R Notebook

Bibek Sapkota

Vector Operators using R

task 1: Create two vector a and b

```
a <- 1:10
b <- 1:10
```

```
class(a)
```

```
## [1] "integer"
```

```
class(b)
```

```
## [1] "integer"
```

task 2: This is same as task-1

```
c <- seq(1:10)
d <- as.integer(c(1, 2, 3, 4, 5, 6, 7, 8, 9,10))
```

```
class(c)
```

```
## [1] "integer"
```

```
class(d)
```

```
## [1] "integer"
```

task 3:Adding two vector

```
(a+b)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

Using () for printing the output

```
y <- seq(1, 10, length.out= 5)
```

task 1: printing the expression

```
(y <- seq(1, 10, length.out= 5))
```

```
## [1]  1.00  3.25  5.50  7.75 10.00
```

Vector Operation using R

task 1: Multiplying two vector

```
a <- 1:10
b <- 5

(a*b)
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50
```

```
a <- 1:10
b <- 5
c <- c(2,3,4,5,6,7,8,9,10,11)

class(a)
```

```
## [1] "integer"
```

```
class(b)
```

```
## [1] "numeric"
```

```
class(c)
```

```
## [1] "numeric"
```

Blinding two column vector

```
ac = cbind(a,c)
print(ac)
```

```
##        a  c
##  [1,]  1  2
##  [2,]  2  3
##  [3,]  3  4
##  [4,]  4  5
```

```
##  [5,]  5  6
##  [6,]  6  7
##  [7,]  7  8
##  [8,]  8  9
##  [9,]  9 10
## [10,] 10 11
```

```r
acb <- ac * b
print(acb)
```

```
##        a  c
##  [1,]  5 10
##  [2,] 10 15
##  [3,] 15 20
##  [4,] 20 25
##  [5,] 25 30
##  [6,] 30 35
##  [7,] 35 40
##  [8,] 40 45
##  [9,] 45 50
## [10,] 50 55
```

```r
class(acb)
```

```
## [1] "matrix" "array"
```

Calculating mean

```r
d <- apply(acb, MARGIN = 1, FUN =mean)
print(d)
```

```
##  [1]  7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5
```

```r
(acbd <- cbind(acb, d))
```

```
##        a  c    d
##  [1,]  5 10  7.5
##  [2,] 10 15 12.5
##  [3,] 15 20 17.5
##  [4,] 20 25 22.5
##  [5,] 25 30 27.5
##  [6,] 30 35 32.5
##  [7,] 35 40 37.5
##  [8,] 40 45 42.5
##  [9,] 45 50 47.5
## [10,] 50 55 52.5
```

```r
class(acbd)
```

```
## [1] "matrix" "array"
```

```r
summary(acbd)
```

```
##        a               c               d        
##  Min.   : 5.00   Min.   :10.00   Min.   : 7.50  
##  1st Qu.:16.25   1st Qu.:21.25   1st Qu.:18.75  
##  Median :27.50   Median :32.50   Median :30.00  
##  Mean   :27.50   Mean   :32.50   Mean   :30.00  
##  3rd Qu.:38.75   3rd Qu.:43.75   3rd Qu.:41.25  
##  Max.   :50.00   Max.   :55.00   Max.   :52.50
```

Vector Recycling

```r
a <- 1:10
b <- 1:5

(a+b)
```

```
##  [1]  2  4  6  8 10  7  9 11 13 15
```

```r
a <- 1:10
b <- 1:7

(a+b)
```

```
## Warning in a + b: longer object length is not a multiple of shorter object
## length
```

```
##  [1]  2  4  6  8 10 12 14  9 11 13
```

Function in R

```r
best_practice <- c("Let", "the", "compiter", "do", "the", "work")

print_words <- function(sentence){
  print(sentence[1])
  print(sentence[2])
  print(sentence[3])
  print(sentence[4])
  print(sentence[5])
  print(sentence[6])
}

print_words(best_practice)
```

```
## [1] "Let"
## [1] "the"
## [1] "compiter"
## [1] "do"
## [1] "the"
## [1] "work"
```

```r
print_words(best_practice[-6])
```

```
## [1] "Let"
## [1] "the"
## [1] "compiter"
## [1] "do"
## [1] "the"
## [1] NA
```

Deleting element from vector

```r
best_practice[-6]
```

```
## [1] "Let"      "the"      "compiter" "do"       "the"
```

```r
best_practice <- c("Let", "the", "compiter", "do", "the", "work")

print_words <- function(sentence){
  for (word in sentence){
    print(word)
  }
}

print_words(best_practice)
```

```
## [1] "Let"
## [1] "the"
## [1] "compiter"
## [1] "do"
## [1] "the"
## [1] "work"
```

```r
print_words(best_practice[-6])
```

```
## [1] "Let"
## [1] "the"
## [1] "compiter"
## [1] "do"
## [1] "the"
```

Apply, lapply, sapply, vapply

task 1: apply

```
a <- 1:10
b <- 10:20

df <- data.frame(cbind (a, b) )
```

```
## Warning in cbind(a, b): number of rows of result is not a multiple of vector
## length (arg 1)
```

```
apply(df, MARGIN = 1, FUN = mean)
```

```
##  [1]  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5 14.5 10.5
```

task 2: lapply

```
a <- 1:10

lapply(a, MARGIN=1, FUN= mean)
```

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

task 3:sapply

```
sapply(a, MARGIN =1 , FUN= mean)
```

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

task4:vapply

```
data <- list(l1 = c(1, 2, 3, 4),
             l2 = c(5, 6, 7, 8))

vapply(data, max, double(1) )
```

```
## l1 l2
##  4  8
```

Conditional Statements

```
y <-10
if (y<20){
  x <- "Too Low"
}else{
  x <- "Too High"
}

print(x)
```

```
## [1] "Too Low"
```

```
temp <- 35

if (temp <= 0){
    "freezing"
}else if (temp <= 10){
  "cold"
}else if (temp <=20){
  "cool"
}else if (temp <=30){
  "warm"
}else{
  "hot"
}
```

```
## [1] "hot"
```

Exploring covid datasets

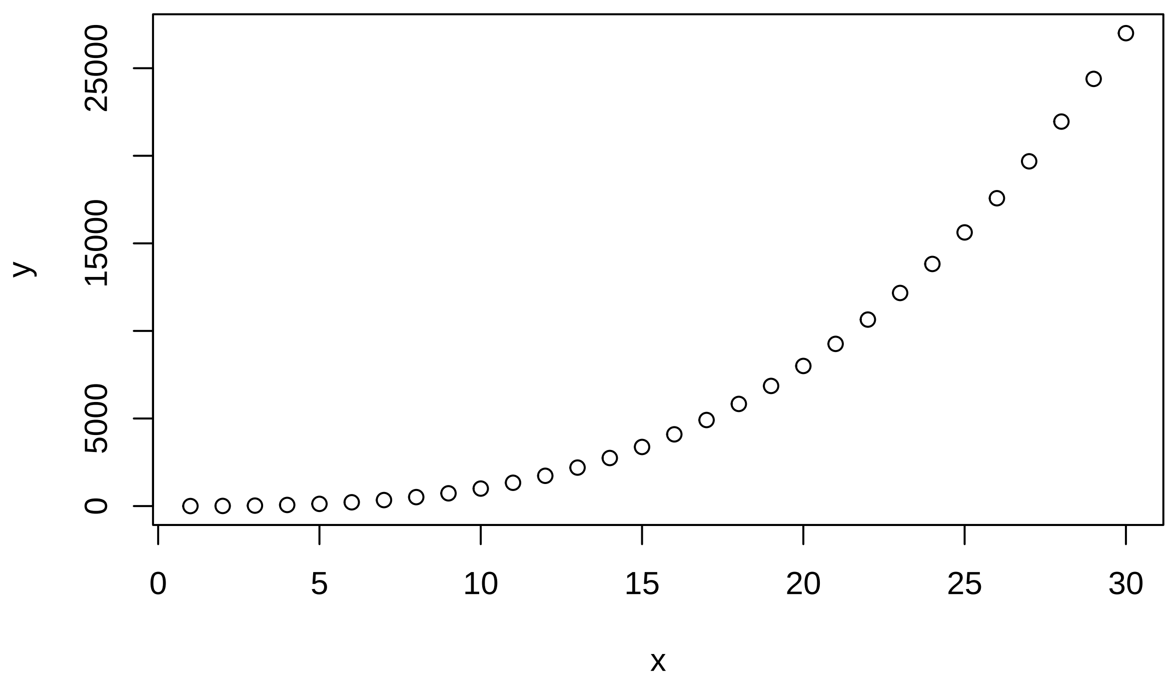
1. Create data frame with these two column vectors in R Studio x = 1:30 y = x^3

```r
x <- as.numeric(1:30)
y <- x^3

df <- data.frame(x,y)

str(df)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ x: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ y: num  1 8 27 64 125 216 343 512 729 1000 ...
```

2. Create plot of x and y variables in R Studio and interpret it carefully

