



# Advanced Analytics with R

Manipulating Data

# Agenda

- Learn how to create/load data in R
  - Offline data
  - Online data
- Learn how to manipulate data
  - Clean data
  - Construct new fields
  - Subset of data
  - Apply functions on data

Loading Data

# Source of Data



# Create Data on Your Own

```
mydata<- data.frame(age=numeric(0),gender=character(0), weight =numeric(0))
```

```
mydata <- edit(mydata)
```

# Loading CSV file

```
care.data<-read.csv("hospital-data.csv")  
head(care.data)
```

```
system.time(care.data<-read.csv("hospital-data.csv"))
```

# More efficient way to load data

- Read a few lines of data and check the structure

```
data.sample<-read.csv("hospital-data.csv",nrows = 5)  
data.sample
```

- Pre-set the colclasses of data while loading

```
colclass<-c("character","character","character","character","character",  
"character","factor","character","factor","character","factor","factor","factor")  
system.time(care.data<-read.csv("hospital-data.csv",colClasses = colclass))
```

# XML file





# Load XML file

```
data<-xmlParse("books.xml")
```

```
data.l <- xmlToList(data)
```

```
data.l[1]$book
```

```
data.f<-xmlToDataFrame(data,stringsAsFactors = FALSE,  
colClasses = c("character","character","character","numeric","character","character"))
```

# Exercise

- Try to load “books&CD.xml”

# Load Online Data

```
xml.url<-"http://www.w3schools.com/xml/plant_catalog.xml"
```

```
plants<-xmlParse(xml.url)
```

# Load Data on HTML Page

```
library(XML)
library(RCurl)
```

```
theurl <- "http://apps.saferoutesinfo.org/legislation_funding/state_apportionment.cfm"
urldata <- getURL(theurl)
```

```
data <- readHTMLTable(urldata, stringsAsFactors = FALSE)
class(data)
```

```
df <- as.data.frame(data)
```

```
#multi-table case
```

```
# http://www.cdc.gov/mmwr/preview/mmwrhtml/mm6128a3.htm?s_cid=mm6128a3_e%0d%0a
```

# Use rvest package

```
library(rvest)
iproperty<-read_html('http://www.iproperty.com.sg/sale/property/?pg=2')
listing <- iproperty %>% html_nodes(".search-listing")
children<-html_children(listing[1])
path=children[1]%>%html_attr("href")
```

# Data Management

# Common data problems

- Missing data
- Duplicate entries and common misspellings
- Inconsistent data
  - Misclassified data
  - Numbers change meaning over time
  - Conflicting data
  - Dirty data

# Knowing your data - *Length*

```
length(1:14)
```

```
length(matrix(3:8,ncol=3,nrow=2))
```

```
length(mtcars)
```



# Knowing your data – *head* and *tail*

- `head(mtcars)`
- `tail(mtcars,n=10)`

# Knowing your data - *order*

- `data1 <- mtcars[order(mpg),]`
- `data2<- mtcars[order(mpg, cyl),]`
- `data3<- mtcars[order(mpg, -cyl),]`

# Knowing your data - *str*

- `str(mtcars)`

# Knowing your data - *table*

- `table(mtcars$cyl)`

# Growing your data – *rbind* and *cbind*

- `cbind(2:7, LETTERS[1:6])`
- `a<-cbind("a", 2:7)`
- `cbind(mtcars, "A")`

# Growing your data - merge

```
authors <- data.frame( surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")), nationality = c("US", "Australia", "US", "UK", "Australia"), deceased = c("yes", rep("no", 4)))
```

```
books <- data.frame( name = I(c("Tukey", "Venables", "Tierney", "Ripley", "Ripley", "McNeil", "R Core")),  
title = c("Exploratory Data Analysis", "Modern Applied Statistics ...", "LISP-STAT", "Spatial Statistics", "Stochastic Simulation", "Interactive Data Analysis", "An Introduction to R"),  
other.author = c(NA, "Ripley", NA, NA, NA, NA, "Venables & Smith"))
```

```
m1 <- merge(authors, books, by.x = "surname", by.y = "name")
```

```
m2 <- merge(authors, books, by.x = "surname", by.y = "name", all = TRUE)
```

# Growing your data – new column

```
mtcars$hp.type <- ifelse(mtcars$hp>100, "Strong","Weak")
```

# Exercise

- suppose we want to classify the car models in mtcars data set. Those with weight higher than 3.5 are to be defined as "Heavy", those with weight between 3 and 3.5 are to be defined as "Medium", and the rest are to be defined as "Light". Add a new column named "weight.type" to mtcars that contains category of car models according to this definition.



# Extracting data - *subset*

- `subset(mtcars,select = c(mpg,hp))`
- `mtcars[,c("mpg","hp")]`
- `mtcars [,c(1,4)]`

# Extracting data – logical condition

- `mtcars[mtcars$mpg>20,]`

# Reducing data – Drop a column

- `a<-mtcars`
- `a$mpg<-NULL`

# Sampling Data

```
a<-sample(c(FALSE,TRUE),nrow(mtcars),replace=TRUE,prob = c(0.2,0.8))  
training.data<-mtcars[a,]  
testing.data<-mtcars[!a,]
```

# Splitting Data - *split*

```
x<-c(letters[1:10],rnorm(10),seq(2,11))  
a<-gl(3,10)  
split(x,a)
```

# Splitting Data - *split*

```
airquality.month<-split(airquality,airquality$Month)
```

```
split(mtcars,list(mtcars$vs,mtcars$am))
```

# Apply Function on Data

# apply

- ***apply*** allows you to apply a function to a margins of an matrix or data frame.
- `z<-cbind(A=1:3,B=4:6)`
- `z`
- `apply(z,2,sum)`

"2" – operate on column basis

"1" – operate on row basis



# Exercise

- Add a new row named “sum” to mtcars, which stores sum of all fields of the data.

# apply with self-defined function

- `f<-function(x) x/c(2,4)`
- `a<-apply(z,1,f)`

# lapply

- ***lapply*** works on a list or array and always returns a list of the same length as the given list or array.

```
x<-list(A=1:4,B=rnorm(8))  
lapply(x,mean)
```

```
#A list of two matrices  
x<-list(matrix(1:12,nrow=3,ncol =  
4),matrix(letters[1:10],nrow=2,ncol=5))  
#Get the first column of each matrix  
lapply(x,function(m) m[,1])
```

# sapply

- ***sapply*** is a wrapper of `lapply` that returns a vector instead of a list
- `sapply` is more useful than `lapply` when working on data frame and matrix.

```
sapply(1:3, function(x) x^2)
```

# mapply

- ***mapply*** is a multivariate version of lapply. It applies FUN to the first elements of each ... arguments, the second elements, and so on

```
mapply(seq, 1:4, 4:1)  
mapply(rep, 1:4, 4:1)
```

```
mapply(rep, times = 1:4, x = 4:1)
```

# tapply

- ***tapply*** applies a function to each group of an array, grouped based on levels of certain factors

```
n<-17  
fac<-factor(rep(1:3, length = n), levels = 1:5)  
table(fac)  
tapply(1:n,fac,sum)
```

# Pivot table like features in R

```
tapply(OrchardSprays$decrease, OrchardSprays$treatment, sum)
```

# Multi-dimensional Statistics

```
mtcars$am<-as.factor(mtcars$am)
mtcars$vs<-as.factor(mtcars$vs)
#Average mpg of cars, categorised by "am" and "vs"
tapply(mtcars$mpg,INDEX = list(mtcars$am,mtcars$vs),mean)
```