R statistics

7/28/2022

1.Descriptive statistics (by Kaiyu)

```r
# Load data
mydata <- mtcars
mydata
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

There are several R functions designed to provide a range of descriptive statistics at once. For example:

(1) summary()

- mean,median,25th and 75th quartiles,min,max

```
summary(mydata)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

(2) psych::describe()

- output: item name ,item number, nvalid, mean, sd, median, mad, min, max, skew, kurtosis, se

```
#install.packages('psych')
#install.packages("psych", repos = "https://personality-project.org/r/", type="source")
library(psych)
describe(mydata)
```

```
##      vars  n   mean     sd median trimmed    mad   min    max  range  skew
## mpg     1 32  20.09   6.03  19.20   19.70   5.41 10.40  33.90  23.50  0.61
## cyl     2 32   6.19   1.79   6.00    6.23   2.97  4.00   8.00   4.00 -0.17
## disp    3 32 230.72 123.94 196.30  222.52 140.48 71.10 472.00 400.90  0.38
## hp      4 32 146.69  68.56 123.00  141.19  77.10 52.00 335.00 283.00  0.73
## drat    5 32   3.60   0.53   3.70    3.58   0.70  2.76   4.93   2.17  0.27
## wt      6 32   3.22   0.98   3.33    3.15   0.77  1.51   5.42   3.91  0.42
## qsec    7 32  17.85   1.79  17.71   17.83   1.42 14.50  22.90   8.40  0.37
## vs      8 32   0.44   0.50   0.00    0.42   0.00  0.00   1.00   1.00  0.24
## am      9 32   0.41   0.50   0.00    0.38   0.00  0.00   1.00   1.00  0.36
## gear   10 32   3.69   0.74   4.00    3.62   1.48  3.00   5.00   2.00  0.53
```

```
## carb   11 32   2.81   1.62   2.00    2.65   1.48  1.00   8.00   7.00  1.05
##      kurtosis    se
## mpg      -0.37  1.07
## cyl      -1.76  0.32
## disp     -1.21 21.91
## hp       -0.14 12.12
## drat     -0.71  0.09
## wt       -0.02  0.17
## qsec      0.34  0.32
## vs       -2.00  0.09
## am       -1.92  0.09
## gear     -1.07  0.13
## carb      1.26  0.29
```

psych package is also used for generating summary statistics by *grouping* variables, for example:

```
library(psych)
psych::describeBy(mydata,group = 'vs')
```

```
##
##  Descriptive statistics by group
## vs: 0
##      vars  n   mean     sd median trimmed   mad    min    max  range  skew
## mpg     1 18  16.62   3.86  15.65   16.42  2.97  10.40  26.00  15.60  0.48
## cyl     2 18   7.44   1.15   8.00    7.62  0.00   4.00   8.00   4.00 -1.74
## disp    3 18 307.15 106.77 311.00  308.52 72.65 120.30 472.00 351.70 -0.26
## hp      4 18 189.72  60.28 180.00  186.81 48.18  91.00 335.00 244.00  0.45
## drat    5 18   3.39   0.47   3.18    3.37  0.32   2.76   4.43   1.67  0.74
## wt      6 18   3.69   0.90   3.57    3.68  0.50   2.14   5.42   3.28  0.54
## qsec    7 18  16.69   1.09  17.02   16.75  0.85  14.50  18.00   3.50 -0.71
## vs      8 18   0.00   0.00   0.00    0.00  0.00   0.00   0.00   0.00   NaN
## am      9 18   0.33   0.49   0.00    0.31  0.00   0.00   1.00   1.00  0.65
## gear   10 18   3.56   0.86   3.00    3.50  0.00   3.00   5.00   2.00  0.90
## carb   11 18   3.61   1.54   4.00    3.44  1.48   2.00   8.00   6.00  1.17
##      kurtosis    se
## mpg      -0.05  0.91
## cyl       1.94  0.27
## disp     -1.06 25.16
## hp       -0.15 14.21
## drat     -0.73  0.11
## wt       -0.43  0.21
## qsec     -0.80  0.26
## vs         NaN  0.00
## am       -1.66  0.11
## gear     -1.07  0.20
## carb      1.33  0.36
## ------------------------------------------------------------
## vs: 1
##      vars  n   mean    sd median trimmed   mad   min    max  range  skew
## mpg     1 14  24.56  5.38  22.80   24.34  6.00 17.80  33.90  16.10  0.41
## cyl     2 14   4.57  0.94   4.00    4.50  0.00  4.00   6.00   2.00  0.85
## disp    3 14 132.46 56.89 120.55  127.11 61.82 71.10 258.00 186.90  0.80
```

```
## hp      4 14  91.36 24.42  96.00   92.00 32.62 52.00 123.00  71.00 -0.24
## drat    5 14   3.86  0.51   3.92    3.86  0.26  2.76   4.93   2.17 -0.28
## wt      6 14   2.61  0.72   2.62    2.63  0.95  1.51   3.46   1.95 -0.17
## qsec    7 14  19.33  1.35  19.17   19.24  1.02 16.90  22.90   6.00  0.86
## vs      8 14   1.00  0.00   1.00    1.00  0.00  1.00   1.00   0.00   NaN
## am      9 14   0.50  0.52   0.50    0.50  0.74  0.00   1.00   1.00  0.00
## gear   10 14   3.86  0.53   4.00    3.83  0.00  3.00   5.00   2.00 -0.17
## carb   11 14   1.79  1.05   1.50    1.67  0.74  1.00   4.00   3.00  1.13
##      kurtosis    se
## mpg     -1.40  1.44
## cyl     -1.36  0.25
## disp    -0.49 15.21
## hp      -1.61  6.53
## drat     0.46  0.14
## wt      -1.68  0.19
## qsec     1.25  0.36
## vs        NaN  0.00
## am      -2.14  0.14
## gear    -0.09  0.14
## carb    -0.03  0.28
```

(3) table() & prop.table()

- table() function can be used to quickly create frequency tables.

Example 1:Frequency Table for One Variable

```
table(mydata$mpg)
```

```
##
## 10.4 13.3 14.3 14.7   15 15.2 15.5 15.8 16.4 17.3 17.8 18.1 18.7 19.2 19.7   21
##    2    1    1    1    1    2    1    1    1    1    1    1    1    2    1    2
## 21.4 21.5 22.8 24.4   26 27.3 30.4 32.4 33.9
##    2    1    2    1    1    1    2    1    1
```

Example 2: Frequency Table of Proportions for One Variable

```
prop.table(mydata$mpg)
```

```
##  [1] 0.03266449 0.03266449 0.03546430 0.03328667 0.02908695 0.02815368
##  [7] 0.02224296 0.03795303 0.03546430 0.02986468 0.02768704 0.02550941
## [13] 0.02690932 0.02364287 0.01617670 0.01617670 0.02286514 0.05039664
## [19] 0.04728574 0.05272982 0.03344221 0.02410950 0.02364287 0.02068751
## [25] 0.02986468 0.04246384 0.04044175 0.04728574 0.02457614 0.03064240
## [31] 0.02333178 0.03328667
```

Example 3: Frequency Table for 2 Variables

```
table(mydata$mpg,mydata$cyl)
```

```
##
##         4 6 8
##   10.4 0 0 2
##   13.3 0 0 1
##   14.3 0 0 1
##   14.7 0 0 1
##   15   0 0 1
##   15.2 0 0 2
##   15.5 0 0 1
##   15.8 0 0 1
##   16.4 0 0 1
##   17.3 0 0 1
##   17.8 0 1 0
##   18.1 0 1 0
##   18.7 0 0 1
##   19.2 0 1 1
##   19.7 0 1 0
##   21   0 2 0
##   21.4 1 1 0
##   21.5 1 0 0
##   22.8 2 0 0
##   24.4 1 0 0
##   26   1 0 0
##   27.3 1 0 0
##   30.4 2 0 0
##   32.4 1 0 0
##   33.9 1 0 0
```

Example 4: Frequency Table of Proportions for Two Variables & only display two decimal places

```
options(digits=2)
prop.table(table(mydata$mpg,mydata$cyl))
```

```
##
##             4     6     8
##   10.4 0.000 0.000 0.062
##   13.3 0.000 0.000 0.031
##   14.3 0.000 0.000 0.031
##   14.7 0.000 0.000 0.031
##   15   0.000 0.000 0.031
##   15.2 0.000 0.000 0.062
##   15.5 0.000 0.000 0.031
##   15.8 0.000 0.000 0.031
##   16.4 0.000 0.000 0.031
##   17.3 0.000 0.000 0.031
```

```
##   17.8 0.000 0.031 0.000
##   18.1 0.000 0.031 0.000
##   18.7 0.000 0.000 0.031
##   19.2 0.000 0.031 0.031
##   19.7 0.000 0.031 0.000
##   21   0.000 0.062 0.000
##   21.4 0.031 0.031 0.000
##   21.5 0.031 0.000 0.000
##   22.8 0.062 0.000 0.000
##   24.4 0.031 0.000 0.000
##   26   0.031 0.000 0.000
##   27.3 0.031 0.000 0.000
##   30.4 0.062 0.000 0.000
##   32.4 0.031 0.000 0.000
##   33.9 0.031 0.000 0.000
```

(4) aggregate()

- This example aggregates data frame mtcars by cyl and vs, returning means (FUN can be customized to return other statistical variables like sd, sum etc).
- FUN: a function to compute the summary statistics which can be applied to all data subsets.

```
### for numeric variables
attach(mydata)
aggdata <-aggregate(mydata, by=list(cyl,vs),
  FUN=mean, na.rm=TRUE)
print(aggdata)
```

```
##   Group.1 Group.2 mpg cyl disp  hp drat  wt qsec vs   am gear carb
## 1       4       0  26   4  120  91  4.4 2.1   17  0 1.00  5.0  2.0
## 2       6       0  21   6  155 132  3.8 2.8   16  0 1.00  4.3  4.7
## 3       8       0  15   8  353 209  3.2 4.0   17  0 0.14  3.3  3.5
## 4       4       1  27   4  104  82  4.0 2.3   19  1 0.70  4.0  1.5
## 5       6       1  19   6  205 115  3.4 3.4   19  1 0.00  3.5  2.5
```

```
detach(mydata)
```

2. T tests (by Longfei)

(1) One-Sample t-test

```
# Prerequisites
library(datarium)
packages <- c('tidyverse','ggpubr','rstatix','datarium')
#install.packages(packages)
lapply(packages, library, character.only = TRUE)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::%+%()   masks psych::%+%()
## x ggplot2::alpha() masks psych::alpha()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()

##
## Attaching package: 'rstatix'

## The following object is masked from 'package:stats':
##
##     filter

## [[1]]
##  [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
##  [7] "tibble"    "ggplot2"   "tidyverse" "datarium"  "psych"     "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "ggpubr"    "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "datarium"  "psych"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[3]]
##  [1] "rstatix"   "ggpubr"    "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "datarium"
## [13] "psych"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
##
## [[4]]
##  [1] "rstatix"   "ggpubr"    "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "datarium"
## [13] "psych"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
```

```
# Demo data
# Load and inspect the data
data(mice, package = "datarium")
head(mice, 3)
```

```
## # A tibble: 3 x 2
##   name  weight
##   <chr>  <dbl>
## 1 M_1     18.9
## 2 M_2     19.5
## 3 M_3     23.1
```

```r
# Summary statistics
mice %>% get_summary_stats(weight, type = "mean_sd")
```

```
## # A tibble: 1 x 4
##   variable     n  mean    sd
##   <chr>    <dbl> <dbl> <dbl>
## 1 weight      10  20.1  1.90
```

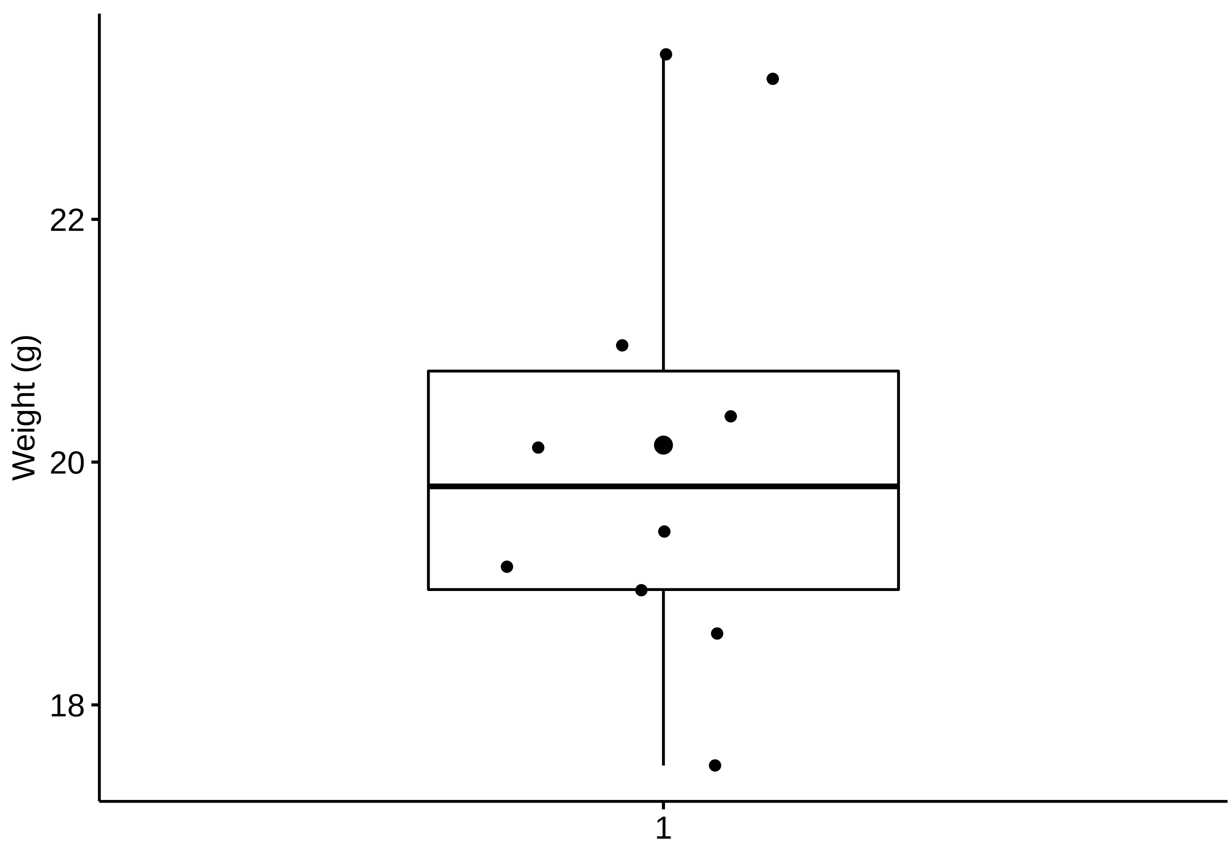
```r
# Visualization
bxp <- ggboxplot(
  mice$weight, width = 0.5, add = c("mean", "jitter"),
  ylab = "Weight (g)", xlab = FALSE
)
```

```
## Warning: ‘fun.y‘ is deprecated. Use ‘fun‘ instead.

## Warning: ‘fun.ymin‘ is deprecated. Use ‘fun.min‘ instead.

## Warning: ‘fun.ymax‘ is deprecated. Use ‘fun.max‘ instead.
```

```r
bxp
```