```r
plot(x,y)
```



3. Get appropriate correlation coefficient of this data in R Studio and interpret it carefully

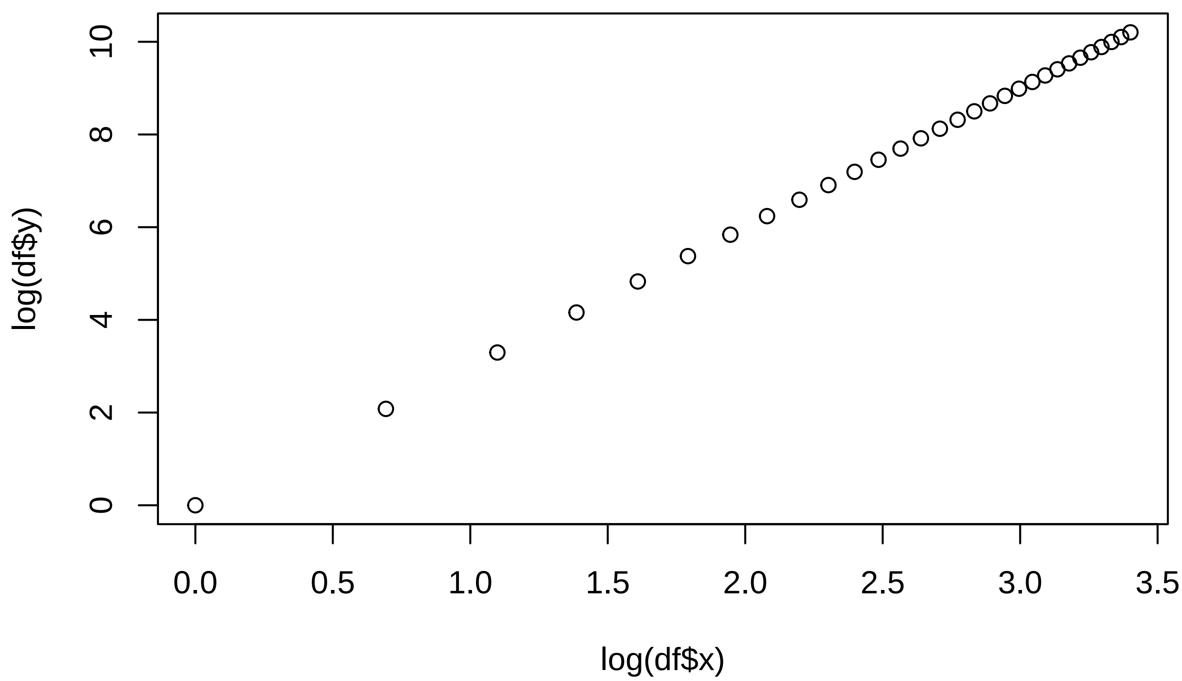
```r
corr <- cor.test(x= df$x, y=df$y, method="spearman")
corr
```

```
##
##  Spearman's rank correlation rho
##
```

```
## data:  df$x and df$y
## S = 0, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
##   1
```

4. Transform the plot to linear using appropriate mathematical function in R Studio

```r
plot(log(df$x), log(df$y))
```

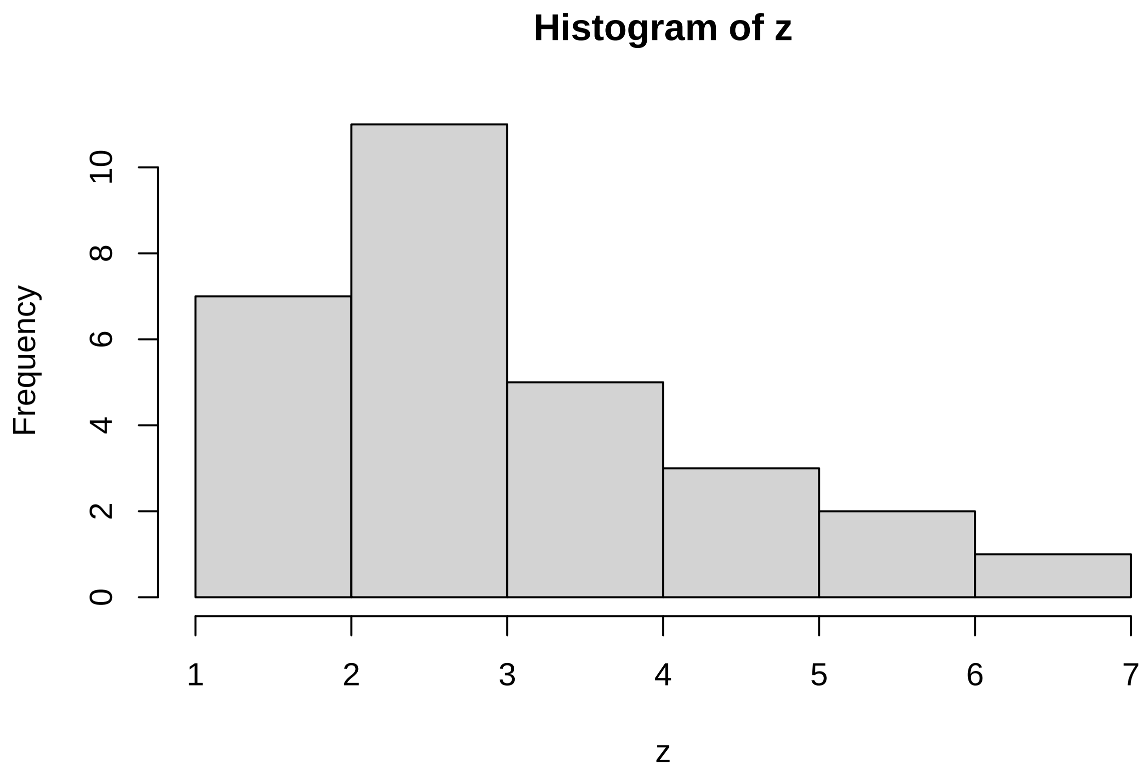


6. Create a new column vector z defined in the slide 18 of session two slide deck in R Studio.

```r
z <- c(1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,  4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 7)
```

7. Create a histogram of z variable in Rstudio and interpret it carefully

```r
hist(z)
```



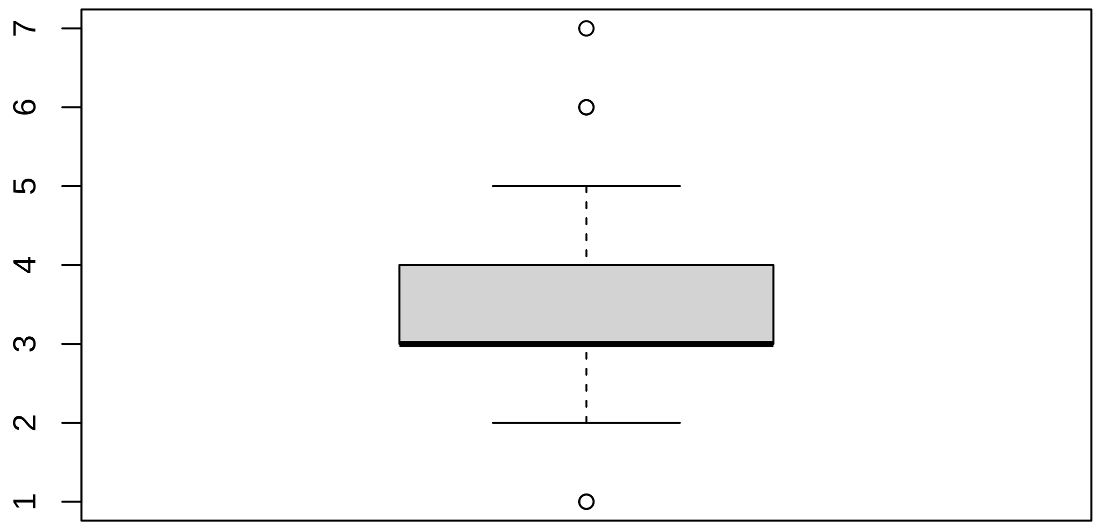
8.Get summary statistics of z variable in R Studio and interpret it carefully

```
summary(z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.000   3.000   3.000   3.414   4.000   7.000
```

9. Get box-plot of z variable in R Studio and interpret the result carefully

```
boxplot(z)
```



10. Import "covnep_252days.csv" data in R Studio and describe the variables in it.

```
packages_to_install <- c("readr")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(readr)
```

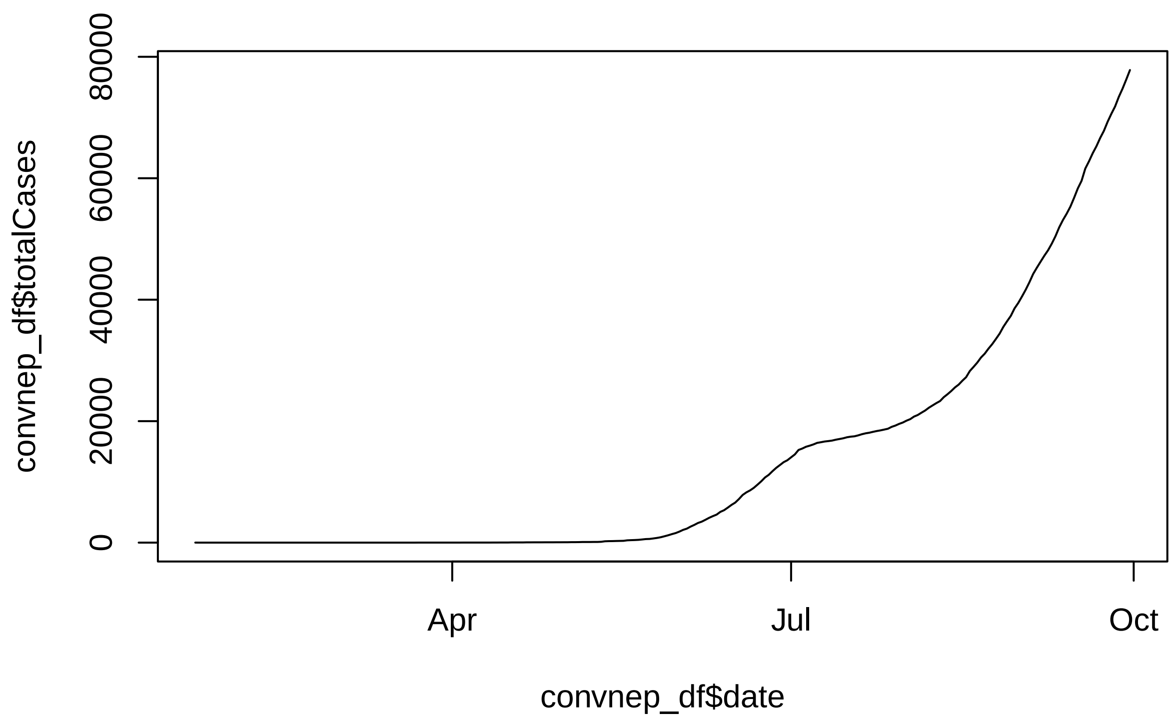
```
convnep_df <- read_csv("covnep_252days.csv", col_names = TRUE, col_types = cols(date = col_date(format =
str(convnep_df)
```

```
## spc_tbl_ [252 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date           : Date[1:252], format: "2020-01-23" "2020-01-24" ...
##  $ totalCases     : num [1:252] 1 0 0 0 0 0 0 0 0 0 ...
##  $ newCases       : num [1:252] 1 0 0 0 0 0 0 0 0 0 ...
##  $ totalRecoveries: num [1:252] 0 0 0 0 0 0 0 0 1 1 ...
##  $ newRecoveries  : num [1:252] 0 0 0 0 0 0 0 0 1 0 ...
##  $ totalDeaths    : num [1:252] 0 0 0 0 0 0 0 0 0 0 ...
##  $ newDeaths      : num [1:252] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
```

```
##    .. cols(
##    ..   date = col_date(format = "%m/%d/%Y"),
##    ..   totalCases = col_double(),
##    ..   newCases = col_double(),
##    ..   totalRecoveries = col_double(),
##    ..   newRecoveries = col_double(),
##    ..   totalDeaths = col_double(),
##    ..   newDeaths = col_double()
##    .. )
##  - attr(*, "problems")=<externalptr>
```

11. Create a chart with "totalCases" variable in y-axis and "date" variable in the x-axis in R Studio, describe the process leading to the creation of this chart.

