Useful R commands for basic statistics and Notes

R as a Calculator

```r
1+1 # Simple Arithmetic
```

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

```r
2+3*4 # Operator precedence
```

```
## [1] 14
```

```r
3^2 # Exponentiation
```

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

```r
exp(1) # Exponential Function
```

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

```r
sqrt(10) # Calculate Square Root
```

```
## [1] 3.162278
```

```r
pi # Calculate Square Root
```

```
## [1] 3.141593
```

R as a Smart Calculator

```
# define variables
# using "<-" operator to set values
x <-1
y <-3
z <-4
x * y * z
```

```
## [1] 12
```

```
# variable names can include period
This.Year <- 2004
This.Year
```

```
## [1] 2004
```

R Vectors

`rep()` to repeat elements or patterns

```
rep(1,10) # repeats the number 1 ten times
```

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

`seq()` or m:n to generate sequences

```
seq(2,6) # sequence of integers between 2 and 6
```

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

```
seq(4,20,by=4) # skip 4 integer between 4 and 20
```

```
## [1]  4  8 12 16 20
```

`c()` to combines multiple values into a vector or list.

```r
# create vector with elements
x <-c (2,0,0,4)
y <-c (1,9,9,9)
# sums elements
x + y
```

```
## [1]  3  9  9 13
```

```r
# multiplication
x * 4
```

```
## [1]  8  0  0 16
```

```r
# sqrt()
sqrt(x)
```

```
## [1] 1.414214 0.000000 0.000000 2.000000
```

Indexing

```r
# create vector with value
x <-c (2,0,0,4)
# index first value of x
x[1]
```

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

```
# index all value exclude the 1st element of x
x[-1]
```

```
## [1] 0 0 4
```

```
# reassign value by indexing
x[1] <- 3; x
```

```
## [1] 3 0 0 4
```

```
# reassign all value exclude the 1st element of x
x[-1] = 5; x
```

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

TRUE OR FALSE

```
# create vector with value
y <-c (1,9,9,9) ; y
```

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

```
# check true or false
y < 9
```

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

```
# reassign value to check true or false
y[4] =1; y; y < 9
```

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

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

```
# Edits elements marked as TRUE in index
y[y<9] = 2; y
```

```
## [1] 2 9 9 2
```

Data Frames

Data Input

```
read.table (file, header = FALSE, sep = "", quote = "\"’",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv (file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2 (file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim (file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2 (file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Create from scratch `data.frame()`

```
# template 1
df <- data.frame (Name =c('Jon','Bill','Maria','Ben','Tina'),
                   Age =c(23, 41, 32, 58, 26)
                   )
print(df)
```

```
##    Name Age
## 1   Jon  23
## 2  Bill  41
## 3 Maria  32
## 4   Ben  58
## 5  Tina  26
```

```
# template 2
Name <- c('Jon','Bill','Maria','Ben','Tina')
Age <- c(23, 41, 32, 58, 26)

df <- data.frame(Name, Age)

print (df)
```

```
##    Name Age
## 1   Jon  23
## 2  Bill  41
## 3 Maria  32
## 4   Ben  58
## 5  Tina  26
```

```r
# df['<column_name>'] to retrieve column
df['Name']
```

```
##    Name
## 1   Jon
## 2  Bill
## 3 Maria
## 4   Ben
## 5  Tina
```

```r
# df$<column_name> to retrieve column
df$Name
```

```
## [1] "Jon"   "Bill"  "Maria" "Ben"   "Tina"
```

```r
# df[row,column] to retrieve column
df[,2]
```

```
## [1] 23 41 32 58 26
```

```r
df[,-2]
```

```
## [1] "Jon"   "Bill"  "Maria" "Ben"   "Tina"
```

Control Statements

- `if ... else ...`
- `for` loops
- `repeat` loops
- `while` loops
- `next`, `break` statements

if

```
x <- 5
if (x > 0)
  {
  print('Positive number')
  }
```

```
## [1] "Positive number"
```

if ... else ...

```
if (test_expression)
  {
  statement1
  } else
  {
  statement2
  }
```

```
x <- -5
if (x > 0)
  {
  print('Positive number')
  } else
  {
  print('Negatitve number')
  }
```

```
## [1] "Negatitve number"
```

for Loop

```r
# sample 1 Loop Through Vector in R (Basics)
for(i in 1:10) {
  x1 <- i^2
  print(x1)
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 36
## [1] 49
## [1] 64
## [1] 81
## [1] 100
```

```r
# sample 2 Looping Over Character Vectors
x <- c("Max", "Tina", "Lindsey", "Anton", "Sharon")
for(i in x) {
  print(paste("The name", i, "consists of", nchar(i), "characters."))
}
```