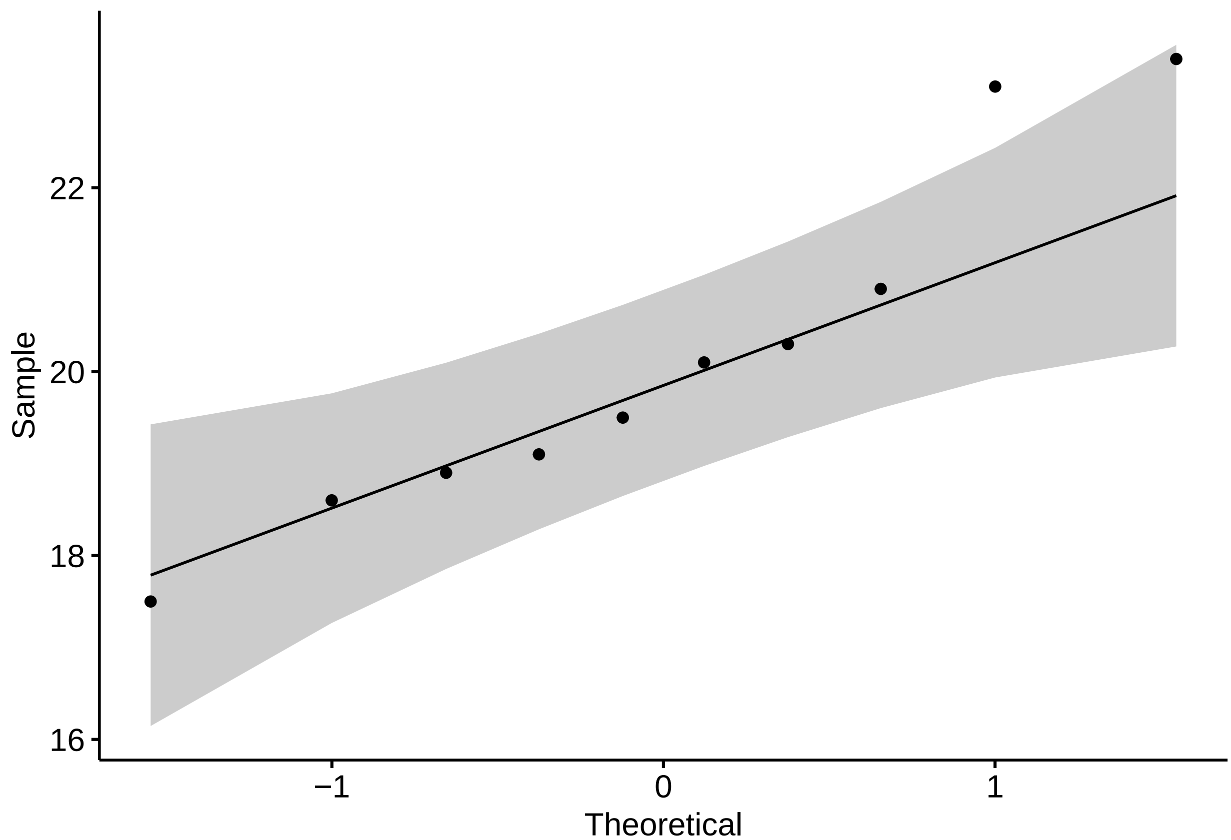
```r
# Identify outliers
mice %>% identify_outliers(weight)
```

```
## [1] name       weight     is.outlier is.extreme
## <0 rows> (or 0-length row.names)
```

```r
# Check normality assumption
mice %>% shapiro_test(weight)
```

```
## # A tibble: 1 x 3
##   variable statistic     p
##   <chr>        <dbl> <dbl>
## 1 weight       0.923 0.382
```

```r
ggqqplot(mice, x = "weight")
```



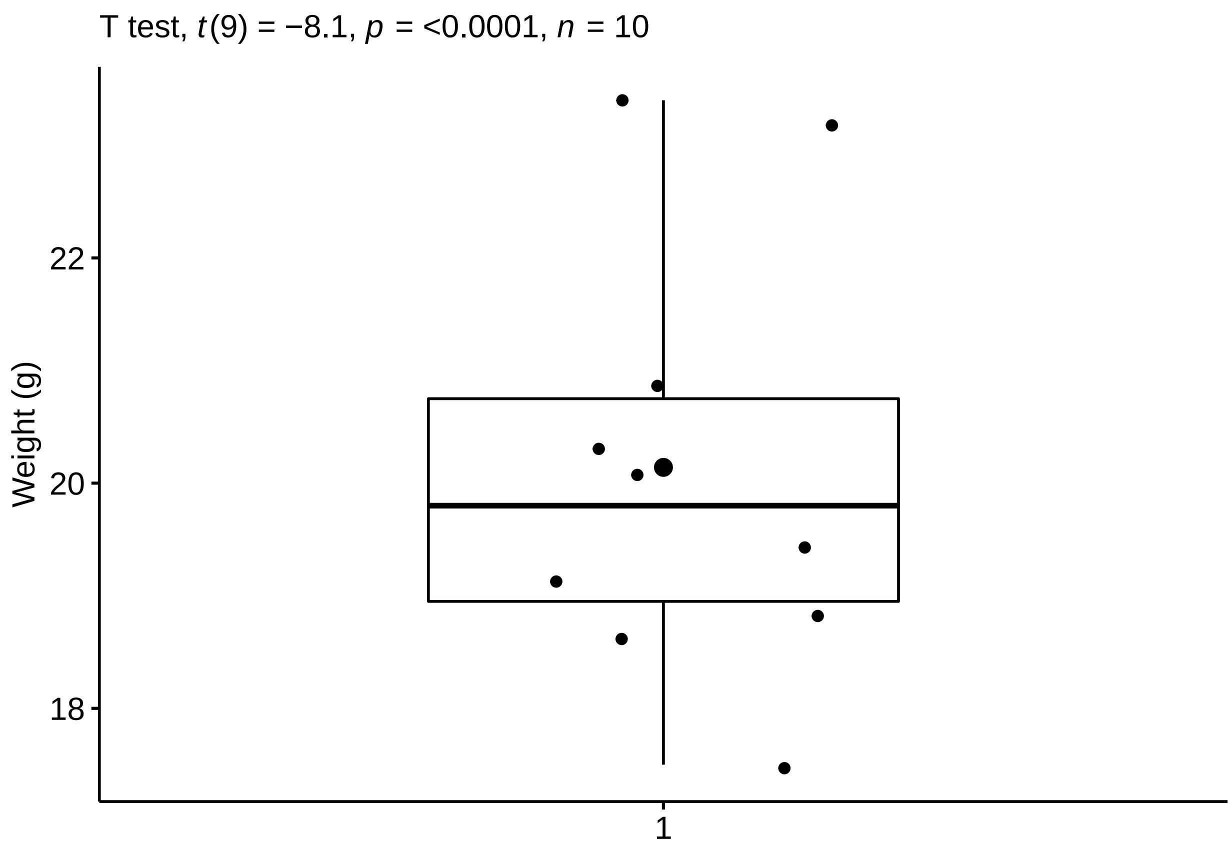

```r
# Computation
stat.test <- mice %>% t_test(weight ~ 1, mu = 25)
stat.test
```

```
## # A tibble: 1 x 7
##   .y.    group1 group2         n statistic    df        p
## * <chr>  <chr>  <chr>      <int>     <dbl> <dbl>    <dbl>
## 1 weight 1      null model    10     -8.10     9 0.00002
```

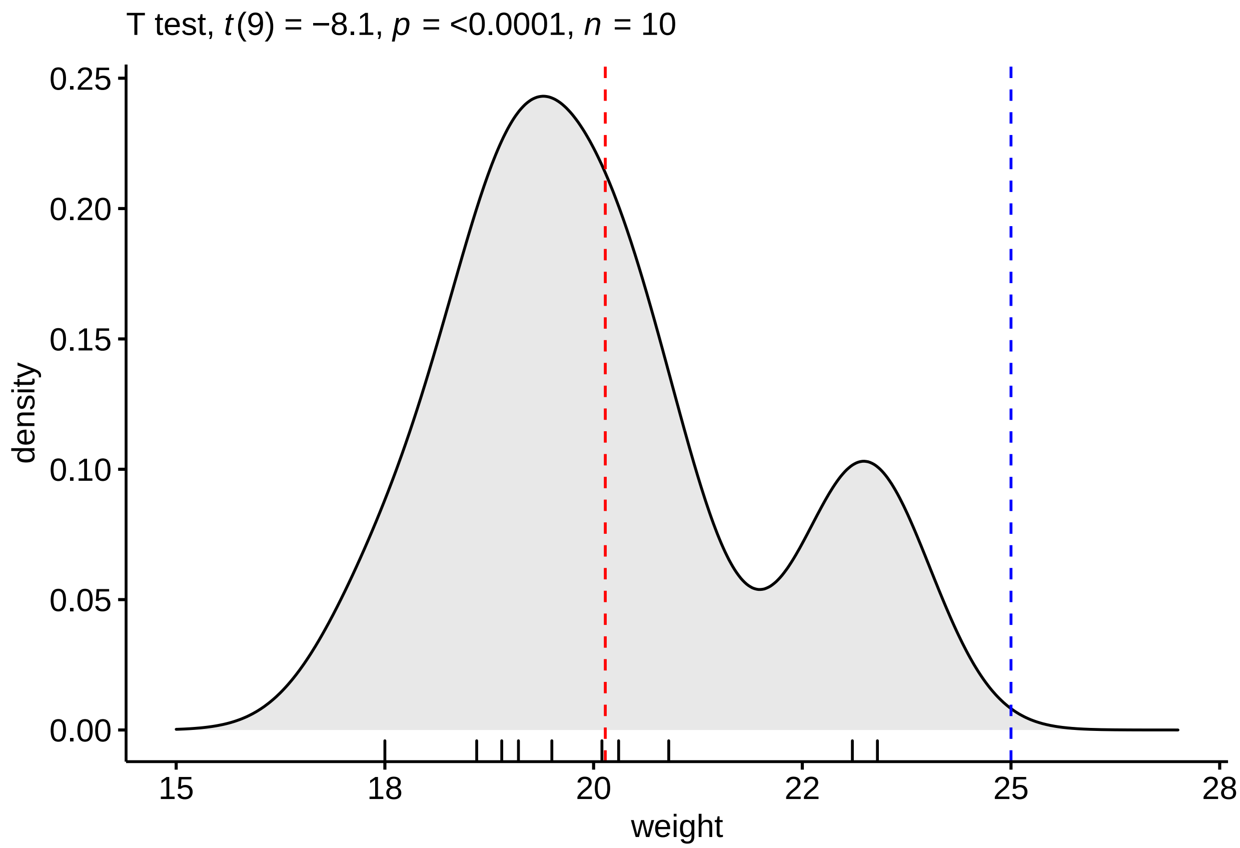
```r
# Effect size
mice %>% cohens_d(weight ~ 1, mu = 25)
```

```
## # A tibble: 1 x 6
##   .y.    group1 group2     effsize     n magnitude
## * <chr>  <chr>  <chr>        <dbl> <int> <ord>
## 1 weight 1      null model   -2.56    10 large
```

```r
# Report
bxp + labs(
  subtitle = get_test_label(stat.test, detailed = TRUE)
)
```




```r
ggdensity(mice, x = "weight", rug = TRUE, fill = "lightgray") +
  scale_x_continuous(limits = c(15, 27)) +
  stat_central_tendency(type = "mean", color = "red", linetype = "dashed") +
  geom_vline(xintercept = 25, color = "blue", linetype = "dashed") +
  labs(subtitle = get_test_label(stat.test,  detailed = TRUE))
```



(2) Independent samples t-test

```
# Demo data
# Load the data
data("genderweight", package = "datarium")
# Show a sample of the data by group
set.seed(123)
genderweight %>% sample_n_by(group, size = 2)
```

