DATA ANALYSIS WORKFLOW

- Getting data
- Cleaning & Transforming
- Exploring & Visualization

-Aditya Gupta
(University of Delhi)

DATA ACCESS

- Built in Datasets
  - datasets provided by R packages
  - `data()` [to see all datasets in packages]
  - `data(package = .package(TRUE))` [include all installed]
  - `data("kidney", package = "survival")` [access selected]
- Reading Tabular data from external file
  - `read.table(filename, header = TRUE, fill = TRUE, ….)`
  - returns a `dataframe`
    - other parameters :
      - `sep` [separator], `dec`, `nrows`, `skip`, `col_types`, `col.names`, `skipNul` (logical), `fileEncoding`

DATA ACCESS

- Other reading functions
  - `read.csv()` [to read comma separated files]
  - `read.csv2()` [to read semicolon separated files]
  - `read.delim(), read.delim2()` [generic fn to read delimited files]
  - `readLines(file)` [for unstructured files like, .txt]
- Writing Output
  - `write.table(output df or matrix, file = “output file”)`
  - `write.csv(...), write.csv2(…)`
  - `writeLines()` [for unstructured files]

SOME USEFUL FUNCTIONS

View a loaded dataset	`View(dfname)`
Display Dataset Structure	`str(dfname)`
Access a column	`dfname$colname`
Access a row	`dfname[rownum,]`
Access Individuals	`dfname[row,col]`
Read first n rows	`head(dfname,n)`
Read last n rows	`tail(dfname,n)`
To known a column nominal count	`levels(dfname$colname)`
No. of columns	`ncol(dfname)`
No.of rows	`nrow(dfname)`

Aditya Gupta

Getting Data

ALSO POSSIBLE TO READ

- Excel files (package: "xlsx")
  - `read.xlsx(filename, sheetIndex, startRow=NULL, endRow=NULL, rowIndex=NULL, colIndex=NULL)`
  - `write.xlxs (x, output_file)`
- XML & HTML files (package : "XML")
  - `xmlParse(xml_file, useInternalNodes=FALSE) & xmlTreeParse(xml_file)`
  - `htmlParse(html_file) & htmlTreeParse(html_file)`
- JSON files (package: "RJSONIO", "rjson")
  - `fromJSON(JSONfile)`
- Matlab files (package : "R.matlab")
  - `readMat()`
  - `writeMat()`

ACCESSING WEB DATA

- Sites with API
  - `WDI, wbstats` package (World Development Indicator data)
  - `SmarterPoland` package (Polish government data)
  - `quantmod, Quandl` package (stock tickers by Yahoo!)
  - `twitterR` package (Twitter's user & their tweets)
  - `rnoaa` package (National Oceanic and Atmospheric Administration)
  - `censusr, acs` package (United States Census)
  - `GuardianR, rdian` package(The Guardian Media Group)
  - `blsAPI` package (Bureau of Labor Statistics)
  - `rtimes` package (New York Times)
- Scraping Web Pages (package : `"RCurl", "httr"`)
  - `getURL(url)`
  - `GET(url)` [`httr` package]

ACCESSING DATABASES

- Install & load DBI package
  - provide unified syntax to access several DBMSs
- Install& load backend database package →

```
library (DBI)
library (RSQLite)

query<-"SELECT * FROM User"

driver <- dbDriver("SQLite")
conn <- dbConnect(driver, db_file)
on.exit (
{
#this code block runs at the end of the function,
#even if an error is thrown
dbDisconnect(conn)
dbUnloadDriver(driver)
})
dbGetQuery(conn, query)
```

SQLite	RSQLite
MySQL	RMySQL
Oracle	ROracle
JDBC	RJDBC
PostgreSQL	PostgreSQL
MongoDB	RMongo

MANIPULATING DATA FRAMES

- Adding & Replacing new columns
  - using `with, within, mutate(plyr)`

```
dfname$colname <- with(dfname, expression)

dfname <- within(dfname, { expression1, expression2 })

dfname <- mutate(dfname, new_col1 = new_col1_value,
                         new_col2 = new_col2_value)
```

- Reading partial data
  - `subset(dfname, where_condn, c(col1,col2))`
  - `sqldf(query) [package = sqldf]`
  - `dfname[,c(col1, col2)]`
- Drop columns
  - `dfname <- subset(dfname, select = -c(a,c))`

