CWRU DSCI351-451: Week01a-p

Roger H. French

28 August, 2018

Contents

1.1.2.1 Reading, Homeworks, Projects, SemProjects

- Readings:
- Homeworks
- Data Science Projects:
- 451 SemProjects:
- Friday Comm. Hour

1.1.2.2 Syllabus

License: CC-BY-SA 4.0

1.1.2.2.1 setup for r-code chunks

```r
options("digits" = 5)
options("digits.secs" = 3)
```

1.1.2.3 A Simple Overview of R

We'll look in more detail, going forward

1.1.2.3.1 Intro to some R: Data Types

- Primitives (numeric, integer, character, logical, factor)
- Data Frames
- Lists
- Tables
- Arrays
- Environments
- Others (functions, closures, promises..)

1.1.2.3.2 Simple Types, and their class

```r
x <- 1
class(x)
## [1] "numeric"

y <- "Hello World"
class(y)
## [1] "character"

z <- TRUE
class(z)
## [1] "logical"

as.integer(z)
## [1] 1
```

1.1.2.3.3 Simple Types - Vectors

The basic type unit in R is a vector

```r
x <- c(1,2,3)
x
## [1] 1 2 3
x <- 1:3
x[1]
## [1] 1
x[0]
## integer(0)
x[-1]
## [1] 2 3
```

Day:Date	Foundation	Practicum	Reading	Due
w1a:Tu:8/28/18	ODS Tool Chain	R, Rstudio, Git		
w1b:Th:8/30/18	Setup ODS Tool Chain	Bash, Git, Twitter	PRP4-33	HW1
w2a:Tu:9/4/18	What is Data Science	OIS:Intro2R	PRP35-64	**HW1 Due**
w2b:Th:9/6/18	Data Analytic Style, Git	Teatime:Intro2R, For loops	PRP65-93	HW2
w3a:Tu:9/11/18*	Struct. of Data Analysis, SemProj	ISLR:Intro2R	PRP94-116	**HW2 Due**
w3b:Th:9/13/18*	OIS3 Intro to Data	GapMinder, Dplyr, Magrittr	OI1-1.9,	
w4a:Tu:9/18/18	OIS3, Intro2Data part 2, Data	EDA: PET Degr.	EDA1-31	Proj1
w4b:Th:9/20/18	Hypothesis Testing	GGPlot2 Tutorial	EDA32-58	HW3
w5a:Tu:9/25/18	Distributions	SemProj RepOut1	R4DS1-3	**HW3 Due**
w5b:Th:9/27/18	Wickham DSCI in Tidyverse	SemProj RepOut1	R4DS4-6	**SemProj1,**
w6a:Tu:10/2/18	OIS Found. of Inference	Inference	R4DS7-8	**Proj1 Due**
w6b:Th:10/4/18		Midterm Review	R4DS9-16 Wrangle	
w7a:Tu:10/9/18*	Summ. Stats & Vis.	Data Wrangling		
w7b:Th:10/11/18*	**MIDTERM EXAM**			HW4
w8a:Tu:10/16/18	Numerical Inference	Tidy Check Explore	OIS4	**HW4 Due**
w8b:Th:10/18/18	Algorithms, Models	Pairwise Corr. Plots	OIS5.1-4	Proj 2, HW5
Tu:10/23	**CWRU FALL BREAK**		R4DS17-21 Program	
w9b:Th:10/25/18	Categorical Infer	Predictive Analytics	OIS6.1,2	
w10a:Tu:10/30/18	SemProj	SemProj	OIS7	**SemProj2 HW5 Du**
w10b:Th:11/1/18	Lin. Regr.	Lin. Regr.	OIS8	**Proj.2 due**
w11a:Tu:11/6/18	Inf. for Regression	Curse of Dim.	OIS8	Proj 3
w11b:Th:11/8/18	Model Accuracy	Training Testing	ISLR3	HW6
w12a:Tu:11/13/18	Multiple Regr.	Mul. Regr. & Pred.	ISLR4	**HW6 due**
w12b:Th:11/15/18	Classification		ISLR6	
w13a:Tu:11/20/18	Classification	Clustering	ISLR5	**Proj 3 due**
Th:11/22/18	**THANKSGIVING**			Proj 4
w14a:Tu:11/27/18	Big Data	Hadoop		
w14b:Th:11/29/18	InfoSec	VerisDB		**SemProj3**
w15a:Tu:12/4/18	SemProj ReportOut3			
w15b:Th:12/6/18	SemProj ReportOut3			**Proj4**
	FINAL EXAM	**Monday12/17, 12:00-3:00pm**	Olin 313	**SemProj4 due**

