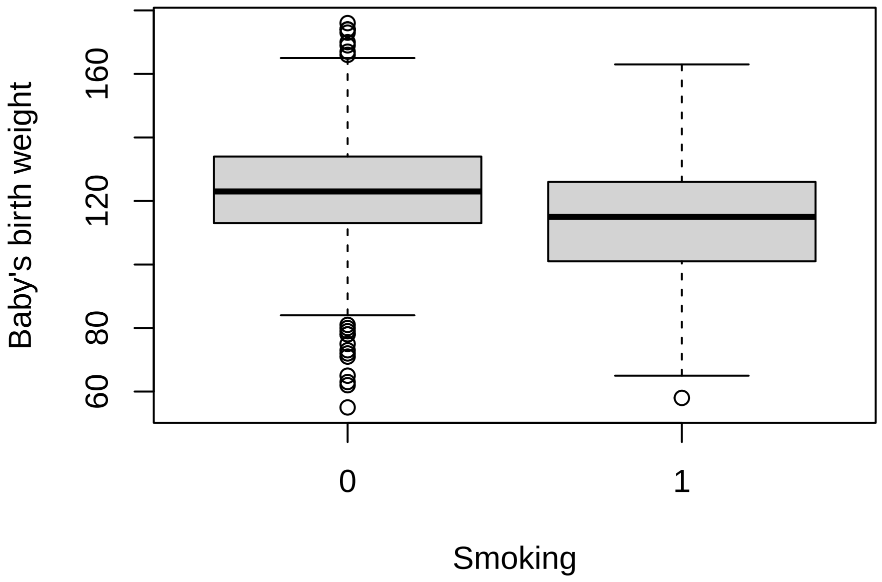
3. To review and save previous plots, use the left arrow key on the upper left corner



2.3 Students t-test

- Use the data in birth_weight.txt.
- Make boxplots for baby's birth weight (bwt), for smoker and non-smoker, respectively.
- Use T-test to test if the baby's birth weights are different between smoking mother and non-smoking mother.

```r
# plot birth weight as a function of smoke using tilde (~)
boxplot(bwt ~ smoke, data=bw_data, xlab="Smoking", ylab="Baby's birth weight")
```



```r
pdf(file="boxplot_bwt_smoking.pdf") # save plot to a pdf file
boxplot(bwt ~ smoke, data=bw_data, xlab="Smoking", ylab="Baby's birth weight")
dev.off()
```

```
## pdf
##   2
```

Performing t-test on baby's birth weight against mother's smokings status.

```r
t.test(bwt ~ smoke, data=bw_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  bwt by smoke
## t = 8.6265, df = 941.81, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##   7.158132 11.374153
## sample estimates:
## mean in group 0 mean in group 1
##        123.0853        113.8192
```

```r
ttest_result <- t.test(bwt ~ smoke, data=bw_data)
ttest_result
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  bwt by smoke
## t = 8.6265, df = 941.81, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##   7.158132 11.374153
## sample estimates:
## mean in group 0 mean in group 1
##        123.0853        113.8192
```

2.4 Fisher's exact test

There are patients participating a clinical trial for two different therapies. Please test if there is an association between therapy and cure.

	Cured	Not Cured
Therapy 1	34	12
Therapy 2	22	25

```r
# Run the following commands in R
my.data <- matrix(c(34,12,22,25), nrow=2, byrow=T)
my.data
```

```
##      [,1] [,2]
## [1,]   34   12
## [2,]   22   25
```

```r
# add row names and column names
rownames(my.data) <- c("Therapy 1", "Therapy 2")
colnames(my.data) <- c("Cured", "Not cured")
my.data
```

```
##           Cured Not cured
## Therapy 1    34        12
## Therapy 2    22        25
```

```r
fisher.test(my.data)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  my.data
## p-value = 0.01083
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.241007 8.511384
## sample estimates:
## odds ratio
##   3.177804
```

2.5 False discovery rate

2.5.1 Overview

1. Randomly generate vector `wt` (size=20) with mean 10 and standard deviation 3.
2. Randomly generate vector `ko` (size=20) with mean 10 and standard deviation 3.
3. Use T-test to test if `wt` is different from `ko`, and get p value for the test.
4. Repeat this procedure 10000 times
5. Can you find how many $p < 0.05$?
6. Calculate false discovery rate, a.k.a FDR or q value.