```r
plot(convnep_df$date, convnep_df$totalCases, type='l')
```



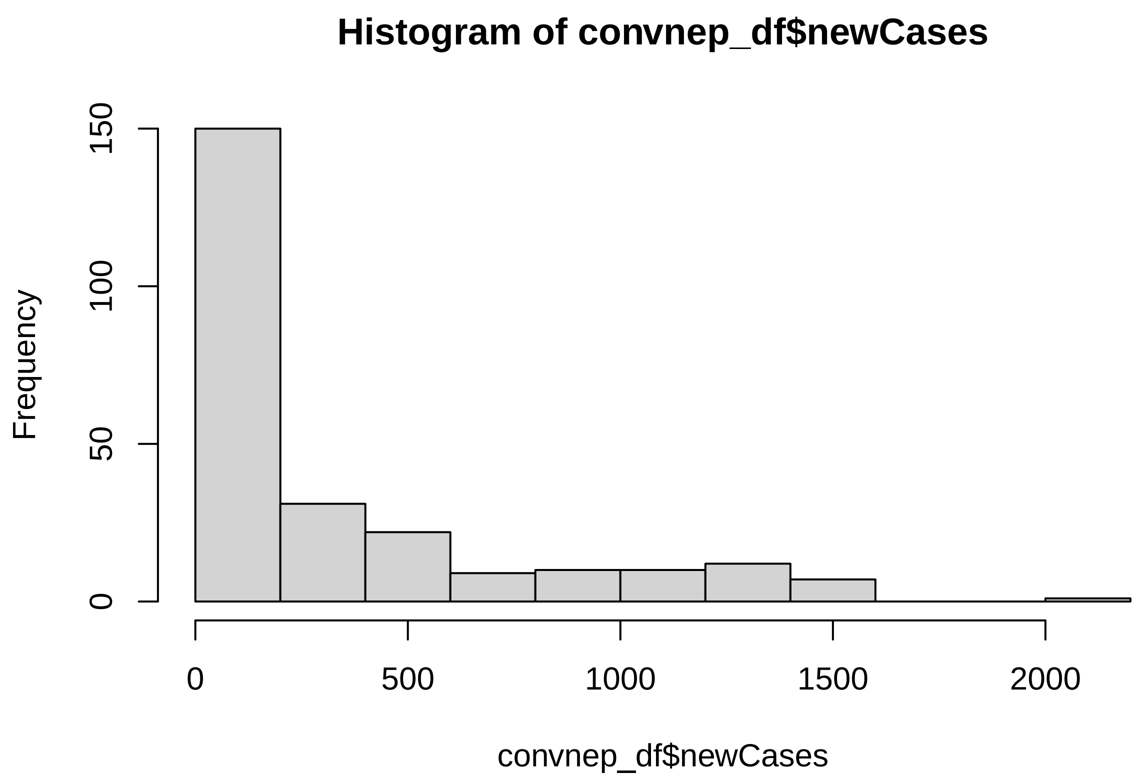
12. Get summary statistics of "totalCases" variable in R Studio and interpret it carefully

```r
summary(convnep_df$totalCases)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##        0       2     963   13376   19341   77816
```

13. Create histogram of "newCases" variable in R Studio and interpret it carefully.

```r
hist(convnep_df$newCases)
```



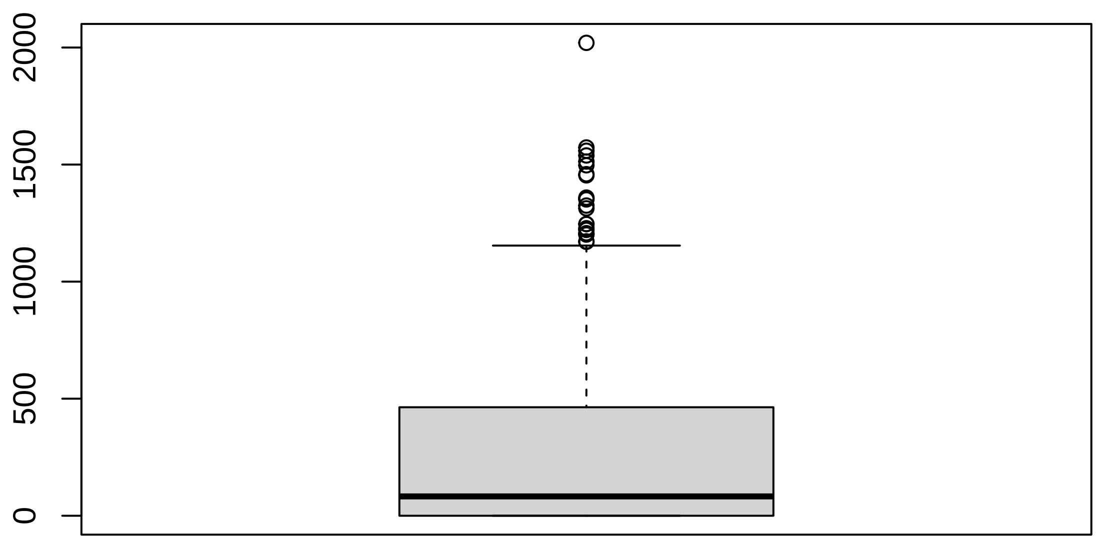
14. Get summary statistics of "newCases" variable in R Studio and interpret it carefully.

```r
summary(convnep_df$newCases)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0     0.0    82.5   308.8   463.2  2020.0
```

15. Get "box and whisker" plot of "newCases" variable in R Studio and interpret it carefully.

```r
boxplot(convnep_df$newCases)
```



16. Import “SAQ8.sav” data in R Studio and get frequency distribution (number and percentage of the attributes) of q01, q03, q06 and q08 variables on R Studio and interpret them carefully.

```
packages_to_install <- c("haven")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(haven)
```

```
saq8_df <- read_sav("SAQ8.sav")

str(saq8_df)
```

```
## tibble [2,571 x 8] (S3: tbl_df/tbl/data.frame)
##  $ q01: dbl+lbl [1:2571] 2, 1, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 3, 1, 2,...
##    ..@ label      : chr "Statistics makes me cry"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:6] 1 2 3 4 5 9
##    .. ..- attr(*, "names")= chr [1:6] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q02: dbl+lbl [1:2571] 1, 1, 3, 1, 1, 1, 3, 2, 3, 4, 1, 1, 1, 2, 2, 1, 2, 2,...
```

```
##    ..@ label      : chr "My friends will think I’m stupid for not being able to cope with SPSS"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q03: dbl+lbl [1:2571] 4, 4, 2, 1, 3, 3, 3, 3, 1, 4, 5, 3, 3, 1, 3, 2, 5, 3,...
##    ..@ label      : chr "Standard deviations excite me"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q04: dbl+lbl [1:2571] 2, 3, 2, 4, 2, 2, 2, 2, 4, 3, 2, 3, 4, 2, 4, 2, 2, 3,...
##    ..@ label      : chr "I dream that Pearson is attacking me with correlation coefficients"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:6] 1 2 3 4 5 9
##    .. ..- attr(*, "names")= chr [1:6] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q05: dbl+lbl [1:2571] 2, 2, 4, 3, 2, 4, 2, 2, 5, 2, 2, 4, 3, 2, 2, 2, 1, 3,...
##    ..@ label      : chr "I don’t understand statistics"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q06: dbl+lbl [1:2571] 2, 2, 1, 3, 3, 4, 2, 2, 3, 1, 1, 3, 2, 2, 2, 2, 1, 4,...
##    ..@ label      : chr "I have little experience of computers"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q07: dbl+lbl [1:2571] 3, 2, 2, 4, 3, 4, 2, 2, 5, 2, 2, 3, 3, 3, 3, 2, 1, 3,...
##    ..@ label      : chr "All computers hate me"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
##  $ q08: dbl+lbl [1:2571] 1, 2, 2, 2, 2, 2, 2, 2, 5, 2, 2, 1, 3, 2, 2, 2, 1, 2,...
##    ..@ label      : chr "I have never been good at mathematics"
##    ..@ format.spss: chr "F1.0"
##    ..@ labels     : Named num [1:5] 1 2 3 4 5
##    .. ..- attr(*, "names")= chr [1:5] "Strongly agree" "Agree" "Neither" "Disagree" ...
```

```r
head(saq8_df, 10)
```

```
## # A tibble: 10 x 8
##    q01                q02        q03     q04     q05     q06     q07     q08    
##    <dbl+lbl>          <dbl+lbl>  <dbl+l> <dbl+l> <dbl+l> <dbl+l> <dbl+l> <dbl+l>
##  1 2 [Agree]          1 [Strong~ 4 [Dis~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 3 [Nei~ 1 [Str~
##  2 1 [Strongly agree] 1 [Strong~ 4 [Dis~ 3 [Nei~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 2 [Agr~
##  3 2 [Agree]          3 [Neithe~ 2 [Agr~ 2 [Agr~ 4 [Dis~ 1 [Str~ 2 [Agr~ 2 [Agr~
##  4 3 [Neither]        1 [Strong~ 1 [Str~ 4 [Dis~ 3 [Nei~ 3 [Nei~ 4 [Dis~ 2 [Agr~
##  5 2 [Agree]          1 [Strong~ 3 [Nei~ 2 [Agr~ 2 [Agr~ 3 [Nei~ 3 [Nei~ 2 [Agr~
##  6 2 [Agree]          1 [Strong~ 3 [Nei~ 2 [Agr~ 4 [Dis~ 4 [Dis~ 4 [Dis~ 2 [Agr~
##  7 2 [Agree]          3 [Neithe~ 3 [Nei~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 2 [Agr~
##  8 2 [Agree]          2 [Agree]  3 [Nei~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 2 [Agr~ 2 [Agr~
##  9 3 [Neither]        3 [Neithe~ 1 [Str~ 4 [Dis~ 5 [Str~ 3 [Nei~ 5 [Str~ 5 [Str~
## 10 2 [Agree]          4 [Disagr~ 4 [Dis~ 3 [Nei~ 2 [Agr~ 1 [Str~ 2 [Agr~ 2 [Agr~
```

