Assignment3

Dipesh Poudel

9/23/2021

A3 - Vectorization, Functions, Pipes, Missing Values and Markdown in R

Instructions

Replicate all the examples/exercises used in the class slides of session 4 and session 5 with interpretations

Vector Opration in R

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

** Adding a and b vectors

```
a+b
```

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

When we give `a+b` command it performs element wise sum between two vectors. The first element of vector a is added with first element of vector b and so on.

What will happen when you type these two codes in R/R studio and run: • y <- seq(1, 10, length.out = 5) • (y <- seq(1, 10, length.out = 5))

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

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

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

When we run the first command the the result is stored in variable y and in in second case the result is also printed.

Vector Operation Multiplication

a <- 1:10 b <- 5 #Vector length = 10 #Vector length = 1 • What will happen if we type: a * b in R console?

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

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

5 i.e `b` is multiplied with each element of vector.

Define: • a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] #Check class of 'a' • b = [5] #Check class of 'b' • c = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11] #Check class of 'c'

```r
a<-c(1:10)
class(a)
```

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

```r
b<-c(5)
class(b)
```

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

```r
c<-c(2:11)
class(c)
```

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

```r
ac<-cbind(a,c)
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
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
```

- Get:
- d = arithmetic mean by cases (row means) #Hint: Use "apply" function
- acbd = add new variable d in the acb matrix #Check class of 'acbd' object
- Summarize all the variables of 'acbd' object and interpret the results carefully!

```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
```

The apply function applies a given function to the entire row or column which is controlled by `MARGIN` parameter. This method is faster than loop.

```
# Adding New Variable in the Matrix
acbd<-cbind(acb,d)
print(acbd)
```

```
##        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
```

```
class(acbd)
```

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

We added a the vector as a new column to a matrix which resulted in a new matrix with three columns.

```
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
```

The summary function gives us the statistical summary of each of the column. The summary contains the minimum and maximum value, mean and median along with 1st and 3rd quartile.

Vector Recycling

```
# Creating two vectors a and b
a<-1:10
b<-1:5

# Adding these two vectors
(a+b)
```

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

The vector `a` has 10 elements and vector `b` has 5 element. When we added these vector the result is of same length of a i.e 10 elements. The element of the smaller vector is repeated until the operation results in the size of larger vector.

If the length of the larger vector(one with more number of elements) is not the multiple of the smaller vector then we get warning. Let us try with an example

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

# Adding a and b
```

```
(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
```

Here we got a warning saying that "longer object length is not a multiple of shorter object length". However the calculation was carried out up to the point where length of result was equal to the length of larger vector.

Function in R

An example of a user defined function is shown below

```
best_practice <- c("Let", "the", "computer", "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] "computer"
## [1] "do"
## [1] "the"
## [1] "work"
```

In this approach, inside the function we are repeating same step `print(sentence[i])` where `i` is the index. It works for the example we have given but for a vector is smaller number of elements we will get `NA` and for a vector with larger number element we will miss the element

```
# Vector with 5 Elements
best_practice <- c("Let", "the", "computer", "do", "work")
print_words(best_practice)
```

```
## [1] "Let"
## [1] "the"
## [1] "computer"
## [1] "do"
## [1] "work"
## [1] NA
```

In this example we got `NA` because there was no 6th element.

```
best_practice <- c("Let", "the", "computer", "do", "the", "work","properly")
print_words(best_practice)
```

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

```
## [1] "work"
```

In this example we did not get the last element.

Deleting element from a vector

We can use negative index to remove element from vector

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

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

```
best_practice[-6]
```

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

Improving the function with Loop

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

print_words(best_practice)
```

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

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

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

Conditional Statements

In conditional statement we check the condition and execute the corresponding statement when the condition is true.

```
y<-10
if (y < 20) {
x <- "Too low"
} else {
x <- "Too high"
}
print(x)
```

```
## [1] "Too low"
```

In this example y is less than 20 so the value of x became "Too low". Now let us assign y greater than 20 and run the same code.

```
y<-50
if (y < 20) {
x <- "Too low"
} else {
x <- "Too high"
}
print(x)
```

```
## [1] "Too high"
```

Here we got too high because the value of y is 50 which is not smaller than 20 so statement in else block was executed.

