Ve572 Lecture 3

Manuel and Jing

UM-SJTU Joint Institute

May 22, 2018

- Data frame

```
> # Vector by crated by replication statement
> manufacturer = rep("audi", 6)

> displ = c(1.80, 1.80, 2.00, 2.80, 3.10, 1.80)
> cyl = as.integer(c(rep(4,4), rep(6,2)))
> cty = c(18, 21, 20, 21, 16, 18)

> mpg.df = data.frame(manufacturer, displ, cyl, cty)
> attributes(mpg.df)
```

```
$names
[1] "manufacturer"   "displ"   "cyl"   "cty"

$row.names
[1] 1 2 3 4 5 6

$class
[1] "data.frame"
```

- For this tiny data set, asking R to display it explicitly makes sense

```
> mpg.df
```

```
  manufacturer displ cyl cty
1         audi   1.8   4  18
2         audi   1.8   4  21
3         audi   2.0   4  20
4         audi   2.8   4  21
5         audi   3.1   6  16
6         audi   1.8   6  18
```

```
> mpg.df[["cty"]]; mpg.df$cty[5]; mpg.df[[4]][5]
```

```
[1] 18 21 20 21 16 18
[1] 16
[1] 16
```

- Notice the indexing is very similar to list indexing.

```
> is.data.frame(mpg.df)
```

```
[1] TRUE
```

```
> is.list(mpg.df)
```

```
[1] TRUE
```

- Notice data frame can be indexed as if they are matrices, but...

```
> mpg.df[3,1]
```

```
[1] audi
Levels: audi
```

```
> is.matrix(mpg.df)
```

```
[1] FALSE
```

Q: What is the difference between list and data.frame?

Conditional and Repetitive Execution

- ```
  > if (is.matrix(mpg.df)) {
  +   print("Data frame and Matrix are the same in R")
  + } else {
  +   print("A data frame is not a matrix")
  + }
  ```

```
[1] "A data frame is not a matrix"
```

```
> is.data.frame(mpg.df) & is.list(mpg.df)    # and
```

```
[1] TRUE
```

```
> is.data.frame(mpg.df) | is.matrix(mpg.df) # or
```

```
[1] TRUE
```

```
> ! is.matrix(mpg.df)                        # not
```

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

- The following functions are often useful

```
> x = 1:10
> all(x>0)
```

```
[1] TRUE
```

```
> any(x>9)
```

```
[1] TRUE
```

```
> x = 1:10; (x = ifelse(x > 5, x, -x))
```

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

Q: What is `s` after the following `for` loop statement?

```
> s = 0
> for (eps in x) {              # Repetitive execution
+   s = s + eps
+ }
```

- Notice R does not need to use an integer loop variable.

```
> myfunc =
+   function(x) {          # Define myfunc
+     s = 0
+     for (eps in x) {
+       if (eps <= 0) {
+         next             # jump to the end of the loop
+       }
+       s = s + eps
+       if (s > 20) {
+         break            # stop the loop
+       }
+     }
+     s                    # output value
+   }
```

Q: What do you think the result of the following statement is?

```
> x = 1:10; x = ifelse(x > 5, x, -x); myfunc(x)
```

- Subsetting

```
> mpg.df[c(1,3,5,2,4,6), ]  # A selection of rows
```

```
  manufacturer displ cyl cty
1         audi   1.8   4  18
3         audi   2.0   4  20
5         audi   3.1   6  16
2         audi   1.8   4  21
4         audi   2.8   4  21
6         audi   1.8   6  18
```

```
> mpg.df[c(1,3,5), c(1,3)]  # A selection of cols
```

```
  manufacturer cyl
1         audi   4
3         audi   4
5         audi   6
```

```
> mpg.df[, c("displ", "cty")]
```

```
  displ cty
1   1.8  18
2   1.8  21
3   2.0  20
4   2.8  21
5   3.1  16
6   1.8  18
```

```
> tmp = mpg.df[, 1]
> class(tmp)                    # Vector
```

```
[1] "factor"
```

```
> tmp = mpg.df[, 1, drop = FALSE]
> class(tmp)                    # Data frame
```

```
[1] "data.frame"
```

```
> rowid = sample(nrow(mpg.df), size = 3)
> rowid
```

```
[1] 6 3 1
```

```
> rowidsort = sort(rowid)
> rowidsort
```

```
[1] 1 3 6
```

```
> mpg.df[rowidsort,]
```

```
  manufacturer displ cyl cty
1         audi   1.8   4  18
3         audi   2.0   4  20
6         audi   1.8   6  18
```

```
> sample(nrow(mpg.df), 3, replace = TRUE)
```

```
[1] 2 4 4
```

```
> # Extracting rows met the conditions using subset
> tmp.df = subset(mpg.df, cyl == 4 & displ >= 2.0)
> tmp.df
```

```
  manufacturer displ cyl cty
3         audi   2.0   4  20
4         audi   2.8   4  21
```

```
> subset(mpg.df, 1:nrow(mpg.df) %in% rowid)
```

```
  manufacturer displ cyl cty
1         audi   1.8   4  18
3         audi   2.0   4  20
6         audi   1.8   6  18
```

```
> subset(mpg.df, displ >= 2.0, select = c(cyl, cty))
```

```
  cyl cty
3   4  20
4   4  21
5   6  16
```