```r
packages_to_install <- c("plyr")
```

```r
for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(plyr)
```

```r
# Define a function
col_list <- list("q01", "q03", "q06", "q08")

# Create a function frequency table
frequency_table <- function(df, col_list) {
  for(item in col_list) {
    new_df <- count(df, item)
    new_df$percentage <- round(new_df$freq / sum(new_df$freq) * 100, 2)
    new_df$cumulative_percentage <- cumsum(new_df$percentage)
    print(new_df)
  }
}

frequency_table(saq8_df, col_list)
```

```
##   q01 freq percentage cumulative_percentage
## 1   1  270      10.50                 10.50
## 2   2 1338      52.04                 62.54
## 3   3  735      28.59                 91.13
## 4   4  187       7.27                 98.40
## 5   5   41       1.59                 99.99
##   q03 freq percentage cumulative_percentage
## 1   1  497      19.33                 19.33
## 2   2  672      26.14                 45.47
## 3   3  878      34.15                 79.62
## 4   4  448      17.43                 97.05
## 5   5   76       2.96                100.01
##   q06 freq percentage cumulative_percentage
## 1   1  702      27.30                 27.30
## 2   2 1127      43.84                 71.14
## 3   3  344      13.38                 84.52
## 4   4  252       9.80                 94.32
## 5   5  146       5.68                100.00
##   q08 freq percentage cumulative_percentage
## 1   1  383      14.90                 14.90
## 2   2 1487      57.84                 72.74
## 3   3  482      18.75                 91.49
## 4   4  147       5.72                 97.21
## 5   5   72       2.80                100.01
```

17. Import "MR_drugs.xls" data in R Studio and replicate multiple response frequency distribution as shown in the slide 35 of the session 2 slide deck.

```r
packages_to_install <- c("readxl")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(readxl)
```

```r
drugs_df <- readxl::read_excel("MR_drugs.xls")

str(drugs_df)
```

```
## tibble [972 x 27] (S3: tbl_df/tbl/data.frame)
##  $ id    : num [1:972] 1001 1002 1003 1004 1005 ...
##  $ sex   : num [1:972] 2 2 2 2 2 2 2 2 2 2 ...
##  $ city  : num [1:972] 1 1 1 1 1 1 1 1 1 1 ...
##  $ inco1 : num [1:972] 0 0 0 0 0 1 0 0 1 0 ...
##  $ inco2 : num [1:972] 0 1 0 1 0 1 1 1 1 1 ...
##  $ inco3 : num [1:972] 0 0 0 0 0 0 0 0 0 0 ...
##  $ inco4 : num [1:972] 0 0 0 0 0 0 0 0 0 0 ...
##  $ inco5 : num [1:972] 0 0 0 0 0 0 0 0 0 0 ...
##  $ inco6 : num [1:972] 1 0 1 0 0 0 0 0 0 0 ...
##  $ inco7 : num [1:972] 0 0 0 0 1 0 0 0 0 0 ...
##  $ pinco1: num [1:972] 6 2 6 2 7 2 2 2 2 2 ...
##  $ pinco2: num [1:972] -1 -1 -1 -1 -1 1 -1 -1 1 -1 ...
##  $ pinco3: num [1:972] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ pinco4: num [1:972] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ pinco5: num [1:972] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ pinco6: num [1:972] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ sinco1: chr [1:972] "\"mischeln\"/begging" "public support (unemployment insurance, social benefi
##  $ sinco2: chr [1:972] NA NA NA NA ...
##  $ sinco3: chr [1:972] NA NA NA NA ...
##  $ sinco4: chr [1:972] NA NA NA NA ...
##  $ sinco5: chr [1:972] NA NA NA NA ...
##  $ sinco6: chr [1:972] NA NA NA NA ...
##  $ crime1: num [1:972] 0 0 0 0 0 3 0 0 0 0 ...
##  $ crime2: num [1:972] 0 0 0 0 0 1 0 0 0 0 ...
##  $ crime3: num [1:972] 0 0 0 0 0 0 0 0 0 0 ...
##  $ crime4: num [1:972] 0 2 0 0 0 1 0 0 0 0 ...
##  $ crime5: num [1:972] 0 0 0 0 0 0 0 0 0 0 ...
```

```r
head(drugs_df, 10)
```

```
## # A tibble: 10 x 27
##       id   sex  city inco1 inco2 inco3 inco4 inco5 inco6 inco7 pinco1 pinco2
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
##  1  1001     2     1     0     0     0     0     0     1     0      6     -1
##  2  1002     2     1     0     1     0     0     0     0     0      2     -1
##  3  1003     2     1     0     0     0     0     0     1     0      6     -1
##  4  1004     2     1     0     1     0     0     0     0     0      2     -1
```

```
## 5 1005 2 1 0 0 0 0 0 0 1 7 -1
## 6 1006 2 1 1 1 0 0 0 0 0 2 1
## 7 1007 2 1 0 1 0 0 0 0 0 2 -1
## 8 1008 2 1 0 1 0 0 0 0 0 2 -1
## 9 1009 2 1 1 1 0 0 0 0 0 2 1
## 10 1010 2 1 0 1 0 0 0 0 0 2 -1
## # i 15 more variables: pinco3 <dbl>, pinco4 <dbl>, pinco5 <dbl>, pinco6 <dbl>,
## #   sinco1 <chr>, sinco2 <chr>, sinco3 <chr>, sinco4 <chr>, sinco5 <chr>,
## #   sinco6 <chr>, crime1 <dbl>, crime2 <dbl>, crime3 <dbl>, crime4 <dbl>,
## #   crime5 <dbl>
```

```
mr_drugs_df <- data.frame(
  N = colSums(drugs_df[4:10]),
  Percent = round(colSums(drugs_df[4:10]) / sum(drugs_df[4:10]) * 100, 2),
  "Percent of Cases" = round(colSums(drugs_df[4:10]) / nrow(drugs_df[4:10]) * 100, 2)
)

print(mr_drugs_df)
```

```
##         N Percent Percent.of.Cases
## inco1 226   12.83            23.25
## inco2 607   34.47            62.45
## inco3 293   16.64            30.14
## inco4  50    2.84             5.14
## inco5  82    4.66             8.44
## inco6 151    8.57            15.53
## inco7 352   19.99            36.21
```

```
# summary table as in spss
# import library for table

packages_to_install <- c("gt")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(gt)
```

```
# create table
mr_drugs_df %>%
  gt()%>%
  tab_header(
    title="$Income Frequencies"
  ) %>%
  tab_spanner(label="Responses", columns = c(N, Percent))
```

$Income Frequencies

Responses

N	Percent	Percent.of.Cases
226	12.83	23.25
607	34.47	62.45
293	16.64	30.14
50	2.84	5.14
82	4.66	8.44
151	8.57	15.53
352	19.99	36.21

```
#summary_rows(
 #   columns = everything(),
  #  fns = list(Total = ~mean(.))
  #)
```

#Pipe Operators

1. Initialize a vector

```
packages_to_install <- c("magrittr")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(magrittr)

x <-c (0.109, 0.359, 0.63, 0.996, 0.515, 0.142, 0.017, 0.829, 0.907)

# Compute the logarithm of `x`,
# return suitably lagged and iterated differences
# compute the exponential function and round the result
round(exp(diff(log(x))), 1)
```

```
## [1]  3.3  1.8  1.6  0.5  0.3  0.1 48.8  1.1
```

2. Using pipe operator

```
x %>%
  log() %>%
  diff() %>%
  exp() %>%
  round(1)
```

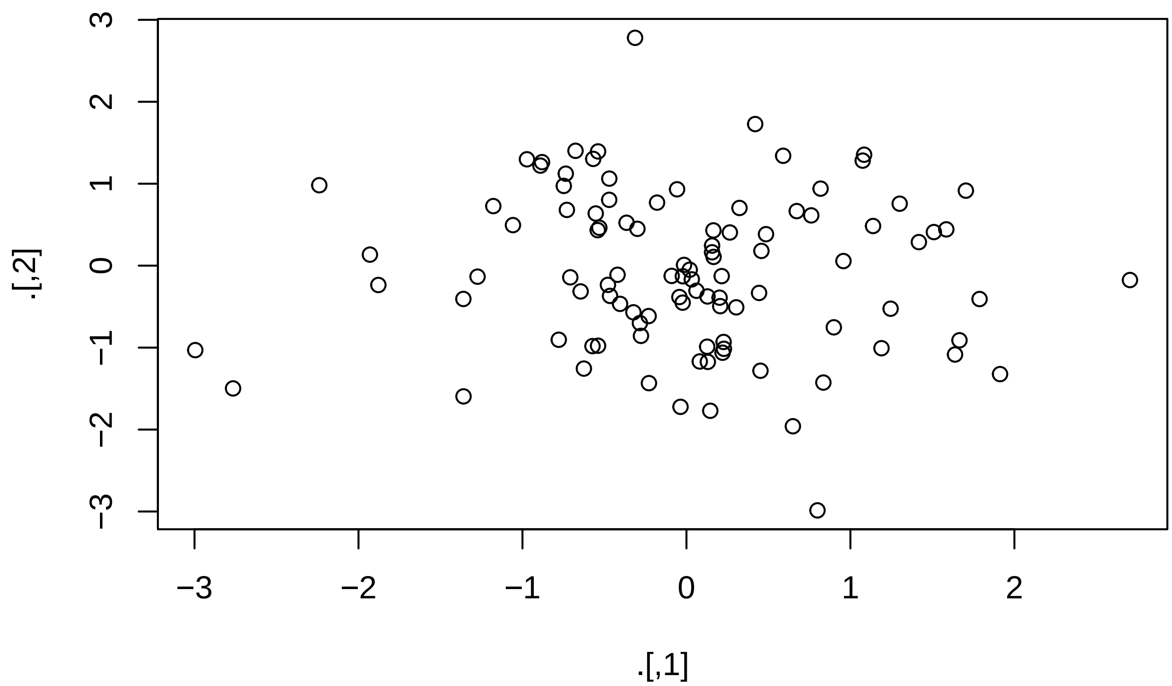
```
## [1]  3.3  1.8  1.6  0.5  0.3  0.1 48.8  1.1
```

Different pipe operators