2.5.2 Code

Create an empty vector to store p values

```
pval <- vector()
```

Randomly generate x and y vector with the same means and perform t-test 10000 times

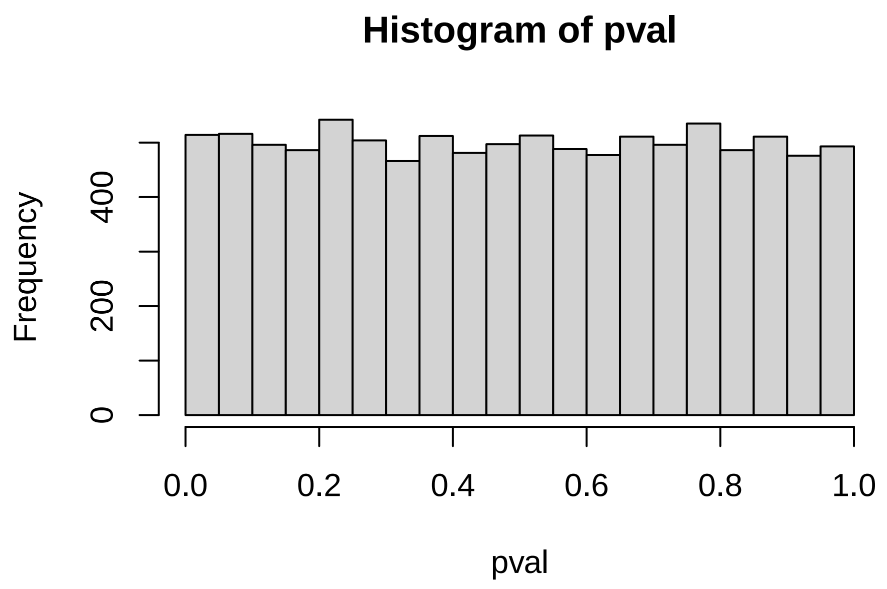
```
for (i in 1:10000)
{
    wt <- rnorm(20, mean=10, sd=3)
    ko <- rnorm(20, mean=10, sd=3)
    pval[i] <- t.test(wt, ko)$p.value
}
```

How many tests have $p < 0.05$?

```
length(pval[pval<0.05])
```

```
## [1] 514
```

```
hist(pval)
```



```
summary(pval)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## 0.0002425 0.2454379 0.4981883 0.4977781 0.7500203 0.9997447
```

Perform FDR correction

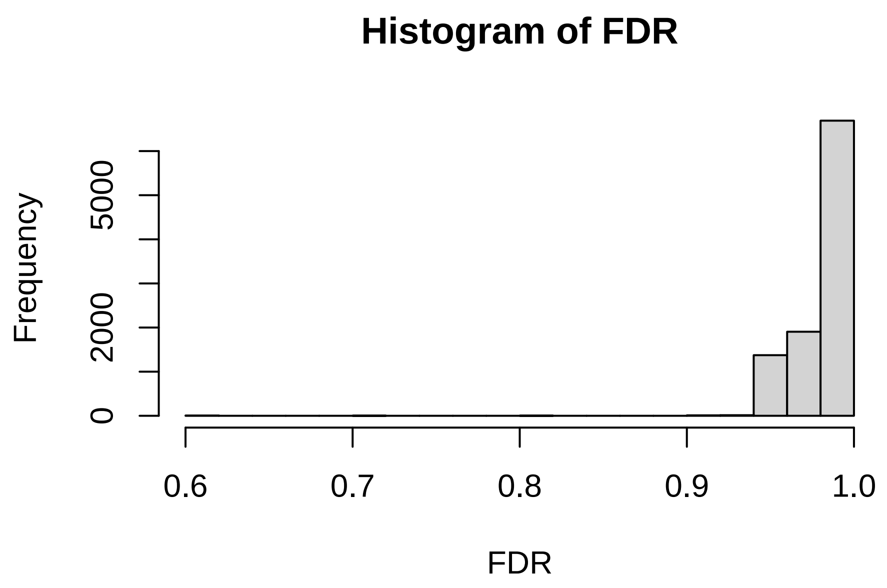
```
FDR <- p.adjust(pval, method="BH")
```