Multiple Conditions: If, else if, else

```
temp=5
if (temp <= 0) {
"freezing"
} else if (temp <= 10) {
"cold"
} else if (temp <= 20) {
"cool"
} else if (temp <= 30) {
"warm"
} else {
"hot"
}
```

```
## [1] "cold"
```

In this example, 5 is assigned to temp. We have multiple condition so we are using if else-if else condition. After the first matching condition is met the corresponding code block is executed which resulted in we getting "cold".

Pipe Operators

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

#Compute logarithm of "x"
#Return lagged differences
#compute the exponential fuction and round the result with one decimal

round(exp(diff(log(x))),1)
```

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

Using pipe operation to computer the same operation

```
library(magrittr)
x %>% log() %>%
  diff() %>%
```

```
  exp() %>%
  round(1)
```

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

In first case we used nested function which can be difficult to track when the number of function increases. In second case we have used pipe operation which makes code much more readable and track able as well.

Different Pipe Operators

```
x <- rnorm(100)
print(x)
```

```
##   [1]  0.31230111 -0.53878446  0.20992771 -2.27809488 -0.41098012  0.58244615
##   [7]  0.36860892 -0.14238946  1.09809798  0.39810071 -0.70314585  0.20164288
##  [13]  0.14785974 -0.77509765 -0.98428728  1.02613174 -1.06429690 -0.40576295
##  [19] -1.12348277 -0.07509676 -2.20915416 -0.92254228 -0.54767972 -0.15229497
##  [25]  0.01740217  1.22894973 -0.93897647 -1.27709245  1.06878002 -2.92390497
##  [31] -0.29049071 -0.42136204  1.03900829 -0.12365462  1.02473249 -1.48059040
##  [37] -0.76147224  1.06855225 -0.40958046 -0.26573089 -0.95535213 -1.43133922
##  [43] -0.17375660 -0.33928177  0.35926144 -0.50066074 -0.14729920  1.00414105
##  [49] -1.46895024  1.67279936  2.04151962 -0.11487182 -1.20224492 -0.43164041
##  [55]  0.46686559  2.36976576 -0.48828186  1.09328724 -0.40338125 -1.46684741
##  [61] -0.10965023 -1.10103581  0.18706669 -0.79596829 -0.88173808  0.72102899
##  [67]  0.13525065  1.28933003  0.40299628  1.78506165 -0.32169287 -0.45923427
##  [73]  1.33352279 -0.25353838  1.30705948 -0.19237032  0.71046270 -1.01909206
##  [79]  0.78843257 -0.28670009  1.15617762 -0.38692429  1.09278196  2.21892824
##  [85] -1.12755835  0.64156099  0.84024259 -0.92631603  0.47982998  0.13303040
##  [91] -0.49479177  0.65489434 -0.31905291  1.58054910 -0.92043169  1.09317966
##  [97]  0.93167791 -1.76145730 -0.99937143 -0.01923773
```

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