```r
#Other pipe operators 1
x <- rnorm(100)

(x %<>% abs %>% sort)
```

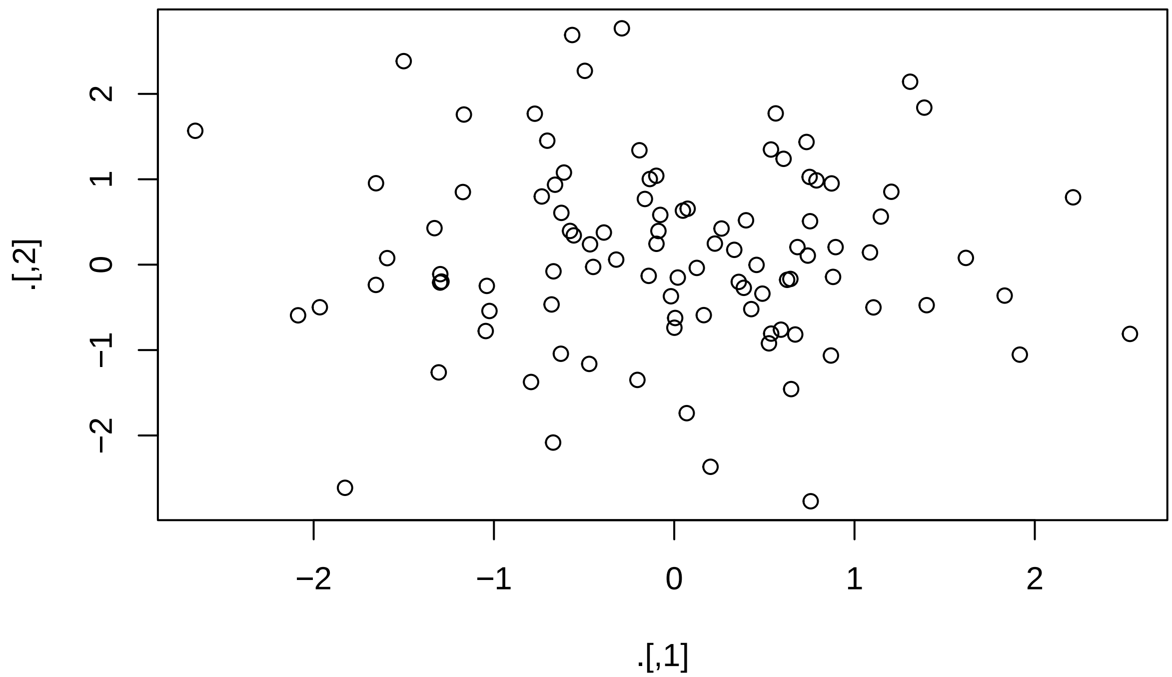
```
##   [1] 0.004093245 0.004262221 0.006113475 0.017795412 0.020892938 0.056065728
##   [7] 0.063070989 0.071135554 0.091030105 0.111880447 0.114010587 0.117473015
##  [13] 0.119974079 0.132067509 0.149131095 0.155004903 0.162610040 0.163555026
##  [19] 0.164954675 0.165536246 0.170908920 0.178528036 0.186923798 0.188977146
##  [25] 0.225020852 0.244298494 0.262143980 0.304448273 0.330951929 0.338661811
##  [31] 0.346328397 0.355931358 0.375466784 0.391652142 0.402323756 0.402498333
##  [37] 0.433014841 0.462301766 0.477870592 0.482685874 0.487264656 0.488267660
##  [43] 0.490093505 0.529369577 0.540302723 0.566040044 0.574668990 0.600145803
##  [49] 0.612207265 0.624845580 0.624889846 0.628504627 0.639769203 0.658771738
##  [55] 0.659626172 0.661889734 0.687013900 0.698318907 0.699669653 0.700467835
##  [61] 0.714460124 0.752403402 0.759130990 0.768196312 0.773560163 0.784808889
##  [67] 0.785807767 0.791135934 0.807299249 0.825196849 0.825280031 0.831745842
##  [73] 0.849033049 0.871872052 0.875690443 0.897737997 0.920792630 0.921526524
##  [79] 0.945481130 1.017022571 1.029749141 1.056786233 1.129341328 1.157679460
##  [85] 1.205346993 1.233996292 1.244801255 1.373790031 1.375166880 1.394655608
##  [91] 1.426830235 1.444876400 1.461537236 1.474708877 1.792193896 1.800239360
##  [97] 1.832710689 2.023938808 2.034946152 2.370883948
```

```r
#Other pipe operators 2
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  plot %>%
  colSums
```



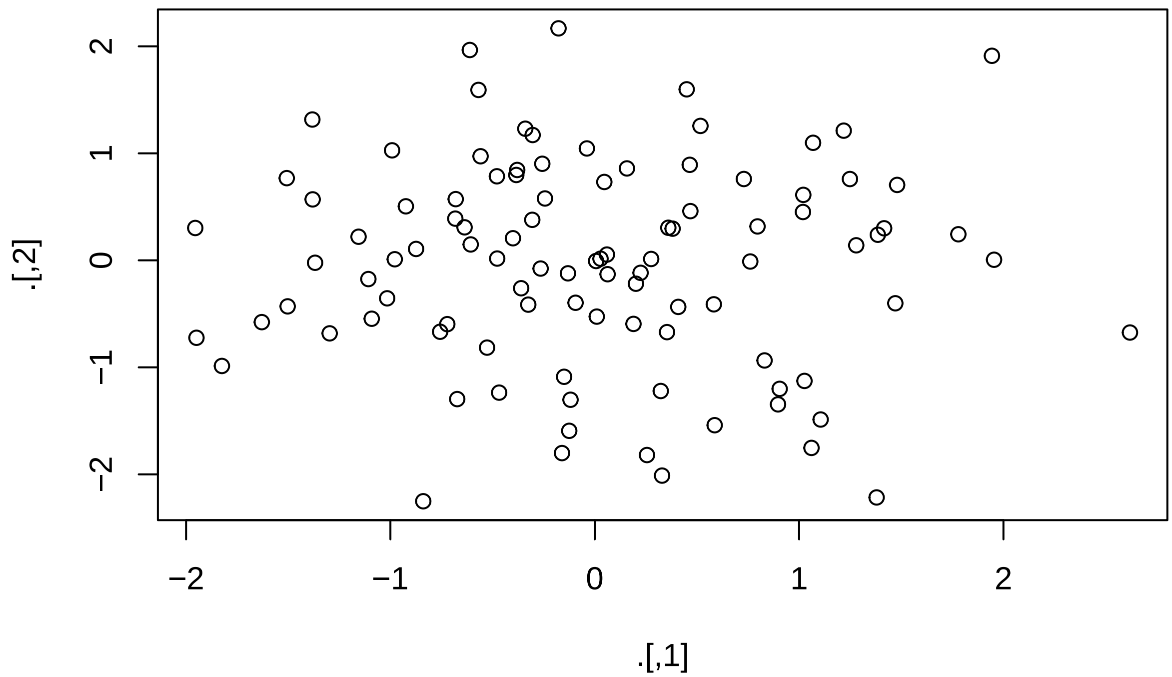

```
## [1] -0.02176629 -6.60027829
```

```
#Other pipe operators 2
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  plot %>%
  colSums
```



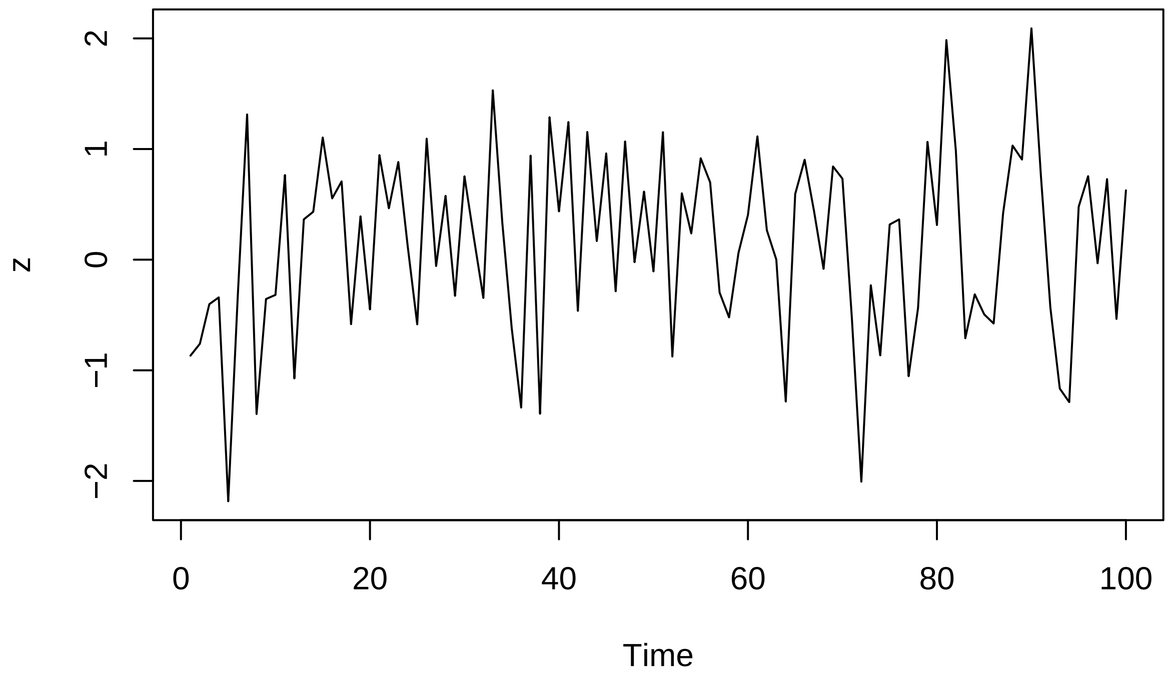