Now, how many tests have $q < 0.05$?

```
length(FDR[FDR<0.05])
```

```
## [1] 0
```

```
hist(FDR)
```



```
summary(FDR)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.6167  0.9760  0.9936  0.9845  0.9946  0.9997
```

2.6 Principal Component Analysis (PCA)

1. Load the example data

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

2. Get subsets of the data frame: just the data (numbers), and just the species names

```
ir <- iris[, 1:4]
ir.species <- factor(iris[, 5])
```

3. To examine variability of all numeric variables

sapply: traverse over a set of data like a list or vector, and calling the specified function for each item.

```
sapply(ir,var)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##    0.6856935    0.1899794    3.1162779    0.5810063
```

4. The range of variability is big in this context. Thus, we should standardize the variables with scale() function. Set scale equal to TRUE in the call to prcomp to standardize the variables prior to the application of PCA

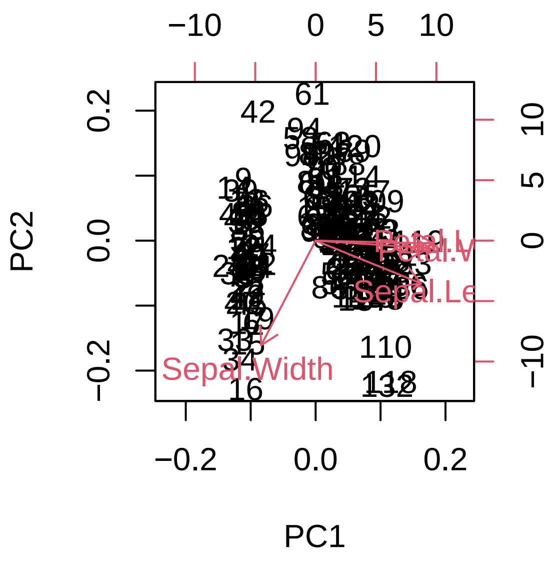
```
pca <- prcomp(ir, scale=TRUE)
```

5. Look at the pca$x object: principal components

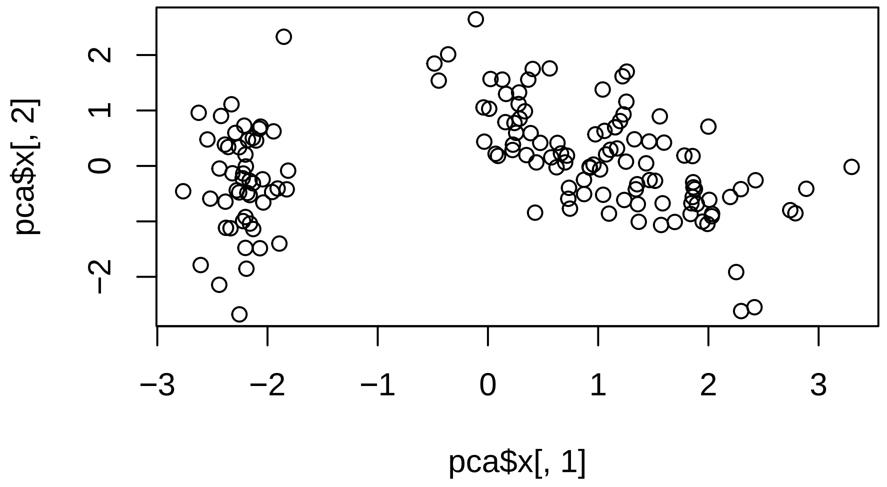
```
head(pca$x)
```

```
##            PC1        PC2         PC3          PC4
## [1,] -2.257141 -0.4784238  0.12727962  0.024087508
## [2,] -2.074013  0.6718827  0.23382552  0.102662845
## [3,] -2.356335  0.3407664 -0.04405390  0.028282305
## [4,] -2.291707  0.5953999 -0.09098530 -0.065735340
## [5,] -2.381863 -0.6446757 -0.01568565 -0.035802870
## [6,] -2.068701 -1.4842053 -0.02687825  0.006586116
```

```
biplot(pca)
```



```r
# simple plot
plot(pca$x[,1], pca$x[,2])
```