```
##   [1] 0.01740217 0.01923773 0.07509676 0.10965023 0.11487182 0.12365462
##   [7] 0.13303040 0.13525065 0.14238946 0.14729920 0.14785974 0.15229497
##  [13] 0.17375660 0.18706669 0.19237032 0.20164288 0.20992771 0.25353838
##  [19] 0.26573089 0.28670009 0.29049071 0.31230111 0.31905291 0.32169287
##  [25] 0.33928177 0.35926144 0.36860892 0.38692429 0.39810071 0.40299628
##  [31] 0.40338125 0.40576295 0.40958046 0.41098012 0.42136204 0.43164041
##  [37] 0.45923427 0.46686559 0.47982998 0.48828186 0.49479177 0.50066074
##  [43] 0.53878446 0.54767972 0.58244615 0.64156099 0.65489434 0.70314585
##  [49] 0.71046270 0.72102899 0.76147224 0.77509765 0.78843257 0.79596829
##  [55] 0.84024259 0.88173808 0.92043169 0.92254228 0.92631603 0.93167791
##  [61] 0.93897647 0.95535213 0.98428728 0.99937143 1.00414105 1.01909206
##  [67] 1.02473249 1.02613174 1.03900829 1.06429690 1.06855225 1.06878002
##  [73] 1.09278196 1.09317966 1.09328724 1.09809798 1.10103581 1.12348277
##  [79] 1.12755835 1.15617762 1.20224492 1.22894973 1.27709245 1.28933003
##  [85] 1.30705948 1.33352279 1.43133922 1.46684741 1.46895024 1.48059040
##  [91] 1.58054910 1.67279936 1.76145730 1.78506165 2.04151962 2.20915416
##  [97] 2.21892824 2.27809488 2.36976576 2.92390497
```

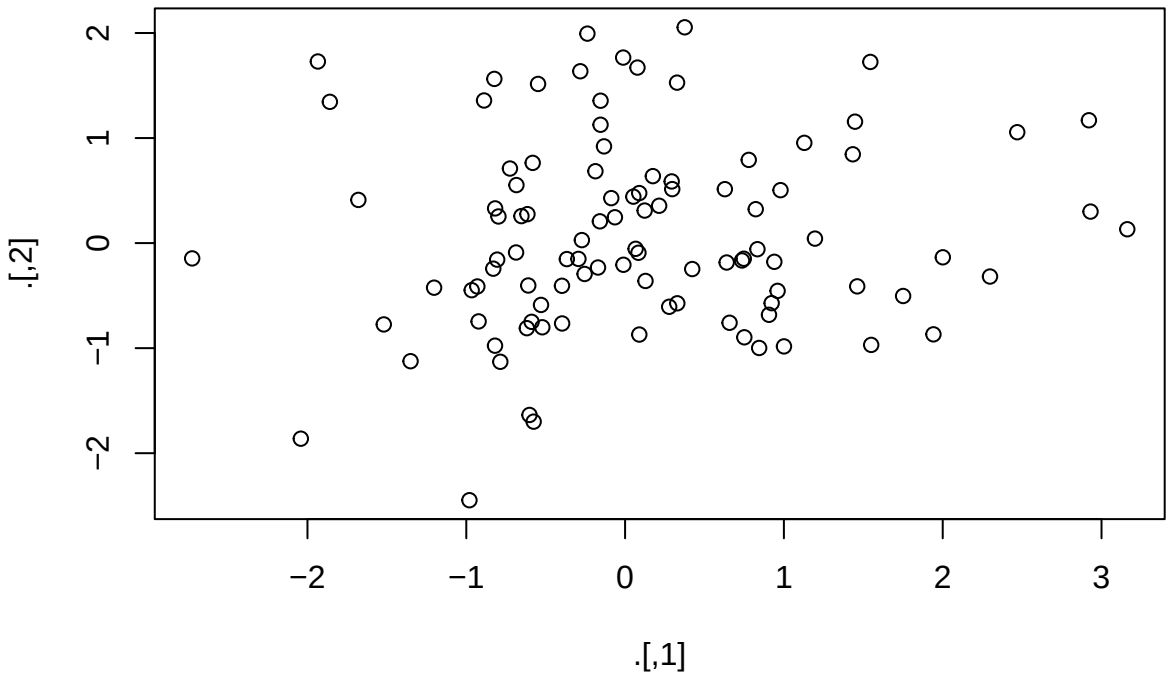
```
print(x)
```

```
##   [1] 0.01740217 0.01923773 0.07509676 0.10965023 0.11487182 0.12365462
##   [7] 0.13303040 0.13525065 0.14238946 0.14729920 0.14785974 0.15229497
##  [13] 0.17375660 0.18706669 0.19237032 0.20164288 0.20992771 0.25353838
```