```
## [1] -2.802883 12.774739
```

```
#The above code is a shortcut for this code:
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  { plot(.); . } %>%
  colSums
```




```
## [1] -0.640430 -3.151475
```

```
#Other pipe operator 3
data.frame(z = rnorm(100)) %$%
  ts.plot(z)
```




```r
#More examples:
#Load the package, install if require!
packages_to_install <- c("babynames")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(babynames)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data("babynames")
```

```
sum(select(filter(babynames,sex=="M",name=="Taylor"),n))
```

```
## [1] 109852
```

```
# Do the same but now with `%>%`
babynames%>%filter(sex=="M",name=="Taylor")%>%
  select(n)%>%
  sum
```

```
## [1] 109852
```

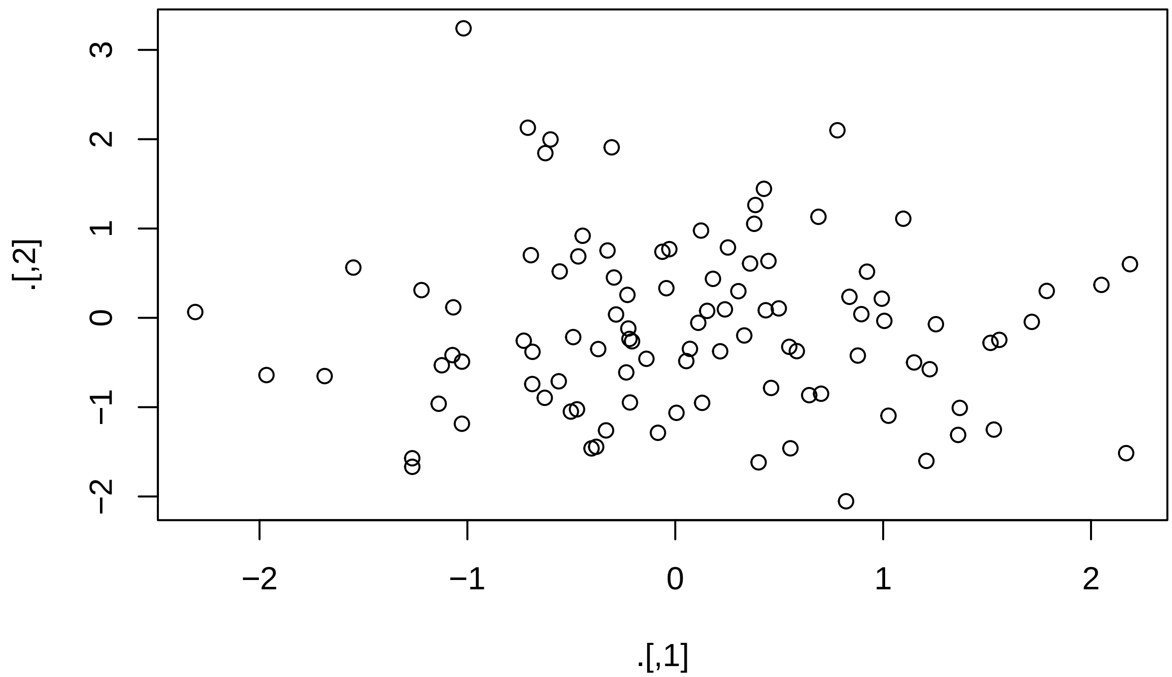
```
#Assigning new variable and using compound assignment pipe operator:
# Load in the Iris data
iris <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header
```

```
# Add column names to the Iris data
names(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")
```

```
# Compute the square root of `iris$Sepal.Length` and assign it to the new variable
iris$Sepal.Length.SQRT <-
  iris$Sepal.Length %>%
  sqrt()
```

```
#Compound pipe operator:
# Compute the square root of `iris$Sepal.Length` and assign it to the same variable
iris$Sepal.Length %<>% sqrt
```

```
#The tee operator:
set.seed(123)
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  plot %>%
  colSums
```




```
## [1]   9.040591 -10.754680
```

```r
#Exposing pipe operator: comes handy when "data" argument is not needed in a function
iris %>%
  subset(Sepal.Length > mean(Sepal.Length)) %$%
  cor(Sepal.Length, Sepal.Width)
```

```
## [1] 0.3365679
```

dplyr Package in R

```r
#install.packages("hflights")
packages_to_install <- c("hflights")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
    install.packages(package_name)
  }
}

library(hflights)
#Without pipe operators:
grouped_flights <- group_by(hflights, Year, Month, DayofMonth)
```