```
## [1] "The name Max consists of 3 characters."
## [1] "The name Tina consists of 4 characters."
## [1] "The name Lindsey consists of 7 characters."
## [1] "The name Anton consists of 5 characters."
## [1] "The name Sharon consists of 6 characters."
```

```
# sample 3 Store for-Loop Results in Vector by Appending
x <- numeric()
for(i in 1:10) {
  x <- c(x, i^2)
}
print(x)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
# sample 4 Nested for-Loop in R
x <- character()
for(i in 1:5) {
  for(j in 1:3) {
    x <- c(x, paste(LETTERS[i], letters[j], sep = "_"))
  }
}
x
```

```
##  [1] "A_a" "A_b" "A_c" "B_a" "B_b" "B_c" "C_a" "C_b" "C_c" "D_a" "D_b" "D_c"
## [13] "E_a" "E_b" "E_c"
```

```
# sample 5: Break for-Loop Based on Logical Condition
for(i in 1:10) {
  x <- i^2
  print(x)
  if(i >= 5) {
    break
  }
}
```

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

```
# sample 6: Continue to Next Iteration of for-Loop
for(i in 1:10) {
  if(i %in% c(1, 5, 7)) { # Conditionally skip iteration
    next
  }
  x <- i^2
  print(x)
}
```

```
## [1] 4
## [1] 9
## [1] 16
## [1] 36
## [1] 64
## [1] 81
## [1] 100
```

```
# sample 7: for-Loop Over Data Frame Columns
data(iris) # Loading iris flower data set
#head(iris) # Inspecting iris flower data set
df <- iris

for(i in 1:ncol(df)) {
  if(grepl("Width", colnames(df)[i])) {  # Logical condition
    df[ , i] <- df[ , i] + 1000
  }
}
head(df)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1      1003.5          1.4      1000.2  setosa
## 2          4.9      1003.0          1.4      1000.2  setosa
## 3          4.7      1003.2          1.3      1000.2  setosa
## 4          4.6      1003.1          1.5      1000.2  setosa
## 5          5.0      1003.6          1.4      1000.2  setosa
## 6          5.4      1003.9          1.7      1000.4  setosa
```

```
# sample 8: for-Loop Through List Object
my_list <- list(1:5, letters[3:1],"XXX")
for(i in 1:length(my_list)) {
  my_list[[i]] <- rep(my_list[[i]], 3)
}
my_list
```

```
## [[1]]
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
##
## [[2]]
## [1] "c" "b" "a" "c" "b" "a" "c" "b" "a"
##
## [[3]]
## [1] "XXX" "XXX" "XXX"
```

while Loop

A while loop in R programming is a function designed to execute some code until a condition is met. Take care! While loops may never stop if the logic condition is always TRUE.

```
while (logic_condition) {
    # Code
}
```

```
# Variable initialization
n <- 0
square <- 0

# While loop
while(square <= 4000) {
    n <- n + 1
    square <- n ^ 2
}

# Results
n
```

```
## [1] 64
```

```
square
```

```
## [1] 4096
```

```
# Variable initialization
x <- c(1, 2, 3, 4)
y <- c(0, 0, 5, 1)
n <- length(x)
i <- 0
z <- numeric(n)


x
```

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

```
y
```

```
## [1] 0 0 5 1
```

```
n
```

```
## [1] 4
```

```
i
```

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

```
z
```

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

```
# While loop
while (i <= n) {
    z[i] <- x[i] + y[i]
    i <- i + 1
}


z
```

```
## [1] 1 2 8 5
```

Functions in R

- **Online Help**
- **Random Generation**
- **Input / Output**
- **Data Summaries**
- **Exiting R**

Defining Functions

```r
# Defining Functions
square <- function(x = 10) x * x

# Results
square()
```

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

```r
square(2)
```

```
## [1] 4
```

```r
square(3)
```

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

Random Generation

```r
runif(n, min = 1, max = 1)
```

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

```r
rbinom(n, size=10, prob=0.5)
```

```
## [1] 5 7 5 6
```

```r
rnorm(n, mean = 0, sd = 1)
```

```
## [1] -1.0520044 -0.7083808 -0.2397466  1.3206366
```

```r
rexp(n, rate = 1)
```

```
## [1] 0.7970554 0.5498658 1.5491640 0.4293379
```

```r
rt(n, df = 20)
```

```
## [1] -0.04059806 -0.17885324  0.57514874 -0.71103768
```

Basic Utility Functions

- `length()` returns the number of elements
- `mean()` returns the sample mean
- `median()` returns the sample mean
- `range()` returns the largest and smallest values
- `unique()` removes duplicate elements
- `summary()` calculates descriptive statistics
- `diff()` takes difference between consecutive elements
- `rev()` reverses elements