```
##  [19] 0.26573089 0.28670009 0.29049071 0.31230111 0.31905291 0.32169287
##  [25] 0.33928177 0.35926144 0.36860892 0.38692429 0.39810071 0.40299628
##  [31] 0.40338125 0.40576295 0.40958046 0.41098012 0.42136204 0.43164041
##  [37] 0.45923427 0.46686559 0.47982998 0.48828186 0.49479177 0.50066074
##  [43] 0.53878446 0.54767972 0.58244615 0.64156099 0.65489434 0.70314585
##  [49] 0.71046270 0.72102899 0.76147224 0.77509765 0.78843257 0.79596829
##  [55] 0.84024259 0.88173808 0.92043169 0.92254228 0.92631603 0.93167791
##  [61] 0.93897647 0.95535213 0.98428728 0.99937143 1.00414105 1.01909206
##  [67] 1.02473249 1.02613174 1.03900829 1.06429690 1.06855225 1.06878002
##  [73] 1.09278196 1.09317966 1.09328724 1.09809798 1.10103581 1.12348277
##  [79] 1.12755835 1.15617762 1.20224492 1.22894973 1.27709245 1.28933003
##  [85] 1.30705948 1.33352279 1.43133922 1.46684741 1.46895024 1.48059040
##  [91] 1.58054910 1.67279936 1.76145730 1.78506165 2.04151962 2.20915416
##  [97] 2.21892824 2.27809488 2.36976576 2.92390497
```

In the code above, we created 100 random numbers with normal distribution and stored in variable `x`. Then, we applied pipe operator `<%>` to apply the functions store the result in x variable.

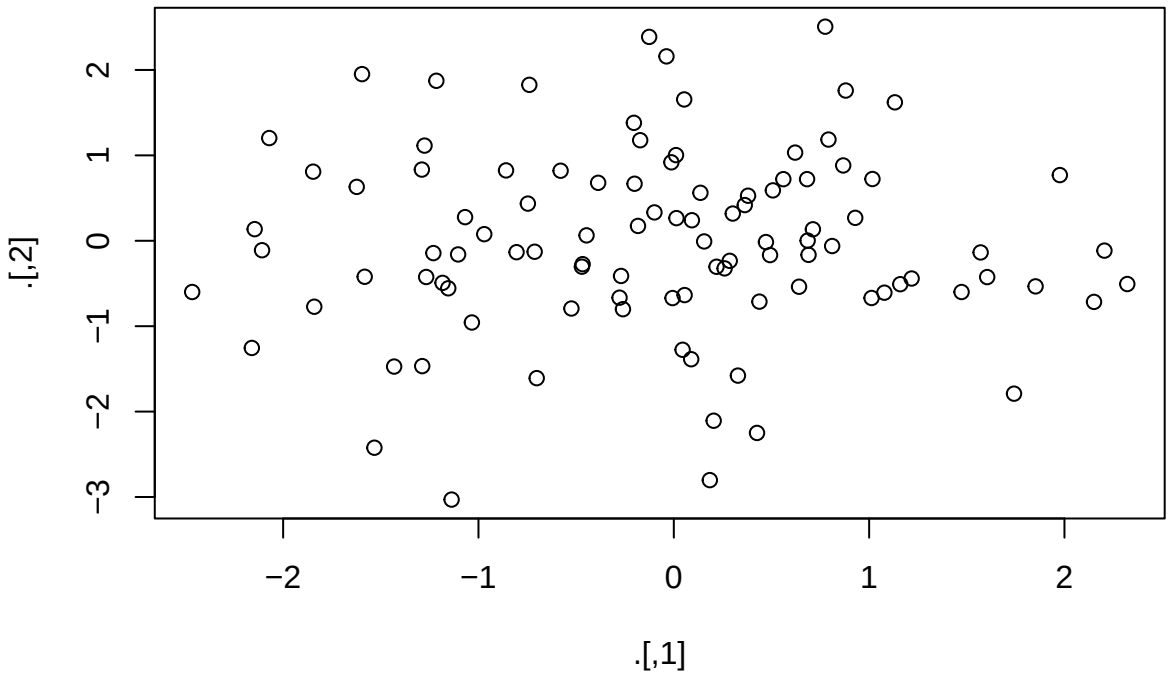
```
#Other pipe operators 2
rnorm(200) %>%
  matrix(ncol = 2) %T>%
  plot %>%
  colSums
```