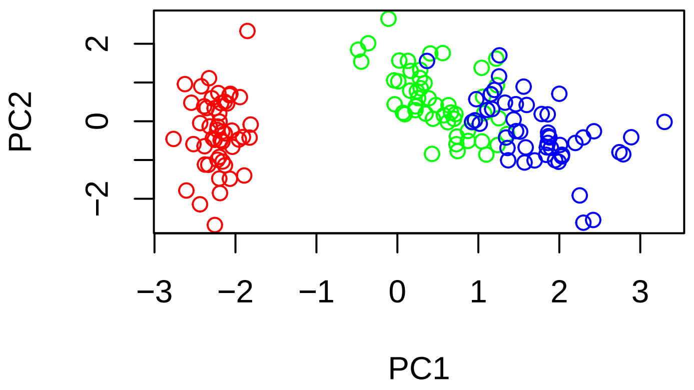


6. Define colors for each species. We'll do this by treating the species names as a factor, then renaming the factor levels with colors.

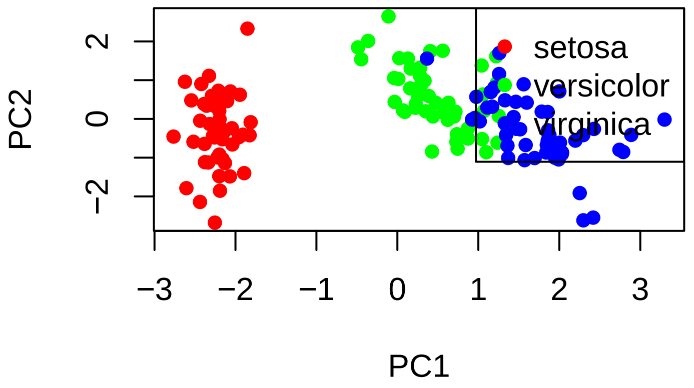
```r
ir.colors <- ir.species
levels(ir.colors) <- rainbow(length(levels(ir.colors)))
```

7. PCA plot of PC1 vs PC2

```
# add colors and axis labels
plot(pca$x[,1], pca$x[,2], col=as.character(ir.colors), xlab="PC1", ylab="PC2")
```



```
# change the symbols plotted to solid circles
plot(pca$x[,1], pca$x[,2], col=as.character(ir.colors), xlab="PC1", ylab="PC2", pch=16)
# add a legend
legend('topright', legend = levels(ir.species),
       col = levels(ir.colors), pch = 16)
```