DEALING WITH MISSING VALUES

- Test for Missing values
  - `is.na(dfname)` [identify NAs in vector or data frame]
  - `is.na(dfname$colname)` [identify NAs in specific df column]
  - `which(is.na(df$colname))` [identify location of NAs]
  - `sum(is.na(df))` [identify count of NAs in data frame]
  - `colSums(is.na(df))` [total missing values in each column]
- Recode Missing values
  - `x[is.na(x)] <- mean(x, na.rm=TRUE)` [recode missing with mean value]
  - `df$col[is.na(df$col)] <- mean(df$col2, na.rm=TRUE)` [recode df column]
- Exclude Missing values
  - `complete.cases(dfname)` [return logical vector identifying rows without missing values]
  - `na.omit(dfname)` [removes rows with missing values]
  - `na.fail(dfname)` [throws error for missing values]

SUMMARY STATISTICS

Mean value	mean(df$col)
Median	median(df$col)
Variance	var(df$col)
Standard Deviation	sd(df$col)
Mean Absolute deviation	mad(df$col)
Minimum	min(df$col)
Maximum	max(df$col)
Smallest at each point across vectors	with(df, pmin(col1,col2))
Largest at each point across vectors	with(df,pmax(col1,col2))
Quantile	quantile(df$col)
Inner-Quartile Range	IQR(df$col)
Five number summary	fivenum(df$col)
Dataset Summary	summary(df)

Exploring & Visualizaton

VISUALIZATION

- Three plotting system
  - `base`
    - oldest of all plotting system.
    - results are drawn on screen.
    - requires lot of fiddling to polish.
    - `grid` was introduced to allow more flexible plotting.
      - draw at low level.
      - require lot of coding.
  - `lattice`
    - built on top of `grid` package.
    - provide high level function for plotting.
    - results plots can be stored in variable for easy updation.
    - can contain multiple panel in a plot (for say comparison plots).
  - `ggplot2`
    - "gg" stands for grammar graphics.
    - also built on top of `grid`.
    - most modern of all plotting system.
    - results can be stored in variables for updation & redrawing.

Remember!

1. They are mostly incompatible.
2. can do everything with ggplot2.
3. ggplot2 does more calculation than others.
   - for quick, dirty plots for large datasets, use other plotting system.

SCATTER PLOTS

```
#base package
with(df, plot(x,y))
with(df,plot(x,y, main=“Graph title”, xlab=“x axis lable”,
               ylab=“y axis lable”, col= “blue”, pch=18, log=“x”))
```

```
#lattice package
library(lattice)
xyplot(y~x, dataset) # simple plot
xyplot(y~x, dataset, main=“Graph title”,col=“blue”, pch=20, scales = list(log=TRUE))
xyplot(y~x | z, dataset, scales = list (relation=“same”, y=list(log=TRUE),
                                                    alternating = FALSE), layout=c(4,2))
```

```
#ggplot2 package
library(lattice)
ggplot (dataset, aes(x,y)) +
geom_point (color = “violet”, shape = 20) +
scale_x_log10 (breaks = seq (0,100,10)) +
ggtitle(“Plot title”) +
xlab (“x axis label”) +
facet_wrap(~ z, ncol = 4) +
theme (axis.text.x = element_text (angle = 30))
```

Exploring & Visualizaton

OTHER EXAMPLE PLOTS

- Line plots

```
#base package
with(df, plot(…, type=“l”,…))

#lattice package
xyplot (y~x, dataset, type =“l”)

#ggplot2 package
ggplot(dataset, aes(x,y))+
geom_line()
```

- Box Plots

```
#base package
boxplot(y~z dataset, main =“Title”)

#lattice package
bwplot (y~z, dataset)

#ggplot2 package
ggplot(dataset, aes(z,y))+
geom_boxplot()
```

- Histogram

```
#base package
with(df, hist(col, bins, freq=FALSE,
                        main=“title”))
#lattice package
histogram (~col, dataset, break=10)
                       #type =“count”
#ggplot2 package
ggplot(dataset, aes(col, ..count..)) +
geom_histogram(bandwidth=4)
```

- Other plotting packages
  - vcd (for categorical data)
  - playwith (interactive plots)
  - iplots (interactive)
  - googleVis (wrapper google chart tools)
  - rCharts (wrapper to JS lib)
  - rgl (3D plots)
  - animation (make GIF's)

> TELL ME AND I FORGET. TEACH ME AND I REMEMBER. INVOLVE ME AND I LEARN.

Benjamin Franklin