```
## [1] 8.952068 7.578969
```

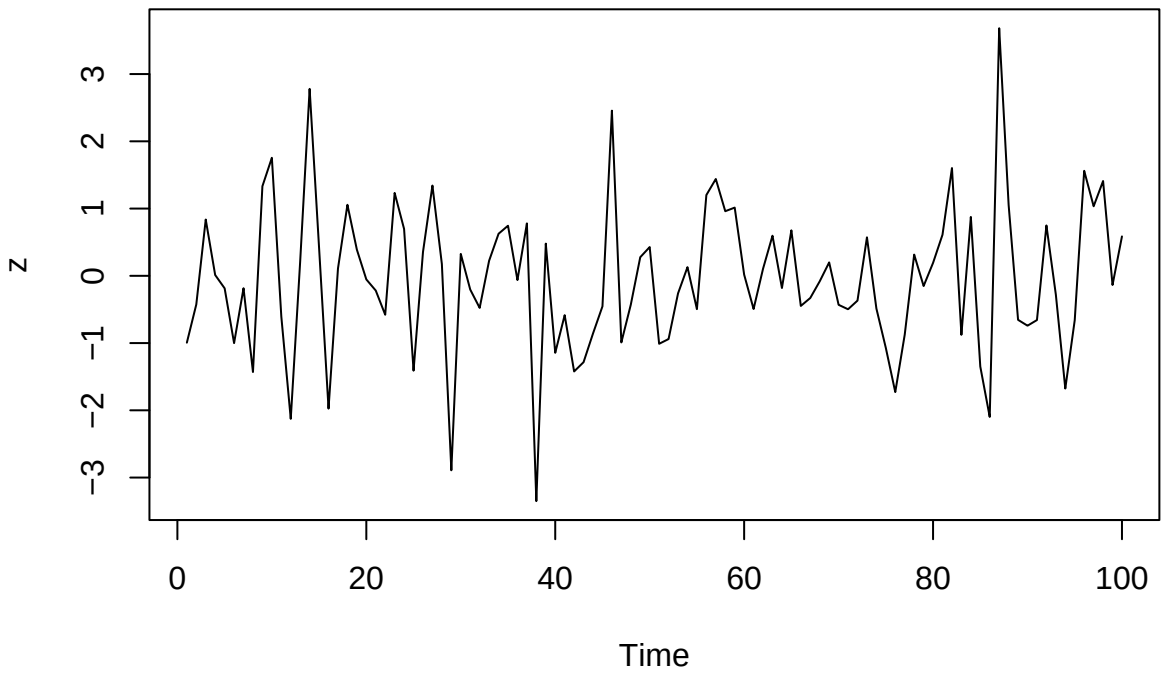
In above example we passed the matrix obtained from the prior operation to the next function using `T` pipe operator.

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




```
## [1] -8.243729 -1.039868
```

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



Here we have a time series plot created using 100 random values that follows normal distribution. We have used exposition operator to pass data to time series plot function.

```
#Load the package, install if require!
# install.packages("babynames")
library(babynames)
```

```
library(dplyr)
```

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

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

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

```
#Load the data:
data(babynames)
```

```
# Count how many young boys with the name "Taylor" are born
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
```

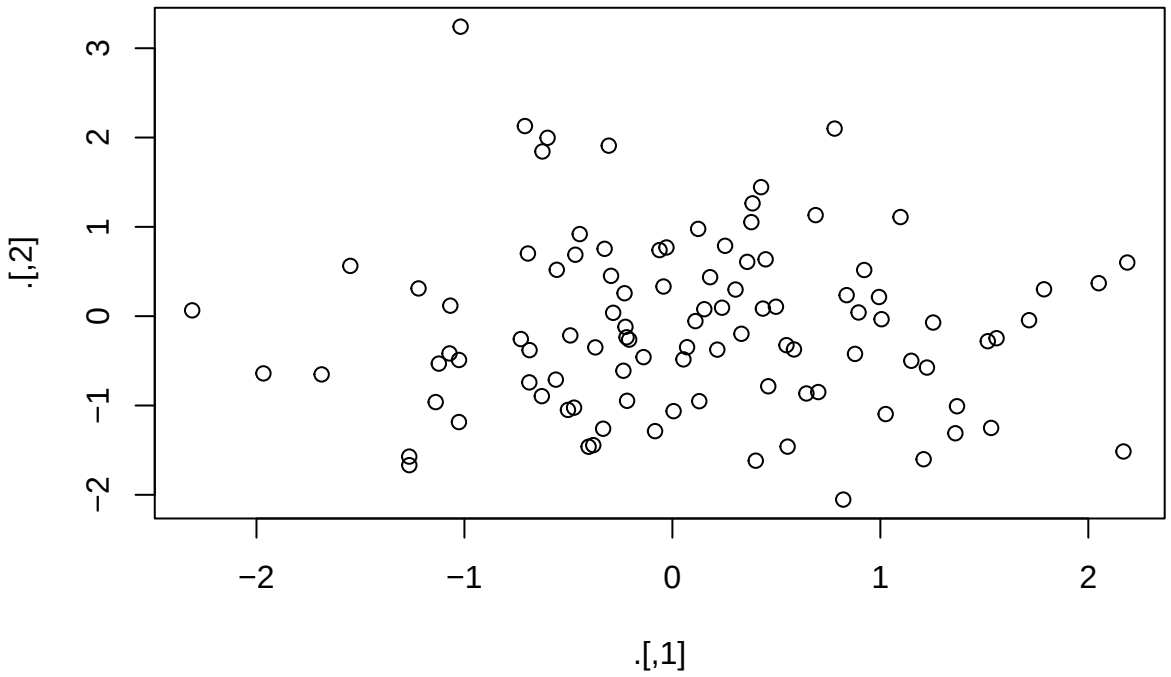
In the example above we counted the number of male babies with name "Taylor" using tested function and pipe operator method. We used `filter` function from `dplyr` package for filtering the data based on the given condition.