```r
flights_data <- select(grouped_flights, Year:DayofMonth, ArrDelay, DepDelay)
summarized_flights <- summarise(flights_data,
                                arr = mean(ArrDelay, na.rm = TRUE),      #Remove missing data!
                                dep = mean(DepDelay, na.rm = TRUE))      #Remove missing data!
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

```r
final_result <- filter(summarized_flights, arr > 30 | dep > 30)
final_result
```

```
## # A tibble: 14 x 5
## # Groups:   Year, Month [10]
##     Year Month DayofMonth   arr   dep
##    <int> <int>      <int> <dbl> <dbl>
##  1  2011     2          4  44.1  47.2
##  2  2011     3          3  35.1  38.2
##  3  2011     3         14  46.6  36.1
##  4  2011     4          4  38.7  27.9
##  5  2011     4         25  37.8  22.3
##  6  2011     5         12  69.5  64.5
##  7  2011     5         20  37.0  26.6
##  8  2011     6         22  65.5  62.3
##  9  2011     7         29  29.6  31.9
## 10  2011     9         29  39.2  32.5
## 11  2011    10          9  61.9  59.5
## 12  2011    11         15  43.7  39.2
## 13  2011    12         29  26.3  30.8
## 14  2011    12         31  46.5  54.2
```

```r
# With pipe operators:
hflights %>%
  group_by(Year, Month, DayofMonth) %>%
  select(Year:DayofMonth, ArrDelay, DepDelay) %>%
  summarise(arr = mean(ArrDelay, na.rm = TRUE), dep = mean(DepDelay, na.rm = TRUE)) %>%
  filter(arr > 30 | dep > 30)
```

```
## `summarise()` has grouped output by 'Year', 'Month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 14 x 5
## # Groups:   Year, Month [10]
##     Year Month DayofMonth   arr   dep
##    <int> <int>      <int> <dbl> <dbl>
##  1  2011     2          4  44.1  47.2
##  2  2011     3          3  35.1  38.2
##  3  2011     3         14  46.6  36.1
##  4  2011     4          4  38.7  27.9
##  5  2011     4         25  37.8  22.3
##  6  2011     5         12  69.5  64.5
##  7  2011     5         20  37.0  26.6
```

```
## 8  2011     6          22  65.5  62.3
## 9  2011     7          29  29.6  31.9
## 10 2011     9          29  39.2  32.5
## 11 2011    10           9  61.9  59.5
## 12 2011    11          15  43.7  39.2
## 13 2011    12          29  26.3  30.8
## 14 2011    12          31  46.5  54.2
```

```
#ARRNGE data with dplyr and pipe operators:
#Ascending order
iris %>%
  select(starts_with("Sepal")) %>%
  filter(Sepal.Length >=70) %>%
  arrange(Sepal.Length)     #Sort data in ascending order
```

```
## [1] Sepal.Length      Sepal.Width       Sepal.Length.SQRT
## <0 rows> (or 0-length row.names)
```

```
#Descending order:
iris %>%
  select(starts_with("Sepal")) %>%
  filter(Sepal.Length >=70) %>%
  arrange(desc(Sepal.Length))       #Sort data in descending order
```

```
## [1] Sepal.Length      Sepal.Width       Sepal.Length.SQRT
## <0 rows> (or 0-length row.names)
```

```
#MUTATE with dplyr and pipe operators:
iris %>%
  select(contains("Sepal")) %>%
  mutate(Sepal.Area = Sepal.Length * Sepal.Width)
```

```
##    Sepal.Length Sepal.Width Sepal.Length.SQRT Sepal.Area
## 1      2.258318         3.5          2.258318   7.904113
## 2      2.213594         3.0          2.213594   6.640783
## 3      2.167948         3.2          2.167948   6.937435
## 4      2.144761         3.1          2.144761   6.648759
## 5      2.236068         3.6          2.236068   8.049845
## 6      2.323790         3.9          2.323790   9.062781
## 7      2.144761         3.4          2.144761   7.292188
## 8      2.236068         3.4          2.236068   7.602631
## 9      2.097618         2.9          2.097618   6.083091
## 10     2.213594         3.1          2.213594   6.862143
## 11     2.323790         3.7          2.323790   8.598023
## 12     2.190890         3.4          2.190890   7.449027
## 13     2.190890         3.0          2.190890   6.572671
## 14     2.073644         3.0          2.073644   6.220932
## 15     2.408319         4.0          2.408319   9.633276
## 16     2.387467         4.4          2.387467  10.504856
## 17     2.323790         3.9          2.323790   9.062781
## 18     2.258318         3.5          2.258318   7.904113
## 19     2.387467         3.8          2.387467   9.072376
```

```
## 20       2.258318           3.8            2.258318   8.581608
## 21       2.323790           3.4            2.323790   7.900886
## 22       2.258318           3.7            2.258318   8.355776
## 23       2.144761           3.6            2.144761   7.721140
## 24       2.258318           3.3            2.258318   7.452449
## 25       2.190890           3.4            2.190890   7.449027
## 26       2.236068           3.0            2.236068   6.708204
## 27       2.236068           3.4            2.236068   7.602631
## 28       2.280351           3.5            2.280351   7.981228
## 29       2.280351           3.4            2.280351   7.753193
## 30       2.167948           3.2            2.167948   6.937435
## 31       2.190890           3.1            2.190890   6.791760
## 32       2.323790           3.4            2.323790   7.900886
## 33       2.280351           4.1            2.280351   9.349438
## 34       2.345208           4.2            2.345208   9.849873
## 35       2.213594           3.1            2.213594   6.862143
## 36       2.236068           3.2            2.236068   7.155418
## 37       2.345208           3.5            2.345208   8.208228
## 38       2.213594           3.1            2.213594   6.862143
## 39       2.097618           3.0            2.097618   6.292853
## 40       2.258318           3.4            2.258318   7.678281
## 41       2.236068           3.5            2.236068   7.826238
## 42       2.121320           2.3            2.121320   4.879037
## 43       2.097618           3.2            2.097618   6.712377
## 44       2.236068           3.5            2.236068   7.826238
## 45       2.258318           3.8            2.258318   8.581608
## 46       2.190890           3.0            2.190890   6.572671
## 47       2.258318           3.8            2.258318   8.581608
## 48       2.144761           3.2            2.144761   6.863235
## 49       2.302173           3.7            2.302173   8.518040
## 50       2.236068           3.3            2.236068   7.379024
## 51       2.645751           3.2            2.645751   8.466404
## 52       2.529822           3.2            2.529822   8.095431
## 53       2.626785           3.1            2.626785   8.143034
## 54       2.345208           2.3            2.345208   5.393978
## 55       2.549510           2.8            2.549510   7.138627
## 56       2.387467           2.8            2.387467   6.684908
## 57       2.509980           3.3            2.509980   8.282934
## 58       2.213594           2.4            2.213594   5.312626
## 59       2.569047           2.9            2.569047   7.450235
## 60       2.280351           2.7            2.280351   6.156947
## 61       2.236068           2.0            2.236068   4.472136
## 62       2.428992           3.0            2.428992   7.286975
## 63       2.449490           2.2            2.449490   5.388877
## 64       2.469818           2.9            2.469818   7.162472
## 65       2.366432           2.9            2.366432   6.862653
## 66       2.588436           3.1            2.588436   8.024151
## 67       2.366432           3.0            2.366432   7.099296
## 68       2.408319           2.7            2.408319   6.502461
## 69       2.489980           2.2            2.489980   5.477956
## 70       2.366432           2.5            2.366432   5.916080
## 71       2.428992           3.2            2.428992   7.772773
## 72       2.469818           2.8            2.469818   6.915490
## 73       2.509980           2.5            2.509980   6.274950
```

```
## 74      2.469818          2.8          2.469818  6.915490
## 75      2.529822          2.9          2.529822  7.336484
## 76      2.569047          3.0          2.569047  7.707140
## 77      2.607681          2.8          2.607681  7.301507
## 78      2.588436          3.0          2.588436  7.765307
## 79      2.449490          2.9          2.449490  7.103520
## 80      2.387467          2.6          2.387467  6.207415
## 81      2.345208          2.4          2.345208  5.628499
## 82      2.345208          2.4          2.345208  5.628499
## 83      2.408319          2.7          2.408319  6.502461
## 84      2.449490          2.7          2.449490  6.613622
## 85      2.323790          3.0          2.323790  6.971370
## 86      2.449490          3.4          2.449490  8.328265
## 87      2.588436          3.1          2.588436  8.024151
## 88      2.509980          2.3          2.509980  5.772954
## 89      2.366432          3.0          2.366432  7.099296
## 90      2.345208          2.5          2.345208  5.863020
## 91      2.345208          2.6          2.345208  6.097540
## 92      2.469818          3.0          2.469818  7.409453
## 93      2.408319          2.6          2.408319  6.261629
## 94      2.236068          2.3          2.236068  5.142956
## 95      2.366432          2.7          2.366432  6.389366
## 96      2.387467          3.0          2.387467  7.162402
## 97      2.387467          2.9          2.387467  6.923655
## 98      2.489980          2.9          2.489980  7.220942
## 99      2.258318          2.5          2.258318  5.645795
## 100     2.387467          2.8          2.387467  6.684908
## 101     2.509980          3.3          2.509980  8.282934
## 102     2.408319          2.7          2.408319  6.502461
## 103     2.664583          3.0          2.664583  7.993748
## 104     2.509980          2.9          2.509980  7.278942
## 105     2.549510          3.0          2.549510  7.648529
## 106     2.756810          3.0          2.756810  8.270429
## 107     2.213594          2.5          2.213594  5.533986
## 108     2.701851          2.9          2.701851  7.835369
## 109     2.588436          2.5          2.588436  6.471090
## 110     2.683282          3.6          2.683282  9.659814
## 111     2.549510          3.2          2.549510  8.158431
## 112     2.529822          2.7          2.529822  6.830520
## 113     2.607681          3.0          2.607681  7.823043
## 114     2.387467          2.5          2.387467  5.968668
## 115     2.408319          2.8          2.408319  6.743293
## 116     2.529822          3.2          2.529822  8.095431
## 117     2.549510          3.0          2.549510  7.648529
## 118     2.774887          3.8          2.774887 10.544572
## 119     2.774887          2.6          2.774887  7.214707
## 120     2.449490          2.2          2.449490  5.388877
## 121     2.626785          3.2          2.626785  8.405712
## 122     2.366432          2.8          2.366432  6.626009
## 123     2.774887          2.8          2.774887  7.769685
## 124     2.509980          2.7          2.509980  6.776946
## 125     2.588436          3.3          2.588436  8.541838
## 126     2.683282          3.2          2.683282  8.586501
## 127     2.489980          2.8          2.489980  6.971944
```

```
## 128     2.469818         3.0          2.469818   7.409453
## 129     2.529822         2.8          2.529822   7.083502
## 130     2.683282         3.0          2.683282   8.049845
## 131     2.720294         2.8          2.720294   7.616823
## 132     2.810694         3.8          2.810694  10.680637
## 133     2.529822         2.8          2.529822   7.083502
## 134     2.509980         2.8          2.509980   7.027944
## 135     2.469818         2.6          2.469818   6.421526
## 136     2.774887         3.0          2.774887   8.324662
## 137     2.509980         3.4          2.509980   8.533932
## 138     2.529822         3.1          2.529822   7.842449
## 139     2.449490         3.0          2.449490   7.348469
## 140     2.626785         3.1          2.626785   8.143034
## 141     2.588436         3.1          2.588436   8.024151
## 142     2.626785         3.1          2.626785   8.143034
## 143     2.408319         2.7          2.408319   6.502461
## 144     2.607681         3.2          2.607681   8.344579
## 145     2.588436         3.3          2.588436   8.541838
## 146     2.588436         3.0          2.588436   7.765307
## 147     2.509980         2.5          2.509980   6.274950
## 148     2.549510         3.0          2.549510   7.648529
## 149     2.489980         3.4          2.489980   8.465932
## 150     2.428992         3.0          2.428992   7.286975
```

```r
iris %>%
  select(ends_with("Length")) %>%
  mutate(Length.Diff = Sepal.Length - Petal.Length)
```

```
##     Sepal.Length Petal.Length Length.Diff
## 1       2.258318          1.4   0.8583180
## 2       2.213594          1.4   0.8135944
## 3       2.167948          1.3   0.8679483
## 4       2.144761          1.5   0.6447611
## 5       2.236068          1.4   0.8360680
## 6       2.323790          1.7   0.6237900
## 7       2.144761          1.4   0.7447611
## 8       2.236068          1.5   0.7360680
## 9       2.097618          1.4   0.6976177
## 10      2.213594          1.5   0.7135944
## 11      2.323790          1.5   0.8237900
## 12      2.190890          1.6   0.5908902
## 13      2.190890          1.4   0.7908902
## 14      2.073644          1.1   0.9736441
## 15      2.408319          1.2   1.2083189
## 16      2.387467          1.5   0.8874673
## 17      2.323790          1.3   1.0237900
## 18      2.258318          1.4   0.8583180
## 19      2.387467          1.7   0.6874673
## 20      2.258318          1.5   0.7583180
## 21      2.323790          1.7   0.6237900
## 22      2.258318          1.5   0.7583180
## 23      2.144761          1.0   1.1447611
## 24      2.258318          1.7   0.5583180
## 25      2.190890          1.9   0.2908902
```

```
## 26       2.236068          1.6   0.6360680
## 27       2.236068          1.6   0.6360680
## 28       2.280351          1.5   0.7803509
## 29       2.280351          1.4   0.8803509
## 30       2.167948          1.6   0.5679483
## 31       2.190890          1.6   0.5908902
## 32       2.323790          1.5   0.8237900
## 33       2.280351          1.5   0.7803509
## 34       2.345208          1.4   0.9452079
## 35       2.213594          1.5   0.7135944
## 36       2.236068          1.2   1.0360680
## 37       2.345208          1.3   1.0452079
## 38       2.213594          1.5   0.7135944
## 39       2.097618          1.3   0.7976177
## 40       2.258318          1.5   0.7583180
## 41       2.236068          1.3   0.9360680
## 42       2.121320          1.3   0.8213203
## 43       2.097618          1.3   0.7976177
## 44       2.236068          1.6   0.6360680
## 45       2.258318          1.9   0.3583180
## 46       2.190890          1.4   0.7908902
## 47       2.258318          1.6   0.6583180
## 48       2.144761          1.4   0.7447611
## 49       2.302173          1.5   0.8021729
## 50       2.236068          1.4   0.8360680
## 51       2.645751          4.7  -2.0542487
## 52       2.529822          4.5  -1.9701779
## 53       2.626785          4.9  -2.2732149
## 54       2.345208          4.0  -1.6547921
## 55       2.549510          4.6  -2.0504902
## 56       2.387467          4.5  -2.1125327
## 57       2.509980          4.7  -2.1900199
## 58       2.213594          3.3  -1.0864056
## 59       2.569047          4.6  -2.0309535
## 60       2.280351          3.9  -1.6196491
## 61       2.236068          3.5  -1.2639320
## 62       2.428992          4.2  -1.7710084
## 63       2.449490          4.0  -1.5505103
## 64       2.469818          4.7  -2.2301822
## 65       2.366432          3.6  -1.2335681
## 66       2.588436          4.4  -1.8115642
## 67       2.366432          4.5  -2.1335681
## 68       2.408319          4.1  -1.6916811
## 69       2.489980          4.5  -2.0100201
## 70       2.366432          3.9  -1.5335681
## 71       2.428992          4.8  -2.3710084
## 72       2.469818          4.0  -1.5301822
## 73       2.509980          4.9  -2.3900199
## 74       2.469818          4.7  -2.2301822
## 75       2.529822          4.3  -1.7701779
## 76       2.569047          4.4  -1.8309535
## 77       2.607681          4.8  -2.1923190
## 78       2.588436          5.0  -2.4115642
## 79       2.449490          4.5  -2.0505103
```

```
## 80     2.387467           3.5  -1.1125327
## 81     2.345208           3.8  -1.4547921
## 82     2.345208           3.7  -1.3547921
## 83     2.408319           3.9  -1.4916811
## 84     2.449490           5.1  -2.6505103
## 85     2.323790           4.5  -2.1762100
## 86     2.449490           4.5  -2.0505103
## 87     2.588436           4.7  -2.1115642
## 88     2.509980           4.4  -1.8900199
## 89     2.366432           4.1  -1.7335681
## 90     2.345208           4.0  -1.6547921
## 91     2.345208           4.4  -2.0547921
## 92     2.469818           4.6  -2.1301822
## 93     2.408319           4.0  -1.5916811
## 94     2.236068           3.3  -1.0639320
## 95     2.366432           4.2  -1.8335681
## 96     2.387467           4.2  -1.8125327
## 97     2.387467           4.2  -1.8125327
## 98     2.489980           4.3  -1.8100201
## 99     2.258318           3.0  -0.7416820
## 100    2.387467           4.1  -1.7125327
## 101    2.509980           6.0  -3.4900199
## 102    2.408319           5.1  -2.6916811
## 103    2.664583           5.9  -3.2354175
## 104    2.509980           5.6  -3.0900199
## 105    2.549510           5.8  -3.2504902
## 106    2.756810           6.6  -3.8431902
## 107    2.213594           4.5  -2.2864056
## 108    2.701851           6.3  -3.5981488
## 109    2.588436           5.8  -3.2115642
## 110    2.683282           6.1  -3.4167184
## 111    2.549510           5.1  -2.5504902
## 112    2.529822           5.3  -2.7701779
## 113    2.607681           5.5  -2.8923190
## 114    2.387467           5.0  -2.6125327
## 115    2.408319           5.1  -2.6916811
## 116    2.529822           5.3  -2.7701779
## 117    2.549510           5.5  -2.9504902
## 118    2.774887           6.7  -3.9251126
## 119    2.774887           6.9  -4.1251126
## 120    2.449490           5.0  -2.5505103
## 121    2.626785           5.7  -3.0732149
## 122    2.366432           4.9  -2.5335681
## 123    2.774887           6.7  -3.9251126
## 124    2.509980           4.9  -2.3900199
## 125    2.588436           5.7  -3.1115642
## 126    2.683282           6.0  -3.3167184
## 127    2.489980           4.8  -2.3100201
## 128    2.469818           4.9  -2.4301822
## 129    2.529822           5.6  -3.0701779
## 130    2.683282           5.8  -3.1167184
## 131    2.720294           6.1  -3.3797059
## 132    2.810694           6.4  -3.5893061
## 133    2.529822           5.6  -3.0701779
```

```
## 134     2.509980          5.1  -2.5900199
## 135     2.469818          5.6  -3.1301822
## 136     2.774887          6.1  -3.3251126
## 137     2.509980          5.6  -3.0900199
## 138     2.529822          5.5  -2.9701779
## 139     2.449490          4.8  -2.3505103
## 140     2.626785          5.4  -2.7732149
## 141     2.588436          5.6  -3.0115642
## 142     2.626785          5.1  -2.4732149
## 143     2.408319          5.1  -2.6916811
## 144     2.607681          5.9  -3.2923190
## 145     2.588436          5.7  -3.1115642
## 146     2.588436          5.2  -2.6115642
## 147     2.509980          5.0  -2.4900199
## 148     2.549510          5.2  -2.6504902
## 149     2.489980          5.4  -2.9100201
## 150     2.428992          5.1  -2.6710084
```

```
iris %>%
  select(ends_with("Length"), Species) %>%
  rowwise() %>%
  mutate(Length.Diff = Sepal.Length - Petal.Length)
```

```
## # A tibble: 150 x 4
## # Rowwise:
##    Sepal.Length Petal.Length Species     Length.Diff
##           <dbl>        <dbl> <chr>             <dbl>
##  1         2.26          1.4 Iris-setosa       0.858
##  2         2.21          1.4 Iris-setosa       0.814
##  3         2.17          1.3 Iris-setosa       0.868
##  4         2.14          1.5 Iris-setosa       0.645
##  5         2.24          1.4 Iris-setosa       0.836
##  6         2.32          1.7 Iris-setosa       0.624
##  7         2.14          1.4 Iris-setosa       0.745
##  8         2.24          1.5 Iris-setosa       0.736
##  9         2.10          1.4 Iris-setosa       0.698
## 10         2.21          1.5 Iris-setosa       0.714
## # i 140 more rows
```

```
iris %>%
  select(contains("Sepal"), Species) %>%
  transmute(Sepal.Area = Sepal.Length * Sepal.Width)
```

```
##     Sepal.Area
## 1     7.904113
## 2     6.640783
## 3     6.937435
## 4     6.648759
## 5     8.049845
## 6     9.062781
## 7     7.292188
## 8     7.602631
## 9     6.083091
```

```
## 10    6.862143
## 11    8.598023
## 12    7.449027
## 13    6.572671
## 14    6.220932
## 15    9.633276
## 16   10.504856
## 17    9.062781
## 18    7.904113
## 19    9.072376
## 20    8.581608
## 21    7.900886
## 22    8.355776
## 23    7.721140
## 24    7.452449
## 25    7.449027
## 26    6.708204
## 27    7.602631
## 28    7.981228
## 29    7.753193
## 30    6.937435
## 31    6.791760
## 32    7.900886
## 33    9.349438
## 34    9.849873
## 35    6.862143
## 36    7.155418
## 37    8.208228
## 38    6.862143
## 39    6.292853
## 40    7.678281
## 41    7.826238
## 42    4.879037
## 43    6.712377
## 44    7.826238
## 45    8.581608
## 46    6.572671
## 47    8.581608
## 48    6.863235
## 49    8.518040
## 50    7.379024
## 51    8.466404
## 52    8.095431
## 53    8.143034
## 54    5.393978
## 55    7.138627
## 56    6.684908
## 57    8.282934
## 58    5.312626
## 59    7.450235
## 60    6.156947
## 61    4.472136
## 62    7.286975
## 63    5.388877
```

```
## 64    7.162472
## 65    6.862653
## 66    8.024151
## 67    7.099296
## 68    6.502461
## 69    5.477956
## 70    5.916080
## 71    7.772773
## 72    6.915490
## 73    6.274950
## 74    6.915490
## 75    7.336484
## 76    7.707140
## 77    7.301507
## 78    7.765307
## 79    7.103520
## 80    6.207415
## 81    5.628499
## 82    5.628499
## 83    6.502461
## 84    6.613622
## 85    6.971370
## 86    8.328265
## 87    8.024151
## 88    5.772954
## 89    7.099296
## 90    5.863020
## 91    6.097540
## 92    7.409453
## 93    6.261629
## 94    5.142956
## 95    6.389366
## 96    7.162402
## 97    6.923655
## 98    7.220942
## 99    5.645795
## 100   6.684908
## 101   8.282934
## 102   6.502461
## 103   7.993748
## 104   7.278942
## 105   7.648529
## 106   8.270429
## 107   5.533986
## 108   7.835369
## 109   6.471090
## 110   9.659814
## 111   8.158431
## 112   6.830520
## 113   7.823043
## 114   5.968668
## 115   6.743293
## 116   8.095431
## 117   7.648529
```

```
## 118  10.544572
## 119   7.214707
## 120   5.388877
## 121   8.405712
## 122   6.626009
## 123   7.769685
## 124   6.776946
## 125   8.541838
## 126   8.586501
## 127   6.971944
## 128   7.409453
## 129   7.083502
## 130   8.049845
## 131   7.616823
## 132  10.680637
## 133   7.083502
## 134   7.027944
## 135   6.421526
## 136   8.324662
## 137   8.533932
## 138   7.842449
## 139   7.348469
## 140   8.143034
## 141   8.024151
## 142   8.143034
## 143   6.502461
## 144   8.344579
## 145   8.541838
## 146   7.765307
## 147   6.274950
## 148   7.648529
## 149   8.465932
## 150   7.286975
```