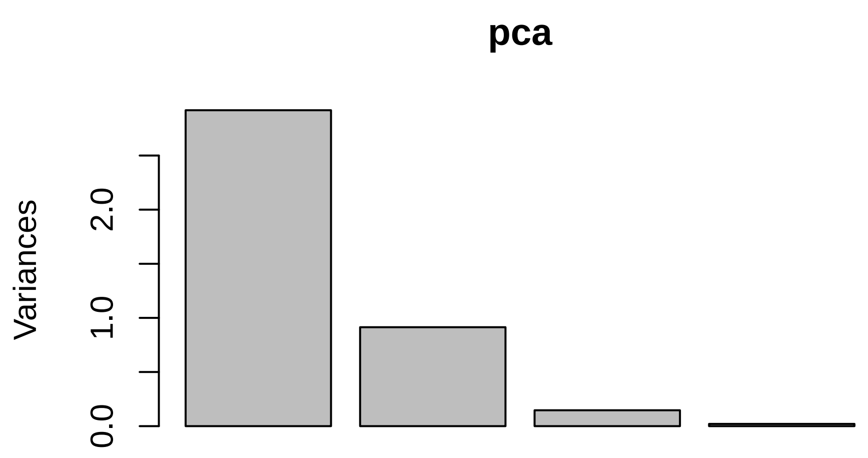
8. Check the percent variance for each Principal Component (PC)

```
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

9. Generate scree plot to visualize the relative proportion of variance

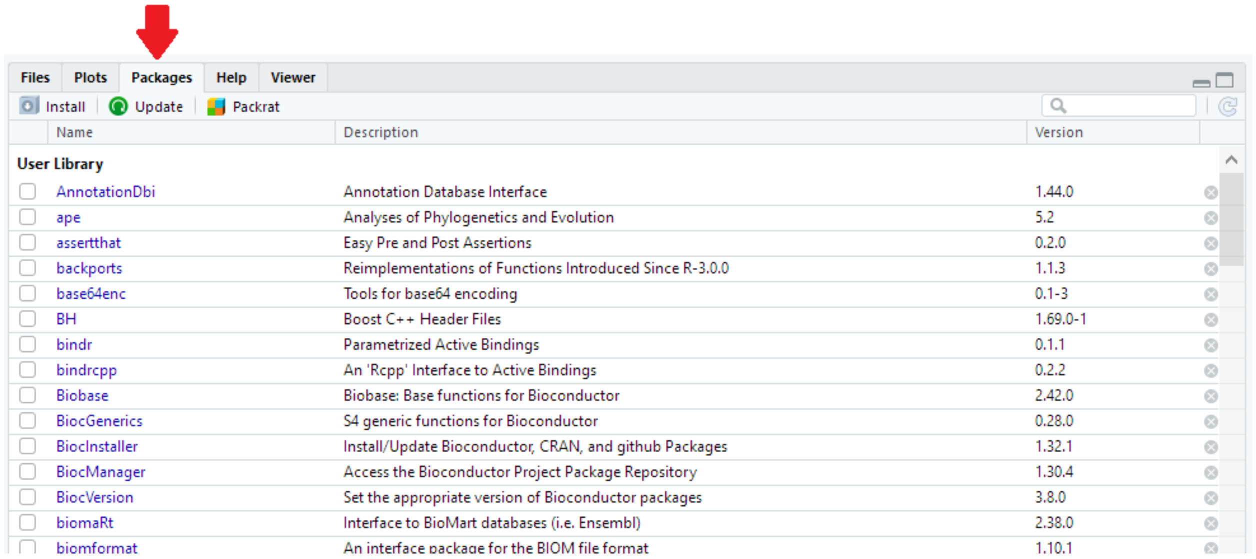
```
screeplot(pca)
```


pca
Variances
0.0
1.0
2.0

2.7 Install CRAN and Bioconductor packages

2.7.1 Check if selected packages are installed?

In R Studio you can select **Packages** tab in lower right pane to view all installed packages.



Name	Description	Version
User Library		
AnnotationDbi	Annotation Database Interface	1.44.0
ape	Analyses of Phylogenetics and Evolution	5.2
assertthat	Easy Pre and Post Assertions	0.2.0
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.3
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.69.0-1
bindr	Parametrized Active Bindings	0.1.1
bindrcpp	An 'Rcpp' Interface to Active Bindings	0.2.2
Biobase	Biobase: Base functions for Bioconductor	2.42.0
BiocGenerics	S4 generic functions for Bioconductor	0.28.0
BiocInstaller	Install/Update Bioconductor, CRAN, and github Packages	1.32.1
BiocManager	Access the Bioconductor Project Package Repository	1.30.4
BiocVersion	Set the appropriate version of Bioconductor packages	3.8.0
biomaRt	Interface to BioMart databases (i.e. Ensembl)	2.38.0
biomformat	An interface package for the BIOM file format	1.10.1

Also possible to list via R command.

```
# List all installed packages
installed.packages()
```

Check if select packages are installed.

```
c("ggplot2", "ComplexHeatmap") %in% rownames(installed.packages())
```

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

2.7.2 Install CRAN package *ggplot2*

If you were asked for a CRAN mirror site, please select one which is close to your location: e.g. “USA (IN)”

```
install.packages("ggplot2")

# load the package
library(ggplot2)
```

2.7.3 Install Bioconductor package: “ComplexHeatmap”

```
if (!requireNamespace("BiocManager"))
    install.packages("BiocManager")
BiocManager::install("ComplexHeatmap", update=F)

library(ComplexHeatmap)
```

2.8 Heatmap

1. Use ComplexHeatmap pacakge

```
library(ComplexHeatmap)
```

```
## Loading required package: grid

## ========================================
## ComplexHeatmap version 2.14.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