- Of course, manually input the data is often not the way nowadays,

```
> # Remove everything in the working environment
> # So don't do this unless you are really sure.
> rm(list = ls())
```

- Built-in Data Sets

```
> # Print a list of all data sets under all packages
> data(package = .packages(all.available = TRUE))
> data(motor, package = "boot")          # Loading
> class(motor)
```

```
[1] "data.frame"
```

```
> head(motor,3)          # Return the first 3 rows
```

```
  times accel strata   v
1   2.4   0.0      1 3.7
2   2.6  -1.3      1 3.7
3   3.2  -2.7      1 3.7
```

```
> help(motor, package = "boot")      # Information
```

- Local

```
> (nz.df = read.table(                # Read a file
+       "~/Desktop/nzislands",        # file
+       col.names = c("Island", "Area"),
+       colClasses = c("character","numeric")))
```

```
   Island   Area
1   South 151215
2   North 113729
3 Stewart   1746
```

```
> str(nz.df)                          # Display structure
```

```
'data.frame':   3 obs. of  2 variables:
 $ Island: chr  "South" "North" "Stewart"
 $ Area  : num  151215 113729 1746
```

- Internet

```
> semiconductor.df = read.table(
+   "http://lib.stat.cmu.edu/jasadata/hughes-l-d-g",
+   skip = 23)                         # skipping summary
```

- Cleaning

```
> (person.df = read.table(
+   "~/Desktop/unnamed",
+   header = TRUE,   # file has column names
+   sep = ","))      # separator
```

```
  age height
1  21    6.0
2  42    5.9
3  18   5.7*
4  21   <NA>
```

```
> class(person.df$height)
```

```
[1] "factor"
```

- Adding the statement `colClasses = rep('numeric',2)` results an error

```
# Error: scan() expected 'a real', got '5.7*'
```

```
> (tmp.txt = readLines("~/Desktop/unnamed"))
```

```
[1] "age,height" "21,6.0"      "42,5.9"
[4] "18,5.7*"    "21,NA"
```

```
> (tmp.clean = gsub("[*]", "", tmp.txt))
```

```
[1] "age,height" "21,6.0"      "42,5.9"
[4] "18,5.7"     "21,NA"
```

```
> help("regex") # Regular expression

> person.df = read.table(
+   textConnection(tmp.clean), # source of data
+   header = TRUE, sep = ",")
>
> str(person.df)
```

```
'data.frame':   4 obs. of  2 variables:
 $ age   : int  21 42 18 21
 $ height: num  6 5.9 5.7 NA
```

- CSV

```
> carprice.df = read.table("~/Desktop/carprice.csv",
+         header = TRUE, sep = ",",
+         stringsAsFactors = FALSE, row.names = "X")

> carprice.df = read.csv("~/Desktop/carprice.csv",
+         stringsAsFactors = FALSE, row.names = "X")

> read.csv
```

```
function (file, header = TRUE, sep = ",",
                quote = "\"", dec = ".",
                fill = TRUE, comment.char = "", ...)
{
        read.table(
                file = file, header = header,
                sep = sep, quote = quote,
                dec = dec, fill = fill,
                comment.char = comment.char, ...)
}
```

- Fixed width file

```
> help(read.fwf)
```

- Use XLConnect for Excel data.

```
> # install.packages("XLConnect", type = "binary")
> library(XLConnect)
```

- Use foreign for Binary, Matlab, Minitab, SAS, SPSS, Stata, and etc.

```
> library(foreign)
```

- Use RMySQL for Relational databases.

```
> library(RMySQL)
```

- Use RHadoop for Non-Relational databases.
- For more information regarding R basics see R-manuals.