```
#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()
```

Here we got the data from web and changed the column names. Then we added a new column `Sepal.Length.SQRT` which is square root of sepal length.

```
#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
```

Here the `Tee` operator helps to pass the data to the next arguement i.e plot and colSums.

The "dplyr" package

With "group_by" function of "base" but Without "dplyr" package and pipe operators

```
# Install package if not installed
# install.packages("hflights")
library(hflights)
#Without pipe operators:
grouped_flights <- group_by(hflights, Year, Month, DayofMonth)
# Using Seect from dplyr
```

The `hflights` contains the dataset of all flights departing from Houston airports IAH (George Bush Intercontinental) and HOU (Houston Hobby).

We grouped the data by Year, Month and Day of the Month and assigned the result to `grouped_flights`.

```
flights_data <- select(grouped_flights, Year:DayofMonth, ArrDelay, DepDelay)
```

Using the dplyr package's select we selected the some columns and created a new dataframe.

```
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.
```

We took mean of ArrDelay, DepDelay removing the missing data and then overiding the original variable with the new one

```
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
```

In the final result we have filtered the data whose average arrival and departure delay is greater than 30

Same operation as above 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 >=2.0) %>%
  arrange(Sepal.Length)     #Sort data in ascending order
```

```
##     Sepal.Length Sepal.Width Sepal.Length.SQRT
## 1       2.073644         3.0          2.073644
```

```
## 2       2.097618        2.9         2.097618
## 3       2.097618        3.0         2.097618
## 4       2.097618        3.2         2.097618
## 5       2.121320        2.3         2.121320
## 6       2.144761        3.1         2.144761
## 7       2.144761        3.4         2.144761
## 8       2.144761        3.6         2.144761
## 9       2.144761        3.2         2.144761
## 10      2.167948        3.2         2.167948
## 11      2.167948        3.2         2.167948
## 12      2.190890        3.4         2.190890
## 13      2.190890        3.0         2.190890
## 14      2.190890        3.4         2.190890
## 15      2.190890        3.1         2.190890
## 16      2.190890        3.0         2.190890
## 17      2.213594        3.0         2.213594
## 18      2.213594        3.1         2.213594
## 19      2.213594        3.1         2.213594
## 20      2.213594        3.1         2.213594
## 21      2.213594        2.4         2.213594
## 22      2.213594        2.5         2.213594
## 23      2.236068        3.6         2.236068
## 24      2.236068        3.4         2.236068
## 25      2.236068        3.0         2.236068
## 26      2.236068        3.4         2.236068
## 27      2.236068        3.2         2.236068
## 28      2.236068        3.5         2.236068
## 29      2.236068        3.5         2.236068
## 30      2.236068        3.3         2.236068
## 31      2.236068        2.0         2.236068
## 32      2.236068        2.3         2.236068
## 33      2.258318        3.5         2.258318
## 34      2.258318        3.5         2.258318
## 35      2.258318        3.8         2.258318
## 36      2.258318        3.7         2.258318
## 37      2.258318        3.3         2.258318
## 38      2.258318        3.4         2.258318
## 39      2.258318        3.8         2.258318
## 40      2.258318        3.8         2.258318
## 41      2.258318        2.5         2.258318
## 42      2.280351        3.5         2.280351
## 43      2.280351        3.4         2.280351
## 44      2.280351        4.1         2.280351
## 45      2.280351        2.7         2.280351
## 46      2.302173        3.7         2.302173
## 47      2.323790        3.9         2.323790
## 48      2.323790        3.7         2.323790
## 49      2.323790        3.9         2.323790
## 50      2.323790        3.4         2.323790
## 51      2.323790        3.4         2.323790
## 52      2.323790        3.0         2.323790
## 53      2.345208        4.2         2.345208
## 54      2.345208        3.5         2.345208
## 55      2.345208        2.3         2.345208
```

```