2. Load the example data

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

3. Get subset of the data frame: just the data (numbers)

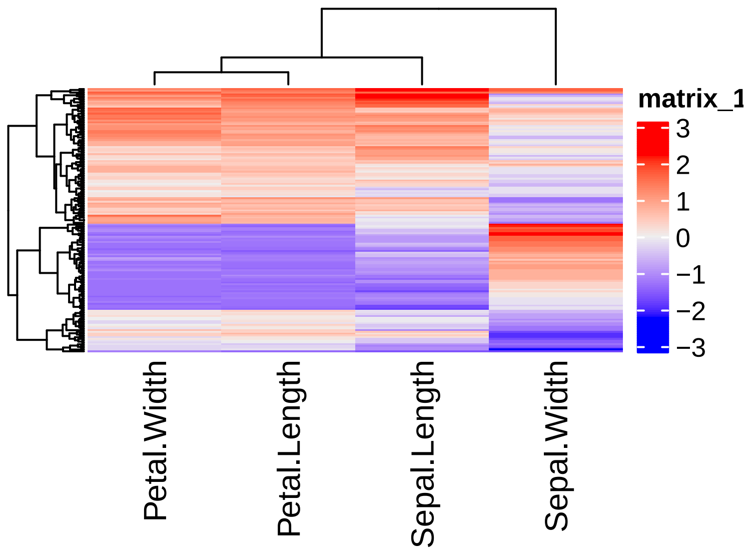
```
ir <- iris[, 1:4]
```

4. z-score (scale) transformation on columns so that we can see relative differences

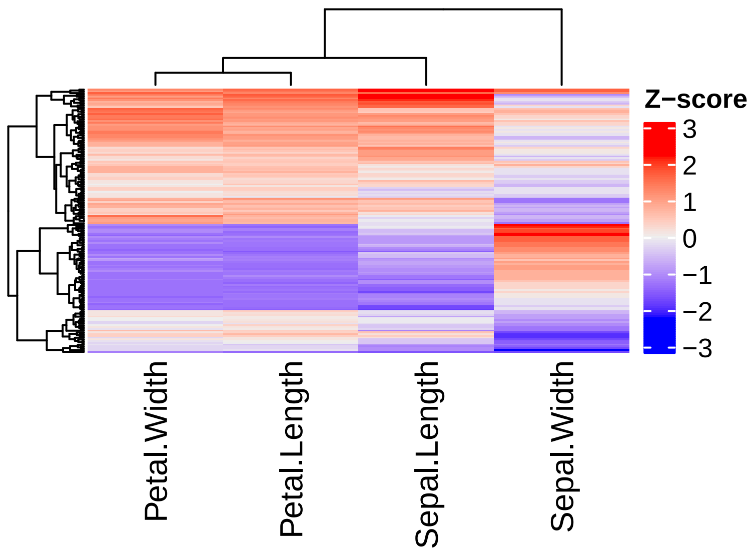
```
std.ir <- scale(ir)
```

5. Generate basic heatmaps

```
# heatmap with default settings
Heatmap(std.ir)
```



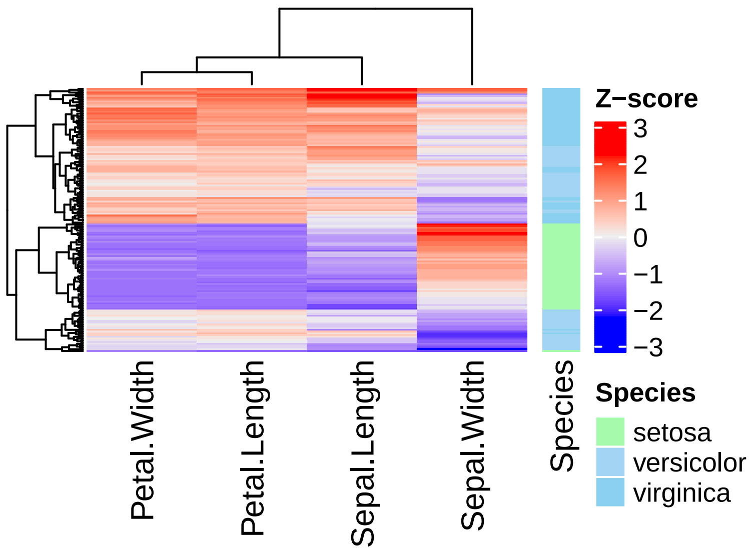
```
# add a label to the color key
Heatmap(std.ir, name = "Z-score")
```