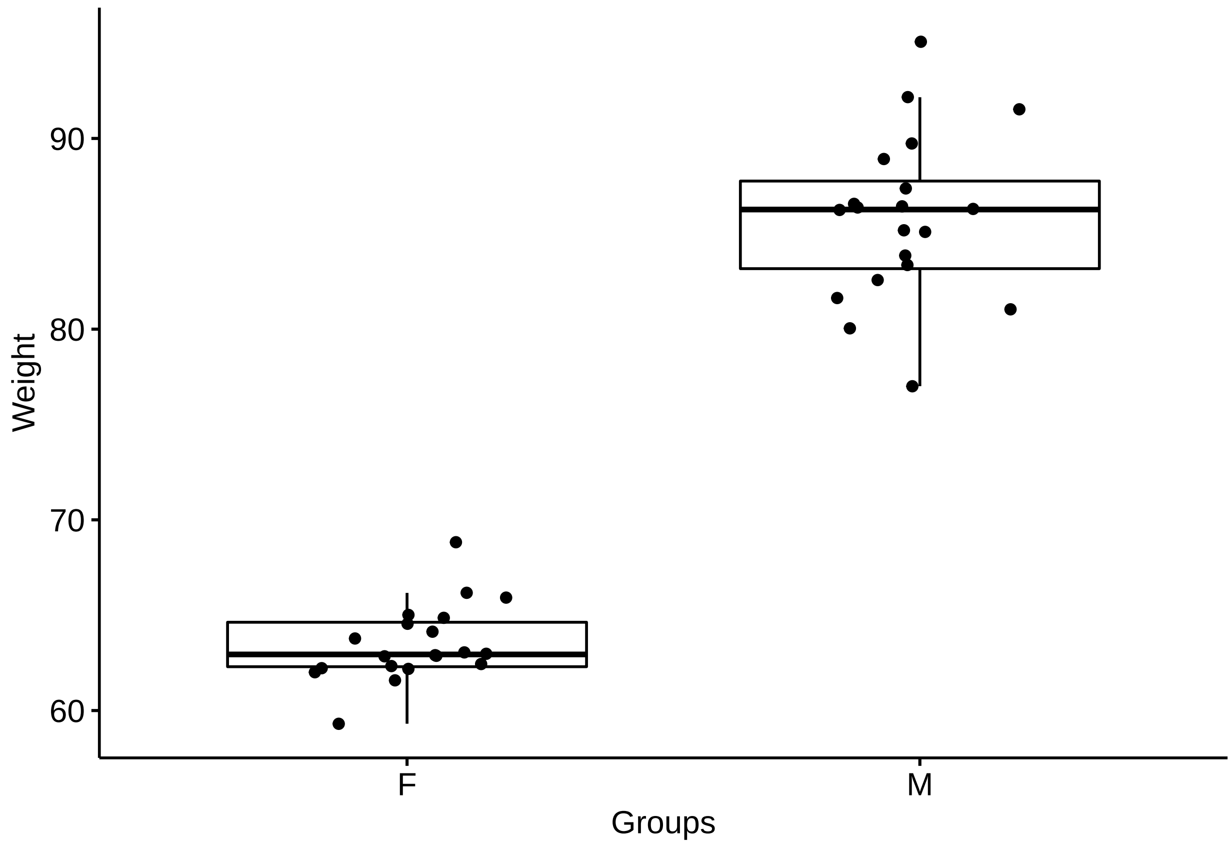
```
## # A tibble: 4 x 3
##   id    group weight
##   <fct> <fct>  <dbl>
## 1 15    F       65.9
## 2 19    F       62.3
## 3 34    M       86.2
## 4 23    M       86.6
```

```
#Summary statistics
genderweight %>%
  group_by(group) %>%
  get_summary_stats(weight, type = "mean_sd")
```

```
## # A tibble: 2 x 5
```

```
##   group variable     n  mean    sd
##   <fct> <chr>    <dbl> <dbl> <dbl>
## 1 F     weight      20  63.5  2.03
## 2 M     weight      20  85.8  4.35
```

```r
#Visualization
bxp <- ggboxplot(
  genderweight, x = "group", y = "weight",
  ylab = "Weight", xlab = "Groups", add = "jitter"
)
bxp
```



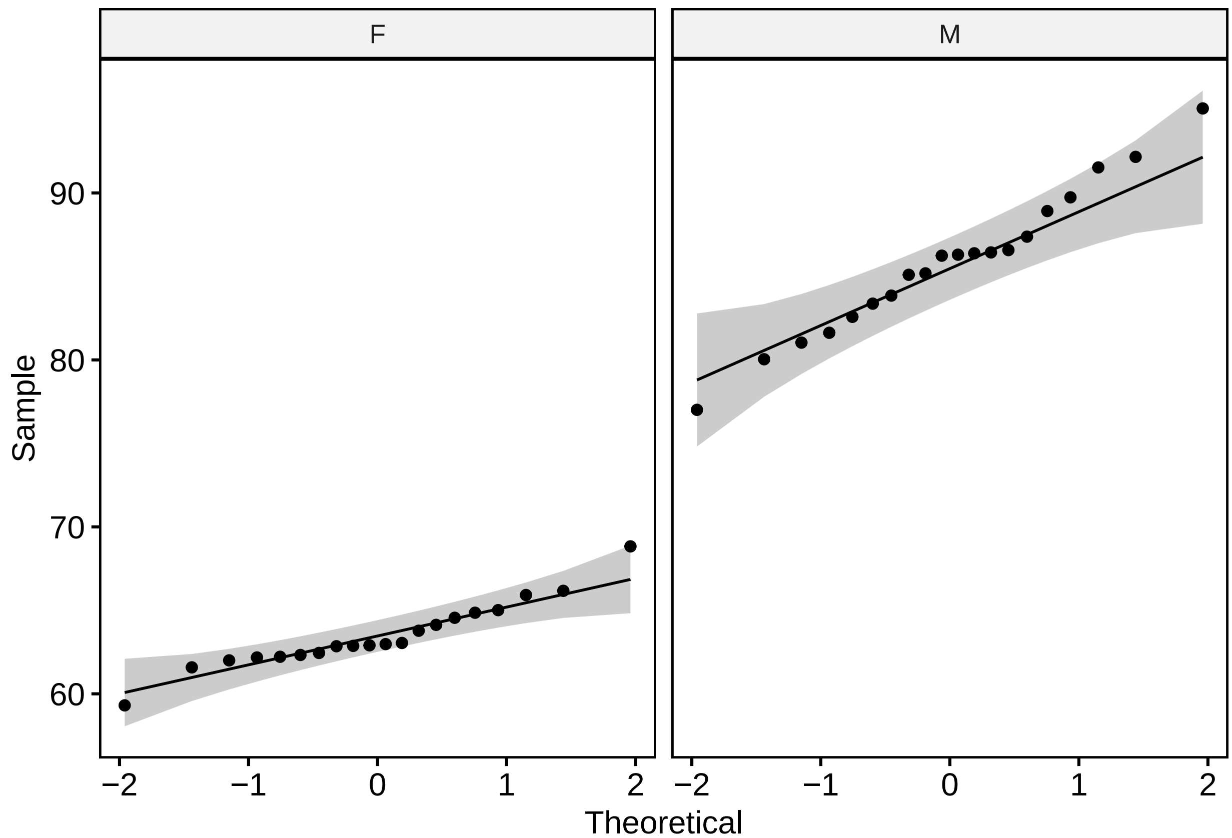

```r
# Identify outliers by groups
genderweight %>%
  group_by(group) %>%
  identify_outliers(weight)
```

```
## # A tibble: 2 x 5
##   group id    weight is.outlier is.extreme
##   <fct> <fct>  <dbl> <lgl>      <lgl>
## 1 F     20      68.8 TRUE       FALSE
## 2 M     31      95.1 TRUE       FALSE
```

```r
# Check normality by groups
# Compute Shapiro wilk test by goups
data(genderweight, package = "datarium")
genderweight %>%
  group_by(group) %>%
  shapiro_test(weight)
```

```
## # A tibble: 2 x 4
##   group variable statistic     p
##   <fct> <chr>        <dbl> <dbl>
## 1 F     weight       0.938 0.224
## 2 M     weight       0.986 0.989
```

```r
# Draw a qq plot by group
ggqqplot(genderweight, x = "weight", facet.by = "group")
```




```r
# Check the equality of variances
genderweight %>% levene_test(weight ~ group)
```

```
## # A tibble: 1 x 4
##     df1   df2 statistic      p
##   <int> <int>     <dbl>  <dbl>
## 1     1    38      6.12 0.0180
```

```r
# Computation
stat.test <- genderweight %>%
  t_test(weight ~ group) %>%
  add_significance()
stat.test
```

```
## # A tibble: 1 x 9
##   .y.    group1 group2    n1    n2 statistic    df        p p.signif
##   <chr>  <chr>  <chr>  <int> <int>     <dbl> <dbl>    <dbl> <chr>
## 1 weight F      M         20    20     -20.8  26.9 4.3e-18 ****
```

```r
stat.test2 <- genderweight %>%
  t_test(weight ~ group, var.equal = TRUE) %>%
  add_significance()
stat.test2
```

```
## # A tibble: 1 x 9
##   .y.    group1 group2    n1    n2 statistic    df         p p.signif
##   <chr>  <chr>  <chr>  <int> <int>     <dbl> <dbl>     <dbl> <chr>
## 1 weight F      M         20    20     -20.8    38 2.33e-22 ****
```

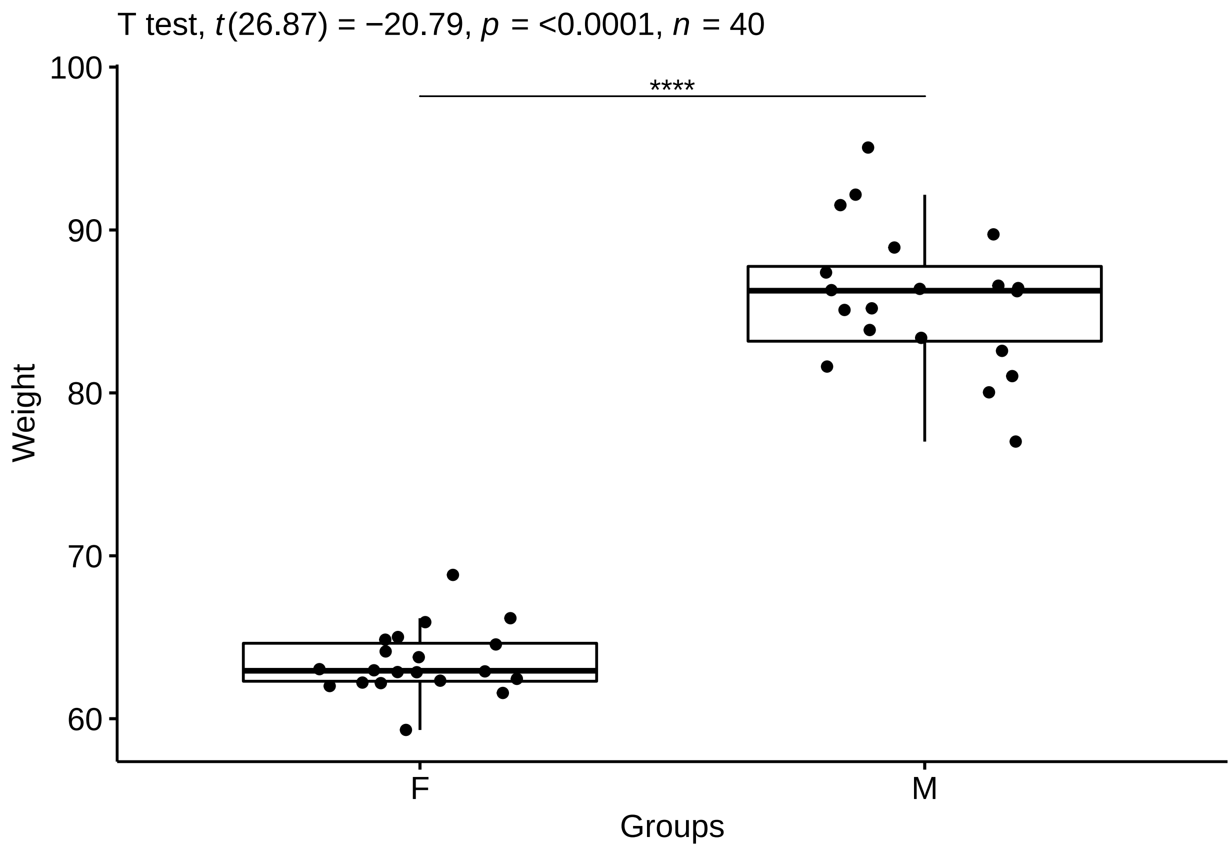
```r
# Effect size
# Cohen's d for Student t-test
genderweight %>%  cohens_d(weight ~ group, var.equal = TRUE)
```

```
## # A tibble: 1 x 7
##   .y.    group1 group2 effsize    n1    n2 magnitude
## * <chr>  <chr>  <chr>    <dbl> <int> <int> <ord>
## 1 weight F      M        -6.57    20    20 large
```

```r
# Cohen's d for Welch t-test
genderweight %>% cohens_d(weight ~ group, var.equal = FALSE)
```

```
## # A tibble: 1 x 7
##   .y.    group1 group2 effsize    n1    n2 magnitude
## * <chr>  <chr>  <chr>    <dbl> <int> <int> <ord>
## 1 weight F      M        -6.57    20    20 large
```

```r
# Report
stat.test <- stat.test %>% add_xy_position(x = "group")
bxp +
  stat_pvalue_manual(stat.test, tip.length = 0) +
  labs(subtitle = get_test_label(stat.test, detailed = TRUE))
```



(3) Paired samples t-test

```
# Demo data
# Wide format
data("mice2", package = "datarium")
head(mice2, 3)
```

```
##   id before after
## 1  1    187   430
## 2  2    194   404
## 3  3    232   406
```