## 56      2.345208        2.4         2.345208
## 57      2.345208        2.4         2.345208
## 58      2.345208        2.5         2.345208
## 59      2.345208        2.6         2.345208
## 60      2.366432        2.9         2.366432
## 61      2.366432        3.0         2.366432
## 62      2.366432        2.5         2.366432
## 63      2.366432        3.0         2.366432
## 64      2.366432        2.7         2.366432
## 65      2.366432        2.8         2.366432
## 66      2.387467        4.4         2.387467
## 67      2.387467        3.8         2.387467
## 68      2.387467        2.8         2.387467
## 69      2.387467        2.6         2.387467
## 70      2.387467        3.0         2.387467
## 71      2.387467        2.9         2.387467
## 72      2.387467        2.8         2.387467
## 73      2.387467        2.5         2.387467
## 74      2.408319        4.0         2.408319
## 75      2.408319        2.7         2.408319
## 76      2.408319        2.7         2.408319
## 77      2.408319        2.6         2.408319
## 78      2.408319        2.7         2.408319
## 79      2.408319        2.8         2.408319
## 80      2.408319        2.7         2.408319
## 81      2.428992        3.0         2.428992
## 82      2.428992        3.2         2.428992
## 83      2.428992        3.0         2.428992
## 84      2.449490        2.2         2.449490
## 85      2.449490        2.9         2.449490
## 86      2.449490        2.7         2.449490
## 87      2.449490        3.4         2.449490
## 88      2.449490        2.2         2.449490
## 89      2.449490        3.0         2.449490
## 90      2.469818        2.9         2.469818
## 91      2.469818        2.8         2.469818
## 92      2.469818        2.8         2.469818
## 93      2.469818        3.0         2.469818
## 94      2.469818        3.0         2.469818
## 95      2.469818        2.6         2.469818
## 96      2.489980        2.2         2.489980
## 97      2.489980        2.9         2.489980
## 98      2.489980        2.8         2.489980
## 99      2.489980        3.4         2.489980
## 100     2.509980        3.3         2.509980
## 101     2.509980        2.5         2.509980
## 102     2.509980        2.3         2.509980
## 103     2.509980        3.3         2.509980
## 104     2.509980        2.9         2.509980
## 105     2.509980        2.7         2.509980
## 106     2.509980        2.8         2.509980
## 107     2.509980        3.4         2.509980
## 108     2.509980        2.5         2.509980
## 109     2.529822        3.2         2.529822
```

```
## 110     2.529822         2.9          2.529822
## 111     2.529822         2.7          2.529822
## 112     2.529822         3.2          2.529822
## 113     2.529822         2.8          2.529822
## 114     2.529822         2.8          2.529822
## 115     2.529822         3.1          2.529822
## 116     2.549510         2.8          2.549510
## 117     2.549510         3.0          2.549510
## 118     2.549510         3.2          2.549510
## 119     2.549510         3.0          2.549510
## 120     2.549510         3.0          2.549510
## 121     2.569047         2.9          2.569047
## 122     2.569047         3.0          2.569047
## 123     2.588436         3.1          2.588436
## 124     2.588436         3.0          2.588436
## 125     2.588436         3.1          2.588436
## 126     2.588436         2.5          2.588436
## 127     2.588436         3.3          2.588436
## 128     2.588436         3.1          2.588436
## 129     2.588436         3.3          2.588436
## 130     2.588436         3.0          2.588436
## 131     2.607681         2.8          2.607681
## 132     2.607681         3.0          2.607681
## 133     2.607681         3.2          2.607681
## 134     2.626785         3.1          2.626785
## 135     2.626785         3.2          2.626785
## 136     2.626785         3.1          2.626785
## 137     2.626785         3.1          2.626785
## 138     2.645751         3.2          2.645751
## 139     2.664583         3.0          2.664583
## 140     2.683282         3.6          2.683282
## 141     2.683282         3.2          2.683282
## 142     2.683282         3.0          2.683282
## 143     2.701851         2.9          2.701851
## 144     2.720294         2.8          2.720294
## 145     2.756810         3.0          2.756810
## 146     2.774887         3.8          2.774887
## 147     2.774887         2.6          2.774887
## 148     2.774887         2.8          2.774887
## 149     2.774887         3.0          2.774887
## 150     2.810694         3.8          2.810694
```