We must use R markdown syntax R markdown with knitr and kable,

```
knitr::kable(head(mtcars), digits = 2, align = c(rep("l", 4), rep("c", 4), rep("r", 4)))
```

	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
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

```
packages_to_install <- c("xtable")

for (package_name in packages_to_install) {
  if (!requireNamespace(package_name, quietly = TRUE)) {
```

```r
    install.packages(package_name)
  }
}

library(xtable)
```

```r
print(xtable(head(mtcars)), type = "html")
```

```
## <!-- html table generated in R 4.3.3 by xtable 1.8-4 package -->
## <!-- Sun Apr 14 12:34:24 2024 -->
## <table border=1>
## <tr> <th>  </th> <th> mpg </th> <th> cyl </th> <th> disp </th> <th> hp </th> <th> drat </th> <th> wt
##   <tr> <td align="right"> Mazda RX4 </td> <td align="right"> 21.00 </td> <td align="right"> 6.00 </td
##   <tr> <td align="right"> Mazda RX4 Wag </td> <td align="right"> 21.00 </td> <td align="right"> 6.00
##   <tr> <td align="right"> Datsun 710 </td> <td align="right"> 22.80 </td> <td align="right"> 4.00 </
##   <tr> <td align="right"> Hornet 4 Drive </td> <td align="right"> 21.40 </td> <td align="right"> 6.0
##   <tr> <td align="right"> Hornet Sportabout </td> <td align="right"> 18.70 </td> <td align="right"> 8
##   <tr> <td align="right"> Valiant </td> <td align="right"> 18.10 </td> <td align="right"> 6.00 </td>
##    </table>
```