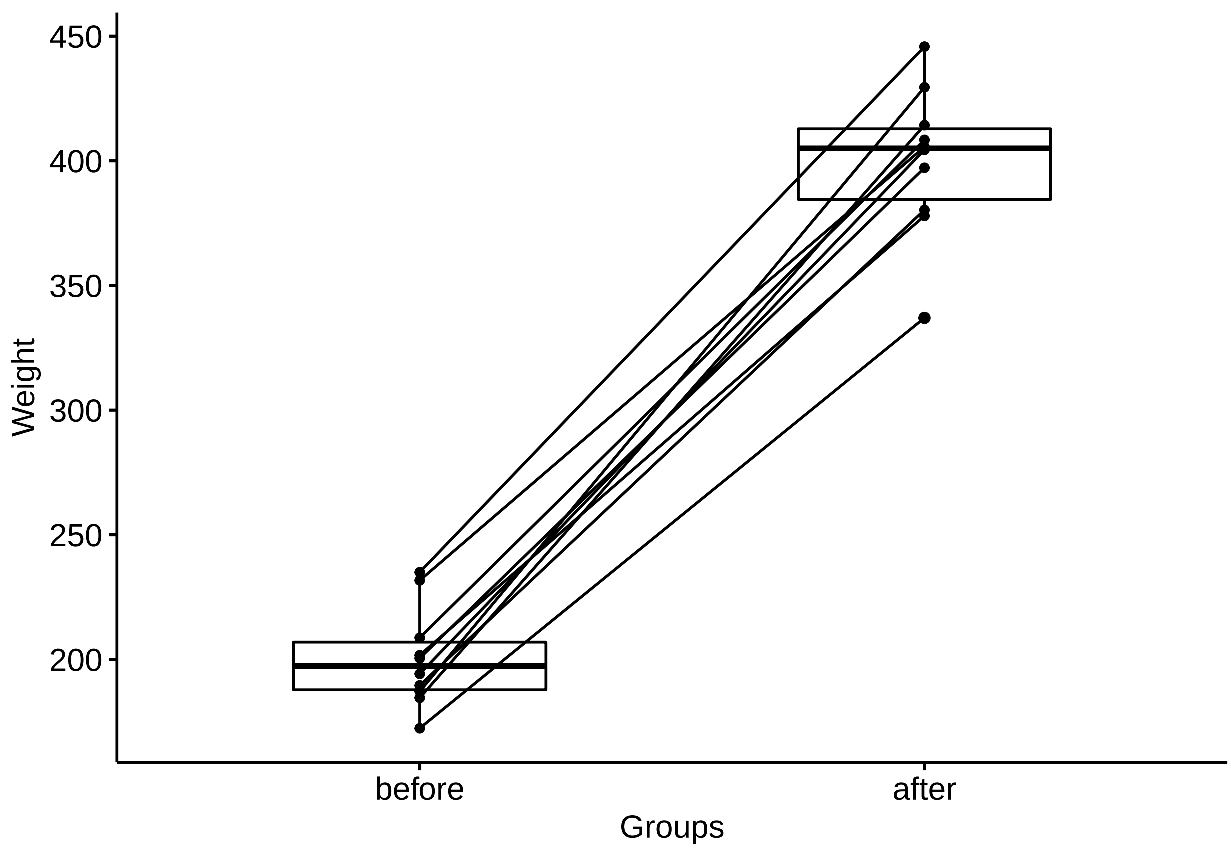
```
# Transform into long data:
# gather the before and after values in the same column
mice2.long <- mice2 %>%
  gather(key = "group", value = "weight", before, after)
head(mice2.long, 3)
```

```
##   id  group weight
## 1  1 before    187
## 2  2 before    194
## 3  3 before    232
```

```r
# Summary statistics
mice2.long %>%
  group_by(group) %>%
  get_summary_stats(weight, type = "mean_sd")
```

```
## # A tibble: 2 x 5
##   group  variable     n  mean    sd
##   <chr>  <chr>    <dbl> <dbl> <dbl>
## 1 after  weight      10  400.  30.1
## 2 before weight      10  201.  20.0
```

```r
# Visualization
bxp <- ggpaired(mice2.long, x = "group", y = "weight",
                order = c("before", "after"),
                ylab = "Weight", xlab = "Groups")
bxp
```




```r
# Assumptions and preliminary tests
mice2 <- mice2 %>% mutate(differences = before - after)
head(mice2, 3)
```

```
##   id before after differences
## 1  1    187   430        -242
## 2  2    194   404        -210
## 3  3    232   406        -174
```

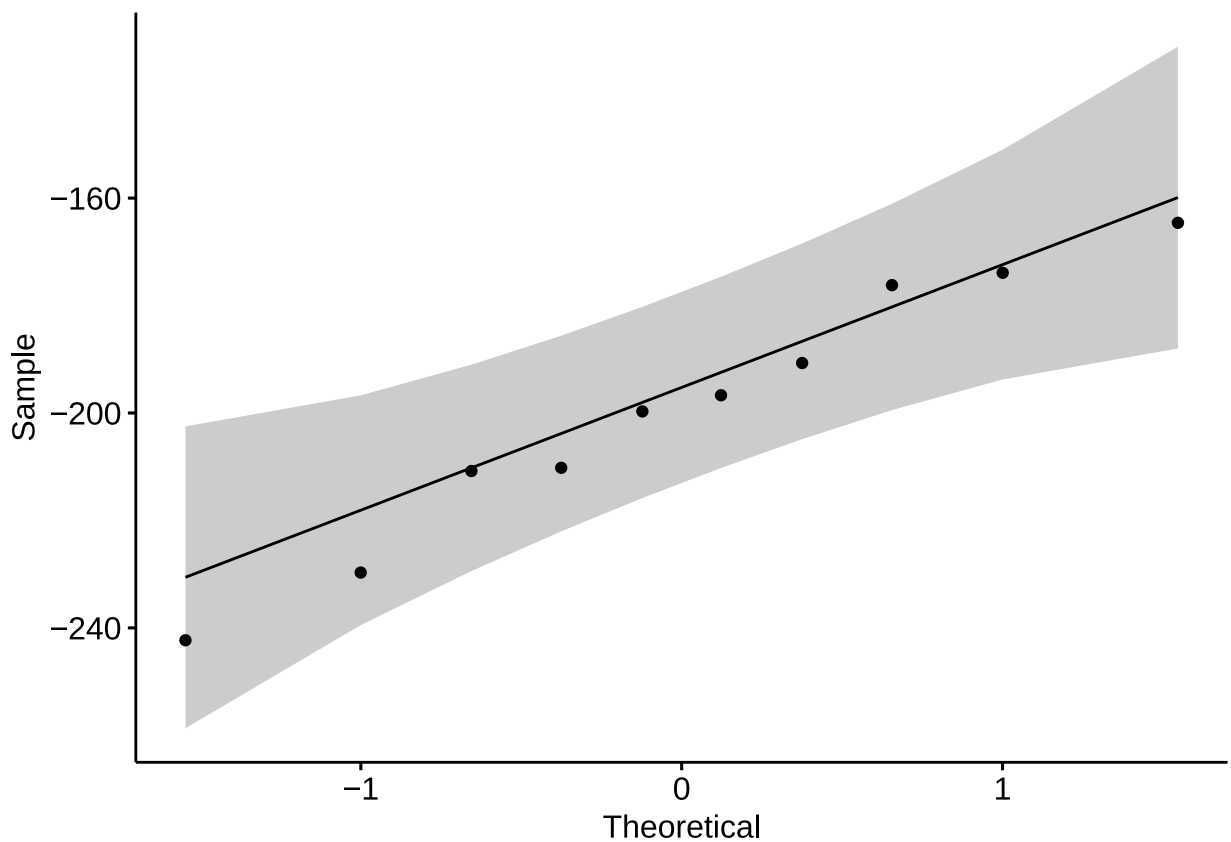
```r
# Identify outliers
mice2 %>% identify_outliers(differences)
```

```
## [1] id          before      after       differences is.outlier  is.extreme
## <0 rows> (or 0-length row.names)
```

```r
# Check normality assumption
# Shapiro-Wilk normality test for the differences
mice2 %>% shapiro_test(differences)
```

```
## # A tibble: 1 x 3
##   variable    statistic     p
##   <chr>           <dbl> <dbl>
## 1 differences     0.968 0.867
```

```r
# QQ plot for the difference
ggqqplot(mice2, "differences")
```



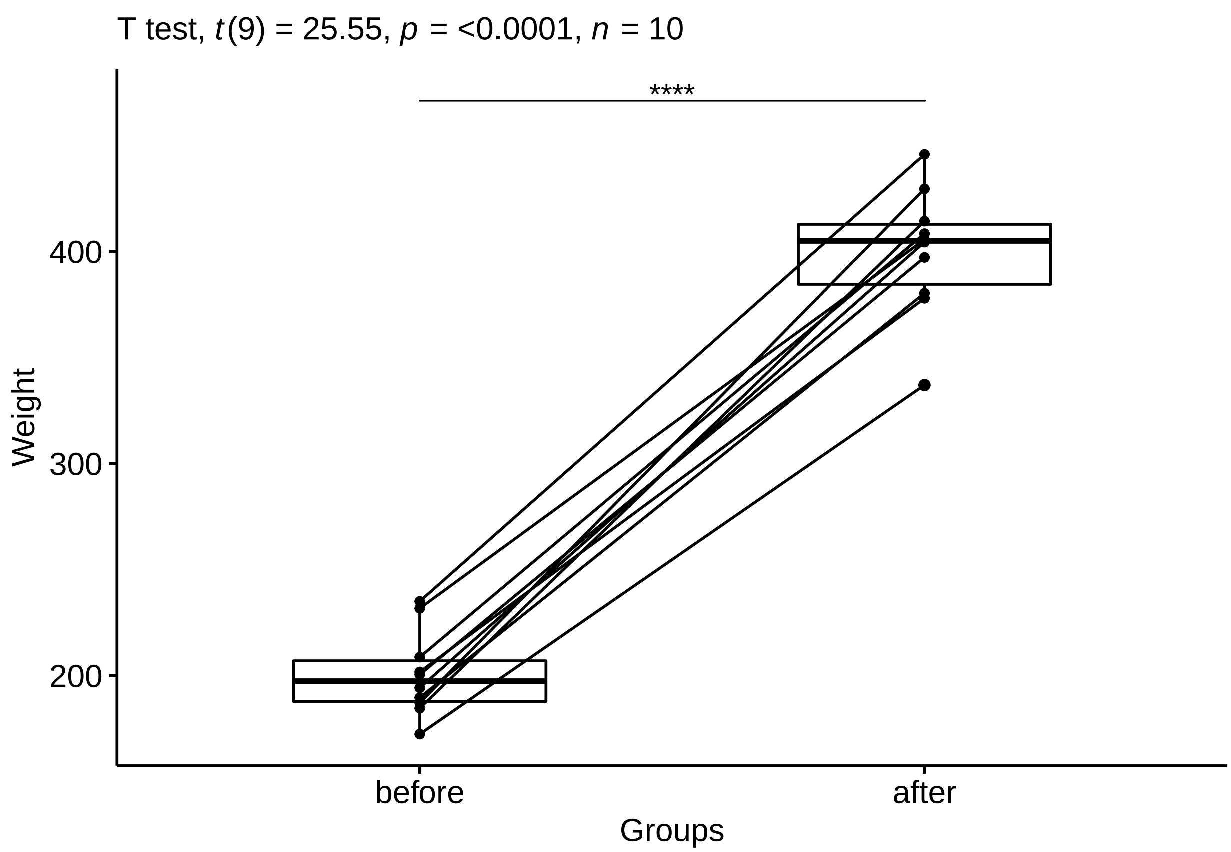

```r
# Computation
stat.test <- mice2.long  %>%
  t_test(weight ~ group, paired = TRUE) %>%
  add_significance()
stat.test
```

```
## # A tibble: 1 x 9
##   .y.    group1 group2    n1    n2 statistic    df            p p.signif
##   <chr>  <chr>  <chr>  <int> <int>     <dbl> <dbl>        <dbl> <chr>
## 1 weight after  before    10    10      25.5     9 0.00000000104 ****
```

```
# Effect size
mice2.long  %>% cohens_d(weight ~ group, paired = TRUE)
```

```
## # A tibble: 1 x 7
##   .y.    group1 group2 effsize    n1    n2 magnitude
## * <chr>  <chr>  <chr>    <dbl> <int> <int> <ord>
## 1 weight after  before    8.08    10    10 large
```

```
# Report
stat.test <- stat.test %>% add_xy_position(x = "group")
bxp +
  stat_pvalue_manual(stat.test, tip.length = 0) +
  labs(subtitle = get_test_label(stat.test, detailed= TRUE))
```




```
# Summary
# One-sample t-test
mice %>% t_test(weight ~ 1, mu = 25)
```

```
## # A tibble: 1 x 7
```

```
##   .y.    group1 group2         n statistic    df        p
## * <chr>  <chr>  <chr>      <int>     <dbl> <dbl>    <dbl>
## 1 weight 1      null model    10     -8.10     9 0.00002
```

```
# Independent samples t-test
genderweight %>% t_test(weight ~ group)
```

```
## # A tibble: 1 x 8
##   .y.    group1 group2    n1    n2 statistic    df        p
## * <chr>  <chr>  <chr>  <int> <int>     <dbl> <dbl>    <dbl>
## 1 weight F      M         20    20     -20.8  26.9 4.3e-18
```

```
# Paired sample t-test
mice2.long %>% t_test(weight ~ group, paired = TRUE)
```

```
## # A tibble: 1 x 8
##   .y.    group1 group2    n1    n2 statistic    df             p
## * <chr>  <chr>  <chr>  <int> <int>     <dbl> <dbl>         <dbl>
## 1 weight after  before    10    10      25.5     9 0.00000000104
```

3. Correlation & Regression analysis (by Liangjun)

```
## load data
mydata<-mtcars
mydata
```

```
##                      mpg cyl disp  hp drat  wt qsec vs am gear carb
## Mazda RX4             21   6  160 110  3.9 2.6   16  0  1    4    4
## Mazda RX4 Wag         21   6  160 110  3.9 2.9   17  0  1    4    4
## Datsun 710            23   4  108  93  3.9 2.3   19  1  1    4    1
## Hornet 4 Drive        21   6  258 110  3.1 3.2   19  1  0    3    1
## Hornet Sportabout     19   8  360 175  3.1 3.4   17  0  0    3    2
## Valiant               18   6  225 105  2.8 3.5   20  1  0    3    1
## Duster 360            14   8  360 245  3.2 3.6   16  0  0    3    4
## Merc 240D             24   4  147  62  3.7 3.2   20  1  0    4    2
## Merc 230              23   4  141  95  3.9 3.1   23  1  0    4    2
## Merc 280              19   6  168 123  3.9 3.4   18  1  0    4    4
## Merc 280C             18   6  168 123  3.9 3.4   19  1  0    4    4
## Merc 450SE            16   8  276 180  3.1 4.1   17  0  0    3    3
## Merc 450SL            17   8  276 180  3.1 3.7   18  0  0    3    3
## Merc 450SLC           15   8  276 180  3.1 3.8   18  0  0    3    3
## Cadillac Fleetwood    10   8  472 205  2.9 5.2   18  0  0    3    4
## Lincoln Continental   10   8  460 215  3.0 5.4   18  0  0    3    4
## Chrysler Imperial     15   8  440 230  3.2 5.3   17  0  0    3    4
## Fiat 128              32   4   79  66  4.1 2.2   19  1  1    4    1
## Honda Civic           30   4   76  52  4.9 1.6   19  1  1    4    2
## Toyota Corolla        34   4   71  65  4.2 1.8   20  1  1    4    1
## Toyota Corona         22   4  120  97  3.7 2.5   20  1  0    3    1
## Dodge Challenger      16   8  318 150  2.8 3.5   17  0  0    3    2
```

```
## AMC Javelin          15   8  304 150  3.1 3.4   17  0  0    3    2
## Camaro Z28           13   8  350 245  3.7 3.8   15  0  0    3    4
## Pontiac Firebird     19   8  400 175  3.1 3.8   17  0  0    3    2
## Fiat X1-9            27   4   79  66  4.1 1.9   19  1  1    4    1
## Porsche 914-2        26   4  120  91  4.4 2.1   17  0  1    5    2
## Lotus Europa         30   4   95 113  3.8 1.5   17  1  1    5    2
## Ford Pantera L       16   8  351 264  4.2 3.2   14  0  1    5    4
## Ferrari Dino         20   6  145 175  3.6 2.8   16  0  1    5    6
## Maserati Bora        15   8  301 335  3.5 3.6   15  0  1    5    8
## Volvo 142E           21   4  121 109  4.1 2.8   19  1  1    4    2
```