In above code, we used the arrange and filter function from dplyr to filter the data with sepal length greater or equal to 2.5 and sorted the data by sepal length in ascending order.

```
#Descending order:
iris %>%
  select(starts_with("Sepal")) %>%
  filter(Sepal.Length >=2.5) %>%
  arrange(desc(Sepal.Length))       #Sort data in descending order
```

```
##    Sepal.Length Sepal.Width Sepal.Length.SQRT
## 1      2.810694         3.8          2.810694
## 2      2.774887         3.8          2.774887
## 3      2.774887         2.6          2.774887
```

```
## 4      2.774887         2.8          2.774887
## 5      2.774887         3.0          2.774887
## 6      2.756810         3.0          2.756810
## 7      2.720294         2.8          2.720294
## 8      2.701851         2.9          2.701851
## 9      2.683282         3.6          2.683282
## 10     2.683282         3.2          2.683282
## 11     2.683282         3.0          2.683282
## 12     2.664583         3.0          2.664583
## 13     2.645751         3.2          2.645751
## 14     2.626785         3.1          2.626785
## 15     2.626785         3.2          2.626785
## 16     2.626785         3.1          2.626785
## 17     2.626785         3.1          2.626785
## 18     2.607681         2.8          2.607681
## 19     2.607681         3.0          2.607681
## 20     2.607681         3.2          2.607681
## 21     2.588436         3.1          2.588436
## 22     2.588436         3.0          2.588436
## 23     2.588436         3.1          2.588436
## 24     2.588436         2.5          2.588436
## 25     2.588436         3.3          2.588436
## 26     2.588436         3.1          2.588436
## 27     2.588436         3.3          2.588436
## 28     2.588436         3.0          2.588436
## 29     2.569047         2.9          2.569047
## 30     2.569047         3.0          2.569047
## 31     2.549510         2.8          2.549510
## 32     2.549510         3.0          2.549510
## 33     2.549510         3.2          2.549510
## 34     2.549510         3.0          2.549510
## 35     2.549510         3.0          2.549510
## 36     2.529822         3.2          2.529822
## 37     2.529822         2.9          2.529822
## 38     2.529822         2.7          2.529822
## 39     2.529822         3.2          2.529822
## 40     2.529822         2.8          2.529822
## 41     2.529822         2.8          2.529822
## 42     2.529822         3.1          2.529822
## 43     2.509980         3.3          2.509980
## 44     2.509980         2.5          2.509980
## 45     2.509980         2.3          2.509980
## 46     2.509980         3.3          2.509980
## 47     2.509980         2.9          2.509980
## 48     2.509980         2.7          2.509980
## 49     2.509980         2.8          2.509980
## 50     2.509980         3.4          2.509980
## 51     2.509980         2.5          2.509980
```

In above code, we used the arrange and filter function from dplyr to filter the data with sepal length greater or equal to 2.5 and sorted the data by sepal length in ascending order.

```
#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
```

```
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
```

The mutate function was used to create new columns using the operation specified. Contains is used for checking if the column names contains the given word and ends_with checks if the column names ended with given word.

```
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
## # ... with 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
```

The `transmute()` function adds new variable and drops existing ones.

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

The `kable()` function from `knitr` package to create tables in LaTeX, HTML, Markdown and reStructured-Text.

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

mpg

cyl

disp

hp

drat

wt

qsec

vs

am

gear

carb

Mazda RX4

21.00

6.00

160.00

110.00

3.90

2.62

16.46

0.00

1.00

4.00

4.00

Mazda RX4 Wag

21.00

6.00

160.00

110.00

3.90

2.88

17.02

0.00

1.00

4.00

4.00

Datsun 710

22.80

4.00

108.00

93.00

3.85

2.32

18.61

1.00

1.00

4.00

1.00

Hornet 4 Drive

21.40

6.00

258.00

110.00

3.08

3.21

19.44

1.00

0.00

3.00

1.00

Hornet Sportabout

18.70

8.00

360.00

175.00

3.15

3.44

17.02

0.00

0.00

3.00

2.00

Valiant

18.10

6.00

225.00

105.00

2.76

3.46

20.22

1.00

0.00

3.00

1.00