6. Add annotations to heatmap

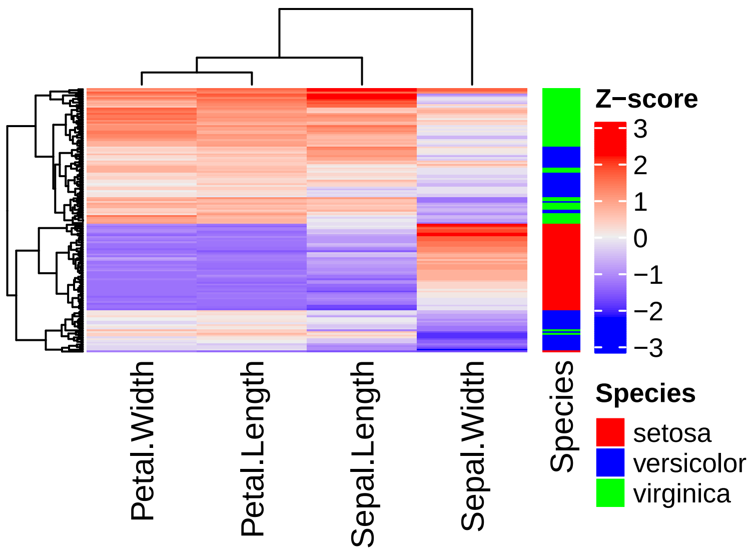
Add species name to heatmap

```
ir.species <- data.frame(Species=iris[, 5])
# add the species names to the rows: called "rowAnnotation"
Heatmap(std.ir,name="Z-score") + rowAnnotation(df=ir.species)
```



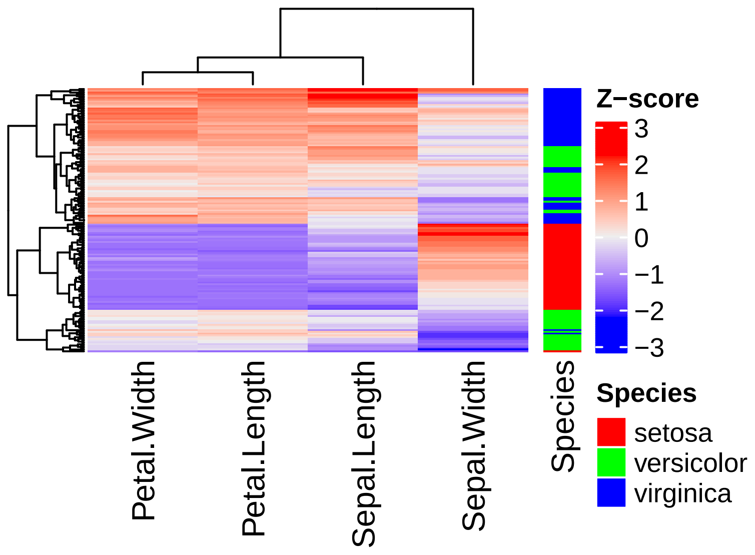
Add custom colors for species

```
# pick our own colors for the species
Heatmap(std.ir,name="Z-score") + rowAnnotation(df=ir.species,
   col=list(Species=c("setosa"="red","versicolor"="blue","virginica"="green")))
```



We can also do this by storing the separate parts of the plot in variables. This makes it easier to modify, and mix and match styles

```
heatmap <- Heatmap(std.ir,name="Z-score")
rowcolors <- rowAnnotation(df=ir.species,
   col=list(Species=c("setosa"="red","versicolor"="green","virginica"="blue")))
heatmap + rowcolors
```


Z−score
3
2
1
0
−1
−2
−3
Petal.Width
Petal.Length
Sepal.Length
Sepal.Width
Species
Species
setosa
versicolor
virginica