(1) Correlation Analysis (e.g.diso vs mpg)

*cor(x,y,use="everything","complete.obs","all.obs"or"pairwise.complete.obs",method="pea options of "use="here refers to different methods it could use to deal with missing data.

this command will give you the result of "r" value but not the "p" value

```
cor(mydata$disp,mydata$mpg, use="everything", method="pearson")
```

```
## [1] -0.85
```

```
## if you need to see more details such as correlation coeficient, you can use cor.test()

cor.test(mydata$disp,mydata$mpg,method = "pearson")
```

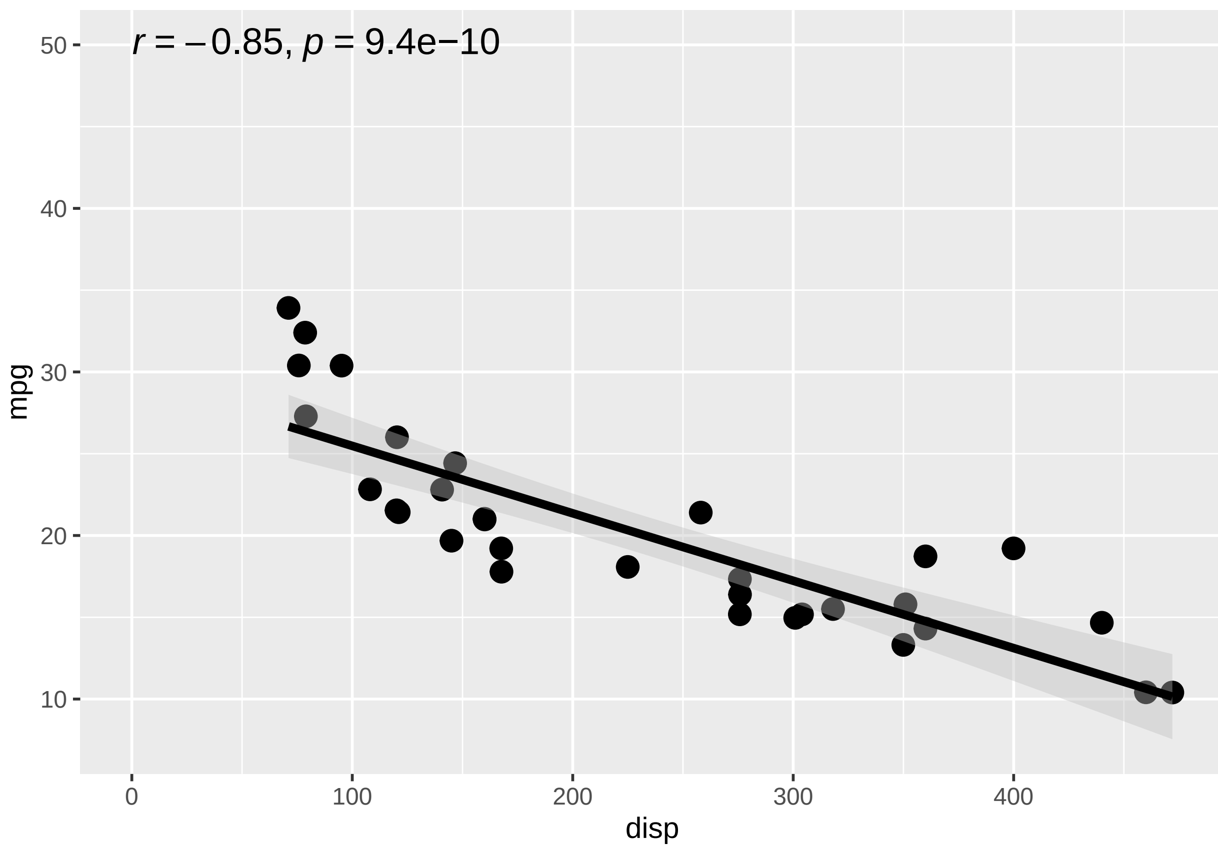
```
##
##  Pearson's product-moment correlation
##
## data:  mydata$disp and mydata$mpg
## t = -9, df = 30, p-value = 9e-10
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.92 -0.71
## sample estimates:
##   cor
## -0.85
```

```
## if you want to check the pearson correlation coefficient in a scatter plot (disp vs mpg in this examp

library("ggpubr")
f <- ggplot(mydata, aes(disp,mpg))

S <- f + geom_jitter(size=3.5)+
  geom_smooth(method="lm",color="black",fill="grey",formula=y~x,size=1.5)+
  stat_cor(method="pearson",cor.coef.name = "r",label.x=0.2,label.y=50,size=5)

S
```



##~~~~~~

(2) Partial correlation

which reduce the effects of other variants when you want to check the relationship between two variants you are interest in (disp vs cyl, regress out wt, drat, in this example)

***"ggm" package is required, command: pcor(c("a", "b", "x", "y", "z"), var(mydata)), partial corr between a and b, controlling for x, y, z**

****Example 1, calculate the correlation between "disp" and "mpg"**

```
#install.packages("ggm")
library("ggm")
mydata<-mtcars
pcor(c("disp", "mpg", "wt", "drat"), var(mydata))
```

```
## [1] -0.31
```

```
## you will only get r value from previous code, then you can use pcor.test() to do a hypothesis test b
## pcor.test (data, control variable number, sample size)
```

```r
pc<-pcor(c("disp", "mpg", "wt", "drat"), var(mydata))
pcor.test(pc,1,32)
```

```
## $tval
## [1] -1.7
##
## $df
## [1] 29
##
## $pvalue
## [1] 0.092
```

or you can use "ppcor" package: pcor.test(x,y,z, method="pearson"/"kendall"/ "spearman"),regress out 'z' to see the correlation between 'x' and 'y'

**Example 2, analize using "ppcor" packages

```r
library(ppcor)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:rstatix':
##
##     select
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
##
## Attaching package: 'ppcor'
```

```
## The following objects are masked from 'package:ggm':
##
##     pcor, pcor.test
```

```r
pcor.test(mydata$disp,mydata$mpg,mydata$wt,method = "pearson")
```

```
##   estimate p.value statistic  n gp  Method
## 1    -0.34   0.064      -1.9 32  1 pearson
```

##*Comparing two packages: "ggm" can regresse out more than one variable, but can only use "pearson" as its correlation method; "ppcor" can only regress out one variable, but can choose other method for correlation.

***Note that partial correlation related command, pcor() & pcor.test, can be performed via either "ggm" package or "ppcor" package, in different usage.It's usage depends on their loading sequence,the laster one will cover the former one. For example, you load "ppcor" after "ggm", then the commands will have to be used in "ppcor"-way, errors will be reported if you try to use it in "ggm"-way**

unless you unattach "ppcor" packages.Code to unattach a package: detach()

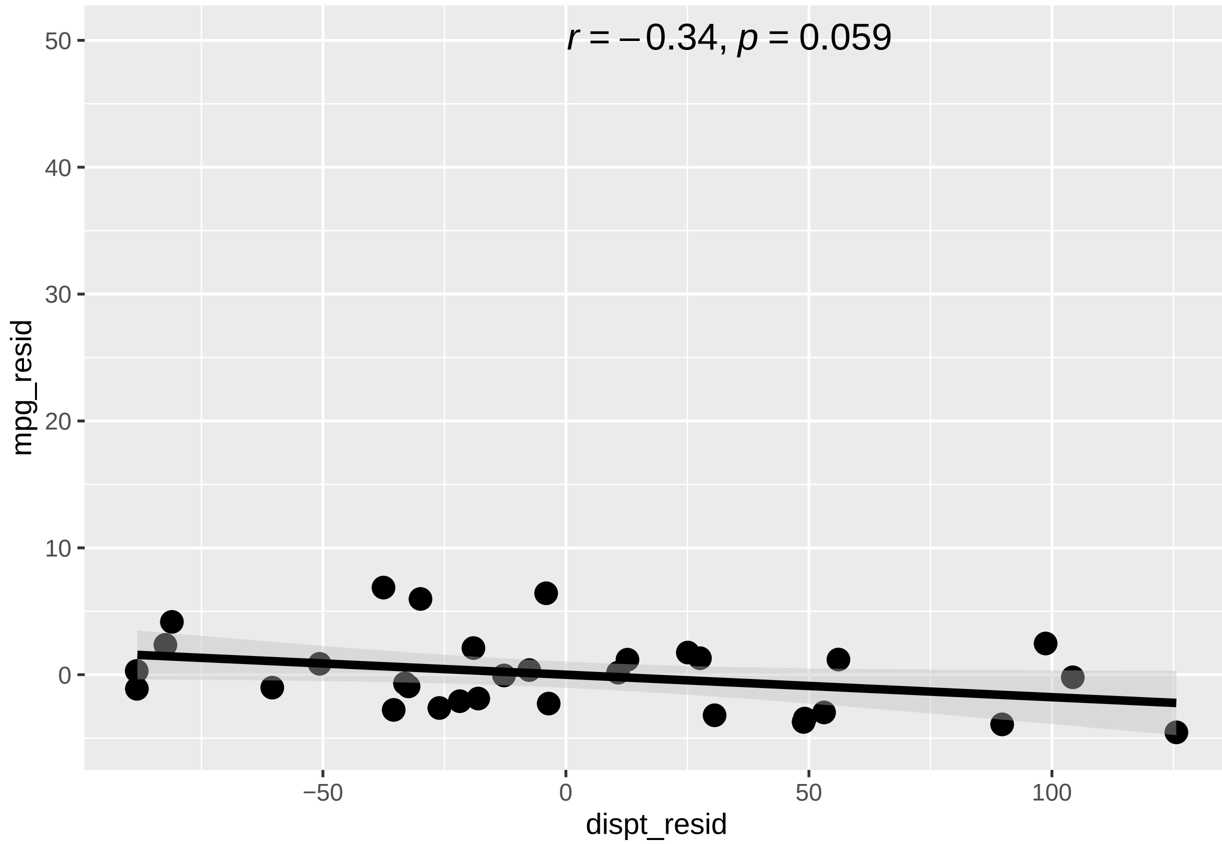
```
detach(package:ppcor)
library("ggm")
mydata<-mtcars
pcor(c("disp", "mpg", "wt", "drat"), var(mydata))
```

****Example 3: polt the partial correlation (use ggplot2 package to polt the graph):**

```
dispt_resid<-resid(lm(mydata$disp~mydata$wt))
mpg_resid<-resid(lm(mydata$mpg~mydata$wt))

F<-ggplot(mydata, aes(dispt_resid,mpg_resid))
S<-F + geom_jitter(size=3.5)+
  geom_smooth(method="lm",color="black",fill="grey",formula=y~x,size=1.5)+
    stat_cor(method="pearson",cor.coef.name = "r",label.x=0.2,label.y=50,size=5)

S
```

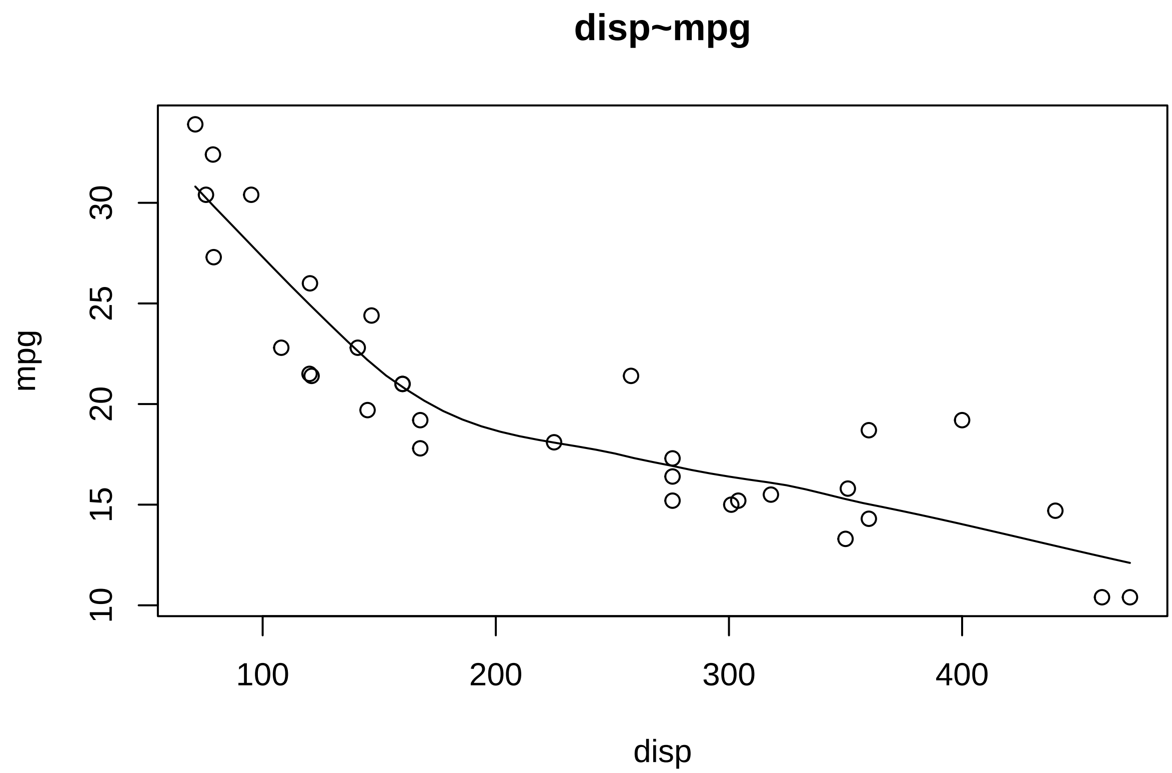


##~~~~~~

(3) Regression Analysis

****Example 1: scatter plot (e.g. disp vs mpg, assume that disp is independent variable, mpg is dependent variable)**

```
scatter.smooth(x=mtcars$disp, y=mtcars$mpg, main="disp~mpg", xlab = "disp", ylab="mpg")
```




```r
## linear regression model, lm(dpendent varable~independent variable, data= data.name)
## use lm()

Lm<-lm(mpg~disp, data = mydata)
print(Lm)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mydata)
##
## Coefficients:
## (Intercept)         disp
##     29.5999      -0.0412
```

```r
## The simple linear regression equation is: Y = b1 + b2X +e, we can use the results above to fit the e

## And you can get the summary of linear regression by using summary()
summary(Lm)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mydata)
##
## Residuals:
##    Min      1Q Median      3Q     Max
```

```
## -4.892 -2.202 -0.963  1.627  7.231
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.59985    1.22972   24.07  < 2e-16 ***
## disp        -0.04122    0.00471   -8.75  9.4e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.2 on 30 degrees of freedom
## Multiple R-squared:  0.718,  Adjusted R-squared:  0.709
## F-statistic: 76.5 on 1 and 30 DF,  p-value: 9.38e-10
```

##**Example 2: multiple linear regression (e.g. mpg vs disp&wt, mpg is denpendent varable, disp and wt are independent variables)

```r
multiple <- lm(mpg~disp+wt, data=mydata)
summary(multiple)
```

```
##
## Call:
## lm(formula = mpg ~ disp + wt, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.409 -2.324 -0.768  1.772  6.348
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.96055    2.16454   16.15  4.9e-16 ***
## disp        -0.01772    0.00919   -1.93   0.0636 .
## wt          -3.35083    1.16413   -2.88   0.0074 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.9 on 29 degrees of freedom
## Multiple R-squared:  0.781,  Adjusted R-squared:  0.766
## F-statistic: 51.7 on 2 and 29 DF,  p-value: 2.74e-10
```

The equation of multiple linear regression is y = c0 + c1*x1 + 2*c2*x2+3*c3*x3... +n*Cn*Xn, combine with the result above, mpg=34.96-0.018*disp-2*3.35*wt

**Example 3, logistic regression

Dependent variable should be binary (True/False, 0/1), such as "am" in mydata

use glm() function

```r
Logistic <- glm(formula=am~hp+wt+cyl, data=mydata, family = binomial)

print(summary(Logistic))
```

```
##
## Call:
## glm(formula = am ~ hp + wt + cyl, family = binomial, data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1727  -0.1491  -0.0146   0.1412   1.2764
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  19.7029     8.1164    2.43    0.015 *
## hp            0.0326     0.0189    1.73    0.084 .
## wt           -9.1495     4.1533   -2.20    0.028 *
## cyl           0.4876     1.0716    0.46    0.649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.2297  on 31  degrees of freedom
## Residual deviance:  9.8415  on 28  degrees of freedom
## AIC: 17.84
##
## Number of Fisher Scoring iterations: 8
```

```
## the result above indicates that only "wt" affects "am" value (p<0.05)

## (a) Predict technique
str(mydata)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
table(mydata$am)
```

```
##
##  0  1
## 19 13
```

```
## split data into training and testing data
library(caTools)
```

```r
set.seed(88)
split<-sample.split(mydata$am, SplitRatio = 0.59)
qt<-subset(mydata,split==TRUE)
qs<-subset(mydata,split==FALSE)

## qt has training set sample data and qs has test set sample data
## Next using Summary () gives the details of deviance and co-efficient tables for regression analysis

Logistic<-glm(formula=am~hp+wt+cyl, data=qt, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
print(summary(Logistic))
```

```
##
## Call:
## glm(formula = am ~ hp + wt + cyl, family = binomial, data = qt)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.32e-05  -2.10e-08  -2.10e-08   2.10e-08   2.20e-05
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.38e+02   3.98e+05       0        1
## hp           2.49e-01   1.32e+03       0        1
## wt          -1.30e+02   1.58e+05       0        1
## cyl          5.04e+00   5.67e+04       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.5864e+01  on 18  degrees of freedom
## Residual deviance: 1.1613e-09  on 15  degrees of freedom
## AIC: 8
##
## Number of Fisher Scoring iterations: 25
```

```r
predicttrain<-predict(Logistic,type="response")
summary(predicttrain)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00    0.42    1.00    1.00
```

```r
## To compute the average for the true probabilities tapply() function
tapply(predicttrain,qt$am, mean)
```

```
##       0       1
## 2.8e-11 1.0e+00
```

```
## (b) Calculating Threshold Value
## if P is > T- prediction is poor Special MM; if P is <T- Prediction is good

table(qt$am,predicttrain >0.7)
```

```
##
##     FALSE TRUE
##   0    11    0
##   1     0    8
```

```
## Compute Sensitivity and Specificity, 11/11=1; 8/8=1

## test set data prediction
predictest<-predict(Logistic, type = "response", newdata = qs)
table(qs$am,predictest >= 0.7)
```

```
##
##     FALSE TRUE
##   0     7    1
##   1     1    4
```

Then calculate the accuracy: (7+4)/13=0.846

****Example 4, possion regression,used for predictive analysis where there are multiple numbers of possible outcomes expected which are countable in numbers**

we use another set of data for this example

```
library(robust)
```

```
## Loading required package: fit.models
```

```
data("stack.dat",package="robust")
stack.dat
```

```
##    Loss Air.Flow Water.Temp Acid.Conc.
## 1    42       80         27         89
## 2    37       80         27         88
## 3    37       75         25         90
## 4    28       62         24         87
## 5    18       62         22         87
## 6    18       62         23         87
## 7    19       62         24         93
## 8    20       62         24         93
## 9    15       58         23         87
## 10   14       58         18         80
## 11   14       58         18         89
## 12   13       58         17         88
## 13   11       58         18         82
```

```
## 14   12       58         19         93
## 15    8       50         18         89
## 16    7       50         18         86
## 17    8       50         19         72
## 18    8       50         19         79
## 19    9       50         20         80
## 20   15       56         20         82
## 21   15       70         20         91
```

```
fit<-glm(Loss~.,data=stack.dat, family = poisson(link="log"))
summary(fit)
```

```
##
## Call:
## glm(formula = Loss ~ ., family = poisson(link = "log"), data = stack.dat)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.992  -0.457  -0.171   0.642   1.651
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.78848    1.04687   -0.75   0.4513
## Air.Flow     0.02874    0.00959    3.00   0.0027 **
## Water.Temp   0.07284    0.02896    2.52   0.0119 *
## Acid.Conc.   0.00300    0.01322    0.23   0.8202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 104.426  on 20  degrees of freedom
## Residual deviance:   9.016  on 17  degrees of freedom
## AIC: 113.1
##
## Number of Fisher Scoring iterations: 4
```

```
## Air flow and Water.Temp have p value <0.05, so their changes will affect "Loss"
## Next you can calculate the coefficient to explain the model.

exp(coef(fit))
```

```
## (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
##        0.45        1.03        1.08        1.00
```

```
## The results indicate that every one unit of Air flow increase will cause 3% of loss, one unit of Wate

## Check if there is over dispersion, by useing "qcc" package

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

```
## Package 'qcc' version 2.7
```

```
## Type 'citation("qcc")' for citing this R package in publications.
```

```
qcc.overdispersion.test(stack.dat$Loss, type = "poisson")
```

```
##
## Overdispersion test Obs.Var/Theor.Var Statistic p-value
##        poisson data               5.9       118 6.7e-16
```

```
## P value < 0.05 so there is overdispersion
## if there is overdispersion, we need to replace "possion" by "quasipossion" in glm()

fit2<-glm(Loss~.,data=stack.dat, family = quasipoisson(link="log"))
summary(fit2)
```

```
##
## Call:
## glm(formula = Loss ~ ., family = quasipoisson(link = "log"),
##     data = stack.dat)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.992  -0.457  -0.171   0.642   1.651
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.78848    0.77484   -1.02  0.32313
## Air.Flow     0.02874    0.00710    4.05  0.00083 ***
## Water.Temp   0.07284    0.02143    3.40  0.00342 **
## Acid.Conc.   0.00300    0.00979    0.31  0.76255
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 0.55)
##
##     Null deviance: 104.426  on 20  degrees of freedom
## Residual deviance:   9.016  on 17  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

##~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

(4) Cluster analysis (using Kmeans)

**For example, we want to seperate data via "mpg" and "wt"

```
library(dplyr) #"dplyr" is for the select() function here
mydata <- mtcars
K <- dplyr::select(mydata,mpg,wt)
head(K)
```

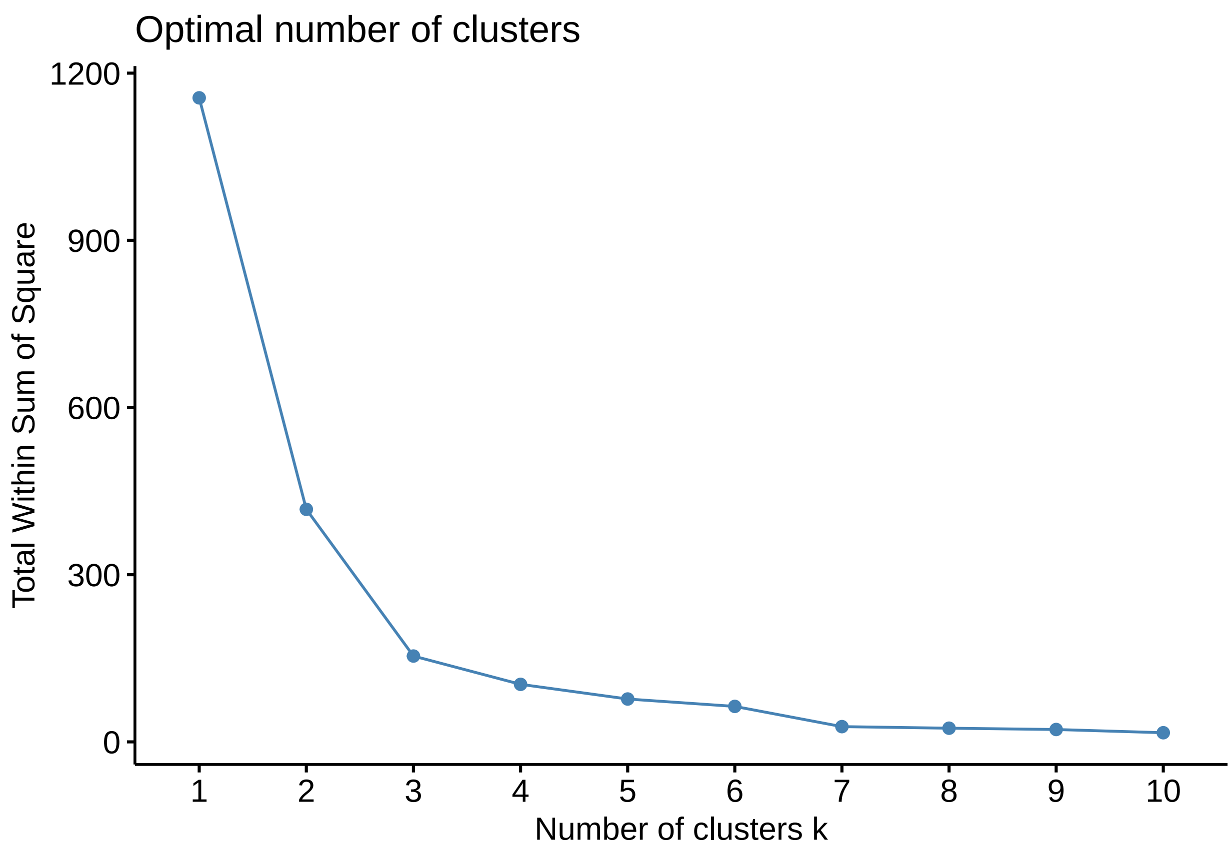
```
##                     mpg  wt
## Mazda RX4           21 2.6
## Mazda RX4 Wag       21 2.9
## Datsun 710          23 2.3
## Hornet 4 Drive      21 3.2
## Hornet Sportabout   19 3.4
## Valiant             18 3.5
```

```
 ## Load factoextra package, to measure the center (how many cluster it should be)

library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(K,kmeans,method = "wss")
```




```
## It coulde be split in to 3 cluster
## Then use kmeans() function, kmeans(data.name, center= )

cl<-kmeans(K,center=3)

## See the details of the cluster analysis result (which group is each object in through cl$cluster and

head(cl)
```

```
## $cluster
##          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##                  1                   1                   1                   1
##   Hornet Sportabout             Valiant          Duster 360           Merc 240D
##                  1                   1                   2                   1
##           Merc 230            Merc 280           Merc 280C          Merc 450SE
##                  1                   1                   1                   2
##         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##                  2                   2                   2                   2
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##                  2                   3                   3                   3
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##                  1                   2                   2                   2
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##                  1                   3                   3                   3
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##                  2                   1                   2                   1
##
## $centers
##   mpg  wt
## 1  21 3.1
## 2  14 4.1
## 3  30 1.9
##
## $totss
## [1] 1156
##
## $withinss
## [1] 51 58 45
##
## $tot.withinss
## [1] 154
##
## $betweenss
## [1] 1002
```

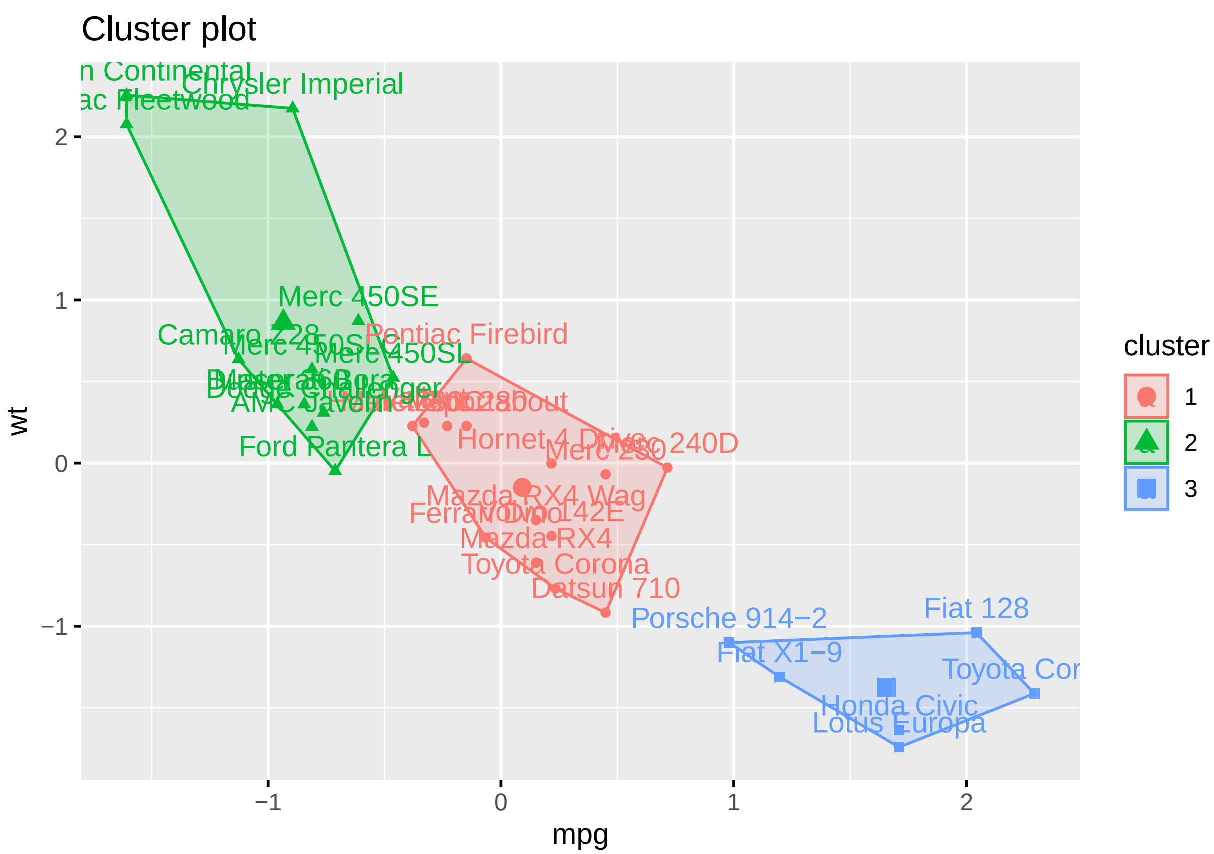
```
cl$cluster
```

```
##          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##                  1                   1                   1                   1
##   Hornet Sportabout             Valiant          Duster 360           Merc 240D
##                  1                   1                   2                   1
##           Merc 230            Merc 280           Merc 280C          Merc 450SE
##                  1                   1                   1                   2
##         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##                  2                   2                   2                   2
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##                  2                   3                   3                   3
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##                  1                   2                   2                   2
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##                  1                   3                   3                   3
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##                  2                   1                   2                   1
```

```
cl$size
```

```
## [1] 14 12  6
```

```
## plot the result
fviz_cluster(object = cl,data=K)
```



4. Permutation Test (by Kaiyu)

In simple words, the permutation hypothesis test in R is a way of comparing a ***numerical value of 2 groups***.

The permutation Hypothesis test is an alternative to:

- Independent two-sample t-test
- Mann-Whitney U aka Wilcoxon Rank-Sum Test

```
# Load data
d <- chickwts[49:71,]
print(d)
```

```
##    weight     feed
```

```
## 49    325 meatmeal
## 50    257 meatmeal
## 51    303 meatmeal
## 52    315 meatmeal
## 53    380 meatmeal
## 54    153 meatmeal
## 55    263 meatmeal
## 56    242 meatmeal
## 57    206 meatmeal
## 58    344 meatmeal
## 59    258 meatmeal
## 60    368   casein
## 61    390   casein
## 62    379   casein
## 63    260   casein
## 64    404   casein
## 65    318   casein
## 66    352   casein
## 67    359   casein
## 68    216   casein
## 69    222   casein
## 70    283   casein
## 71    332   casein
```

```r
# check the names
names(d)
```

```
## [1] "weight" "feed"
```

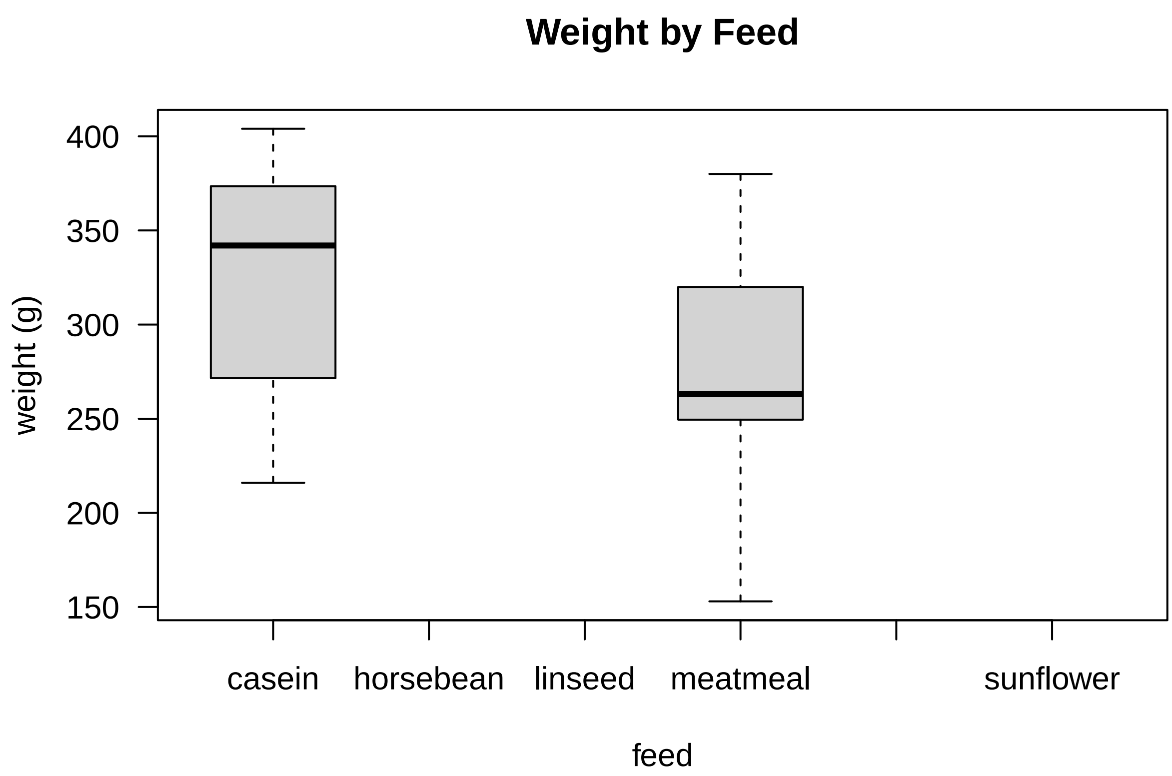
```r
levels(d$feed)
```

```
## [1] "casein"    "horsebean" "linseed"   "meatmeal"  "soybean"   "sunflower"
```

```r
#how many observations in each diet?
table(d$feed)
```

```
##
##    casein horsebean   linseed  meatmeal   soybean sunflower
##        12         0         0        11         0         0
```

```r
#let's look at a boxplot of weight gain by those 2 diets
boxplot(d$weight~d$feed, las = 1,
        ylab = "weight (g)",
        xlab = "feed",
        main = "Weight by Feed")
```




```
# calculate the difference in sample MEANS & their absolute diff
mean(d$weight[d$feed == "casein"]) # mean for casein
```

```
## [1] 324
```

```
mean(d$weight[d$feed == "meatmeal"]) # mean for meatmeal
```

```
## [1] 277
```

```
test.stat1 <- abs(mean(d$weight[d$feed == "casein"]) -
                  mean(d$weight[d$feed == "meatmeal"]))
test.stat1
```

```
## [1] 47
```

```
# calculate the difference in sample MEDIANS & their abs diff
median(d$weight[d$feed == "casein"]) # median for casein
```

```
## [1] 342
```

```
median(d$weight[d$feed == "meatmeal"]) # median for meatmeal
```

```
## [1] 263
```

```r
test.stat2 <- abs(median(d$weight[d$feed == "casein"]) -
                  median(d$weight[d$feed == "meatmeal"]))
test.stat2
```

```
## [1] 79
```

```r
########################
### Permutation Test ###
########################

set.seed(1979)  #for reproducability of results
n <- length(d$feed) #the number of observations to sample
P <- 100  #the number of permutation samples to take.eg 1000,100000
variable <- d$weight  #the variable we will resample from
PermSamples <- matrix(0, nrow = n, ncol = P) #initialize a matrix to store the permutation data
n
```

```
## [1] 23
```

```r
P
```

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

```r
variable
```

```
##  [1] 325 257 303 315 380 153 263 242 206 344 258 368 390 379 260 404 318 352 359
## [20] 216 222 283 332
```

```r
dim(PermSamples)
```

```
## [1]  23 100
```

```r
#each column is a permutation sample of data
#now, get those permutation samples, using a loop
#let's take a moment to discuss what that code is doing

for(i in 1:P)
  {
    PermSamples[, i] <- sample(variable,
                               size = n,
                               replace = FALSE)
}

# we can take a quick look at the first 5 columns of PermSamples (P(=100) columns in total)
PermSamples[, 1:5]
```

```
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]  379  283  380  352  206
##  [2,]  380  303  258  260  380
##  [3,]  257  206  379  380  153
```

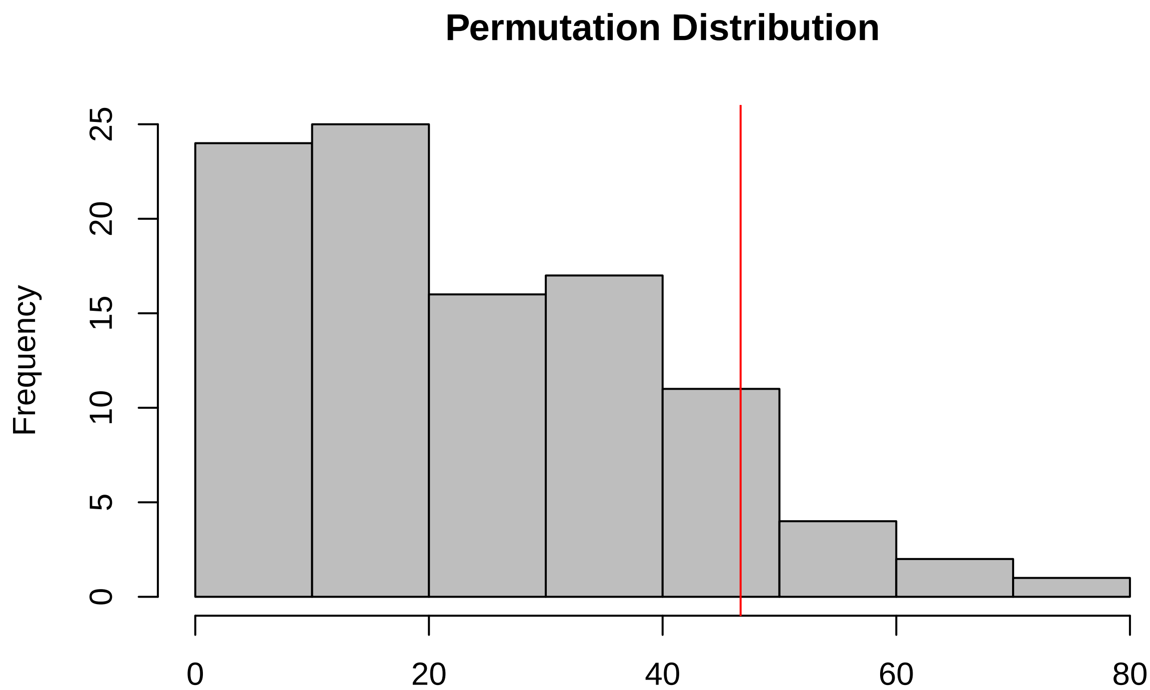
```
##  [4,]  283  242  222  404  359
##  [5,]  222  260  325  258  258
##  [6,]  315  352  153  379  263
##  [7,]  352  263  263  325  325
##  [8,]  153  325  315  359  216
##  [9,]  368  379  344  242  260
## [10,]  344  258  368  368  257
## [11,]  359  257  206  257  315
## [12,]  206  153  404  222  303
## [13,]  404  344  303  390  390
## [14,]  325  318  318  303  352
## [15,]  242  404  332  263  404
## [16,]  390  380  257  206  379
## [17,]  260  332  216  315  318
## [18,]  303  359  352  344  368
## [19,]  263  222  242  283  222
## [20,]  332  368  260  332  344
## [21,]  318  315  283  318  283
## [22,]  216  390  390  153  332
## [23,]  258  216  359  216  242
```

```
# let's calculate the test-statistics for permutation samples
# first, initialize empty vectors to store all of the Test-stats
Perm.test.stat1 <- Perm.test.stat2 <- rep(0, P)

# loop thru, and calculate the test-stats
for (i in 1:P)
  {
    # calculate the perm-test-stat1 and save it
    Perm.test.stat1[i] <- abs(mean(PermSamples[d$feed == "casein",i])
                          - mean(PermSamples[d$feed ==
                                              "meatmeal",i]))

    # calculate the perm-test-stat2 and save it
    Perm.test.stat2[i] <- abs(median(PermSamples[d$feed=="casein",i])
                          - median(PermSamples[d$feed ==
                                               "meatmeal",i]))
}

## plot of results
p <- hist(Perm.test.stat1,col='grey', main="Permutation Distribution", xlab='')
abline(v=test.stat1, col="red")
```




```
p
```

```
## $breaks
## [1]  0 10 20 30 40 50 60 70 80
##
## $counts
## [1] 24 25 16 17 11  4  2  1
##
## $density
## [1] 0.024 0.025 0.016 0.017 0.011 0.004 0.002 0.001
##
## $mids
## [1]  5 15 25 35 45 55 65 75
##
## $xname
## [1] "Perm.test.stat1"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
#Take a look at the first 15 permutation-TEST STATS for 1 and 2
test.stat1
```

```
## [1] 47
```

```
test.stat2
```

```
## [1] 79
```

```
round(Perm.test.stat1[1:15], 1)
```

```
##  [1] 17.1 32.4 17.6 47.1 56.1 28.9 31.0 40.8  6.8 13.8  9.1 46.5 28.9 50.9 32.7
```

```
round(Perm.test.stat2[1:15], 1)
```

```
##  [1] 61.0 75.0  4.5 59.0 78.0 17.0 62.0 38.5  4.5 16.0 23.0 60.5 63.5 75.0 37.0
```

```
# Definition Note:p-value for permuation test:what is the probability of getting the observed test stat

#### p-value = M(Perm.test.stat >= test.stat)/N(total number of Perm.test.stat) ###

#and, let's calculate the permutation p-value; notice how we can ask R a true/false question
M_15 <- (Perm.test.stat1 >= test.stat1)[1:15]
M_15
```

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

```
#and if we ask for the mean of all of those,it treats 0 = FALSE, 1 = TRUE (3/15)
p_mean_15 <- mean((Perm.test.stat1 >= test.stat1)[1:15])
p_mean_15
```

```
## [1] 0.2
```

```
#Calculate the p-value, for all P = 100 (probability getting the observed statistics or larger)
p_mean <- mean(Perm.test.stat1 >= test.stat1)
p_mean
```

```
## [1] 0.11
```

```
#and, let's calculate the p-value for option 2 of the test statistic (abs diff in medians)
p_median <- mean(Perm.test.stat2 >= test.stat2)
p_median
```

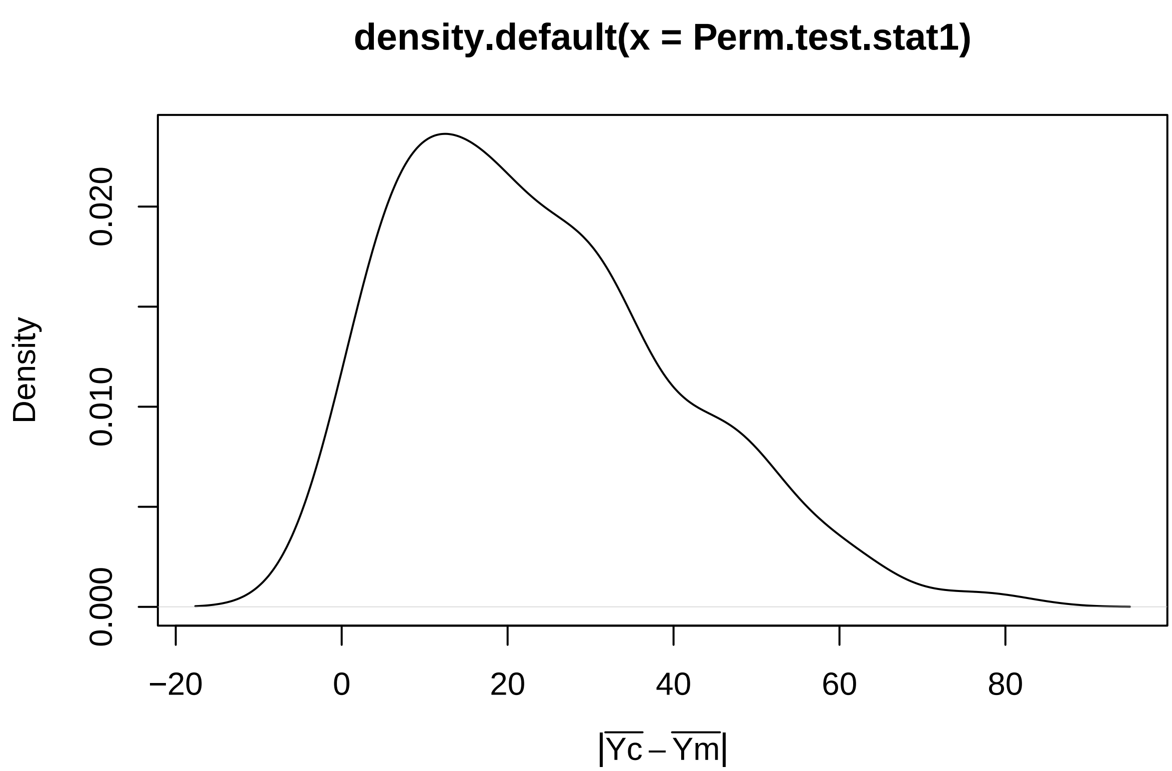
```
## [1] 0.03
```

```
## Finally, draw a conclusion:if p-value >= 0.05, cannot reject H0;if p-value <= 0.05, reject H0, sig. 
```

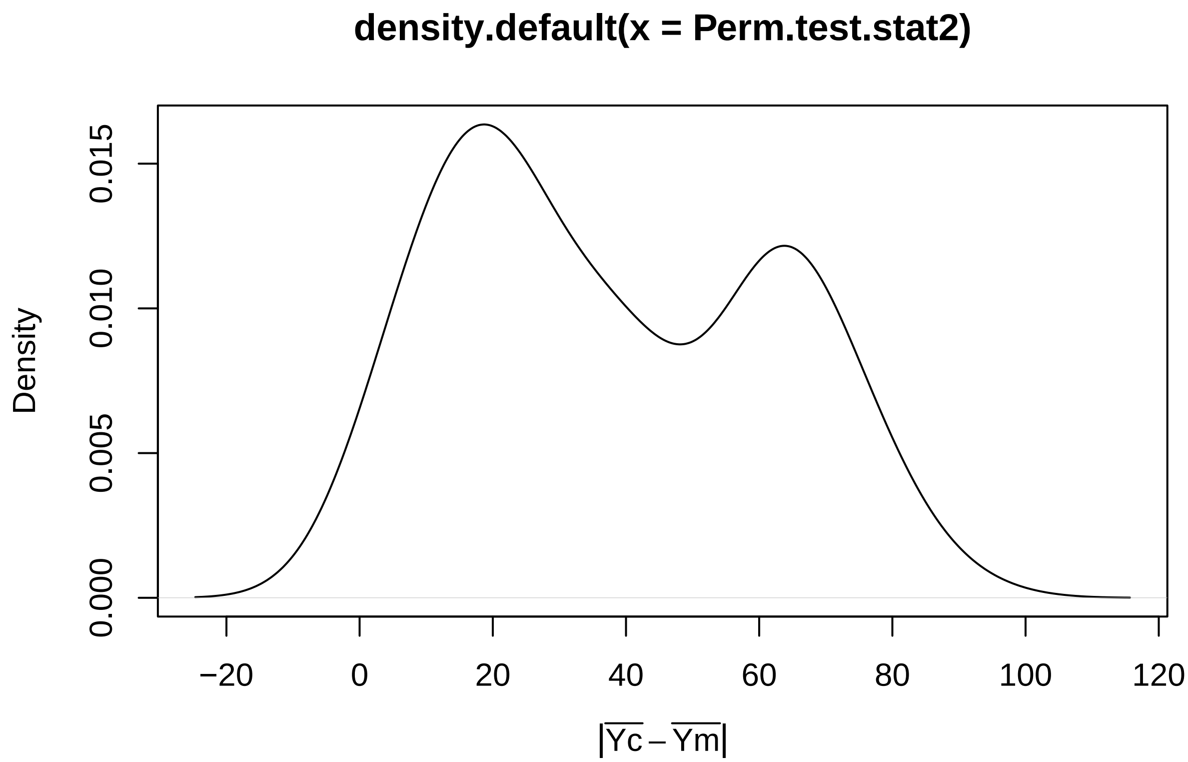
```
table(d$weight)
```

```
##
## 153 206 216 222 242 257 258 260 263 283 303 315 318 325 332 344 352 359 368 379
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 380 390 404
##   1   1   1
```

```
#DENSITY PLOT:(Yc and Ym represent the means for each of the permutation samples
p1 <- plot(density(Perm.test.stat1),
     xlab=expression(group("|", bar(Yc)-bar(Ym),"|")))
```




```
p2 <- plot(density(Perm.test.stat2),
     xlab=expression(group("|", bar(Yc)-bar(Ym),"|")))
```



5.Bootstrapping Resampling & Bootstrap Confidence Interval

Bootstrapping is a non-parametric statistical method for inference about a population using sample data.

For demonstration purposes, we are going to use the ChickWeight dataset due to simplicity and availability as one of the built-in datasets in R.

```
# View the first row of the ChickWeight dataset
d <- ChickWeight
head(d,1)
```

```
## Grouped Data: weight ~ Time | Chick
##   weight Time Chick Diet
## 1     42    0     1    1
```

We want to estimate the correlation between **weight** and **Time**

Steps to Compute the Bootstrap CI in R:

```
# 1. Import the boot library for calculation of bootstrap CI and ggplot2 for plotting.
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:psych':
##
##     logit
```

```
library(ggplot2)
```

```
# 2. Create a function that computes the statistic we want to use such as mean, median, correlation, et
# Custom function to find correlation between the weight and Time
corr.fun <- function(data, idx)
{
  df <- data[idx, ]

  # Find the spearman correlation between
  # the 1st and 4th columns of dataset
  c(cor(df[,1], df[,2], method = 'spearman'))
}
```

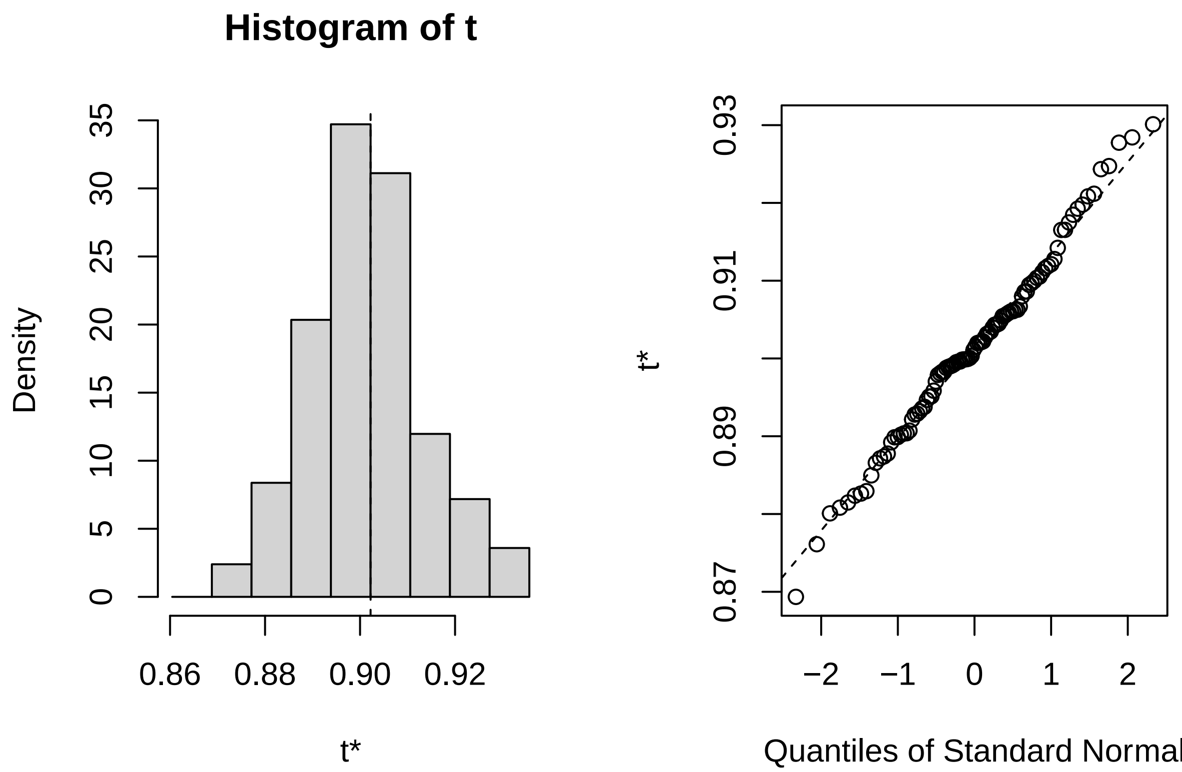
```
# 3. Using the boot function to find the R bootstrap of the statistic.
set.seed(42)# Setting the seed for reproducability of results

bootstrap <- boot(d, corr.fun, R = 100)# Calling the boot function with the dataset our function and no

bootstrap # Display the result of boot function
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = d, statistic = corr.fun, R = 100)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*      0.9 -0.00065       0.012
```

```
# 4. We can plot the generated bootstrap distribution using the plot command with calculated bootstrap.
plot(bootstrap)# Plot the bootstrap sampling distribution using ggplot
```




```r
# 5. Using the boot.ci() function to get the confidence intervals.
boot.ci(boot.out = bootstrap,
        type = c("norm", "basic","perc")) # Function to find the bootstrap Confidence Intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootstrap, type = c("norm", "basic", "perc"))
##
## Intervals :
## Level      Normal              Basic              Percentile
## 95%   ( 0.88,  0.93 )   ( 0.88,  0.93 )   ( 0.88,  0.93 )
## Calculations and Intervals on Original Scale
## Some basic intervals may be unstable
## Some percentile intervals may be unstable
```

Reference

Part 1 Permutation Hypothesis Test in R with Examples:

https://www.youtube.com/watch?v=xRzEWLfEEIA&list=PLqzoL9-eJTNDp_bWyWBdw2ioA43B3dBrl&index=7

https://www.geeksforgeeks.org/permutation-hypothesis-test-in-r-programming/

Part 2 Bootstrapping Resampling & Bootstrap Confidence Interval

<https://www.geeksforgeeks.org/bootstrap-confidence-interval-with-r-programming/>

Bootstrap Hypothesis Testing in R with Example:

<https://www.youtube.com/watch?v=Zet-qmEEfCU&list=PLqzoL9-eJTNDp_bWyWBdw2ioA43B3dBrl&index=4>

Bootstrap Confidence Interval with R:

<https://www.youtube.com/watch?v=Om5TMGj9td4&list=PLqzoL9-eJTNDp_bWyWBdw2ioA43B3dBrl&index=5>

https://rpubs.com/riddhigupta1357/919205

https://data-flair.training/blogs/bootstrapping-in-r/