Figure 1: DSCI351-451 Syllabus

1.1.2.3.4 Generating Vectors

R provides lots of convenience functions for data generation:

```
  rep(0, 5)
## [1] 0 0 0 0 0
  seq(1,10)
##  [1]  1  2  3  4  5  6  7  8  9 10
  seq(1,2,.1)
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
  seq(1,2,length.out = 6)
## [1] 1.0 1.2 1.4 1.6 1.8 2.0
```

1.1.2.3.5 Indexing, c is concatenate

-to see the help on c: help(c)

```
  x <- c(1, 3, 4, 10, 15, 20, 50, 1, 6)
  x > 10
## [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
  which(x > 10)
## [1] 5 6 7
  x[x > 10]
## [1] 15 20 50
  x[!x > 10]
## [1]  1  3  4 10  1  6
  x[x <= 10]
## [1]  1  3  4 10  1  6
  x[x > 10 & x < 30]
## [1] 15 20
```

1.1.2.3.6 Functions, very easy to define

Usually take code in scripts, make functions from them

```
  square <- function(x) x^2
  square(2)
## [1] 4

  pow <- function(x, p=2) x^p
  pow(10)
## [1] 100
  pow(10,3)
## [1] 1000
  pow(p = 3,10)
## [1] 1000
```

Functions can be passed as data:

```
  g <- function(x, f) f(x)
  g(10, square)
## [1] 100

  h <- function(x,f,...) f(x,...)
  h(10, pow, 3)
## [1] 1000
```

1.1.2.3.7 R is Vectorized

Example - multiplying two vectors:

```
mult <- function(x,y) {
  z <- numeric(length(x))
  for (i in 1:length(x)) {
    z[i] <- x[i] * y[i]
    }
  z
  }

mult(1:10,1:10)
```

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

1.1.2.3.8 R is Vectorized

Multiplying two vectors 'the R way':

```
1:10 * 1:10
```

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

NOTE: R recycles vectors of unequal length:

```
1:10 * 1:2
```

```
##  [1]  1  4  3  8  5 12  7 16  9 20
```

1.1.2.3.9 NOTE: Random Number Generation

R contains a huge number of

- built-in random number generators
- for various probability distributions

```
# Normal variates, mean=0, sd=1
  rnorm(10)
```

```
##  [1]  1.11285  0.22347  0.51833 -0.29659 -0.96787 -0.42539 -0.24916
##  [8] -1.52838  1.34570 -0.33464
```

```
rnorm(10, mean = 100, sd = 5)
```

```
##  [1] 102.218  94.909  92.869  97.830  95.389 102.465 106.292 100.273
##  [9]  91.160 103.837
```

Many different distributions available (the r* functions)

1.1.2.4 Data Frames

1.1.2.4.1 Data Frames are fundamental

Data frames are the fundamental structure

- used in data analysis
- Similar to a database table in spirit
- (named columns, distinct types)

```r
d <- data.frame(x = 1:6, y = "AUDUSD", z = c("one","two"))
d
```

```
##   x      y   z
## 1 1 AUDUSD one
## 2 2 AUDUSD two
## 3 3 AUDUSD one
## 4 4 AUDUSD two
## 5 5 AUDUSD one
## 6 6 AUDUSD two
```

1.1.2.4.2 Data Frames can be indexed

Data frames can be indexed like a vector or matrix:

```r
# First row
d[1,]
##   x      y   z
## 1 1 AUDUSD one

# First column
d[,1]
## [1] 1 2 3 4 5 6

# First and third cols, first two rows
d[1:2,c(1,3)]
##   x   z
## 1 1 one
## 2 2 two
```

1.1.2.4.3 Generate a Data Frame

Let's generate some dummy data:

- Using data.frame

```r
generateData <- function(N) data.frame(time = Sys.time() + 1:N,
  sym = "AUDUSD",
  bid = rep(1.2345,N) + runif(min = -.0010,max = .0010,N),
  ask = rep(1.2356,N) + runif(min = -.0010,max = .0010,N),
  exch = sample(c("EBS","RTM","CNX"),N, replace = TRUE))

prices <- generateData(50)
head(prices, 5)
```

```
##                      time    sym    bid    ask exch
## 1 2018-08-28 10:23:33.844 AUDUSD 1.2345 1.2362  RTM
## 2 2018-08-28 10:23:34.844 AUDUSD 1.2344 1.2355  CNX
## 3 2018-08-28 10:23:35.844 AUDUSD 1.2344 1.2347  RTM
## 4 2018-08-28 10:23:36.844 AUDUSD 1.2353 1.2352  EBS
## 5 2018-08-28 10:23:37.844 AUDUSD 1.2348 1.2362  CNX
```

1.1.2.4.4 Data Frames

We can add/remove columns on the fly:

```r
prices$spread <- prices$ask - prices$bid
prices$mid <- (prices$bid + prices$ask) * 0.5
head(prices)
```

```
##                      time    sym    bid    ask exch      spread    mid
## 1 2018-08-28 10:23:33.844 AUDUSD 1.2345 1.2362  RTM  0.00170284 1.2354
## 2 2018-08-28 10:23:34.844 AUDUSD 1.2344 1.2355  CNX  0.00113161 1.2349
## 3 2018-08-28 10:23:35.844 AUDUSD 1.2344 1.2347  RTM  0.00031873 1.2345
## 4 2018-08-28 10:23:36.844 AUDUSD 1.2353 1.2352  EBS -0.00012920 1.2352
## 5 2018-08-28 10:23:37.844 AUDUSD 1.2348 1.2362  CNX  0.00137413 1.2355
## 6 2018-08-28 10:23:38.844 AUDUSD 1.2339 1.2363  RTM  0.00243149 1.2351
```

1.1.2.4.5 Operations on Data Frames

Some basic operations on data frames:

```r
  names(prices)
## [1] "time"   "sym"    "bid"    "ask"    "exch"   "spread" "mid"

  table(prices$exch)
##
## CNX EBS RTM
##  17  17  16

  summary(prices$mid)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.23    1.23    1.24    1.24    1.24    1.24
```

1.1.2.4.6 Operations across data frame dimensions

Operations can be applied across different dimensions of a data frame:

```r
  sapply(prices,class)
## $time
## [1] "POSIXct" "POSIXt"
##
## $sym
## [1] "factor"
##
## $bid
## [1] "numeric"
##
## $ask
## [1] "numeric"
##
## $exch
## [1] "factor"
##
## $spread
## [1] "numeric"
##
## $mid
## [1] "numeric"
```

1.1.2.4.7 These Operators Are Functions

```r
`(`
```

```
## .Primitive("(")
```

```r
(1 + 2)
```

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

```r
`(` <- function(x) 42
(1 + 2)
```

```
## [1] 42
```

```r
rm("(")
```

1.1.2.5 Examples in R

1.1.2.5.1 Example: Median Absolute Deviation

$MAD(x) = median\left(\left|Y_i - \hat{Y}\right|\right)$

```r
mad
```

```
## function (x, center = median(x), constant = 1.4826, na.rm = FALSE,
##     low = FALSE, high = FALSE)
## {
##     if (na.rm)
##         x <- x[!is.na(x)]
##     n <- length(x)
##     constant * if ((low || high) && n%%2 == 0) {
##         if (low && high)
##             stop("'low' and 'high' cannot be both TRUE")
##         n2 <- n%/%2 + as.integer(high)
##         sort(abs(x - center), partial = n2)[n2]
##     }
##     else median(abs(x - center))
## }
## <bytecode: 0x5a0fdd8>
## <environment: namespace:stats>
```

1.1.2.5.2 Example: Simulating Coin Tosses

What is the probability of exactly 3 heads in 10 coin tosses for a fair coin?

Using binomial identity:

```r
# $\binom{n}{k}p^{k}(1-p)^{(n-k)} = # \binom{10}{3}\left(\frac{1}{2}\right)^{3}\left(\frac{1}{2}\righ

choose(10,3)*(.5)^3*(.5)^7
```

```
## [1] 0.11719
```

Using binomial distribution density function:

```r
dbinom(prob = 0.5, size = 10, x = 3)
```

```
## [1] 0.11719
```

Using simulation (100,000 tosses):

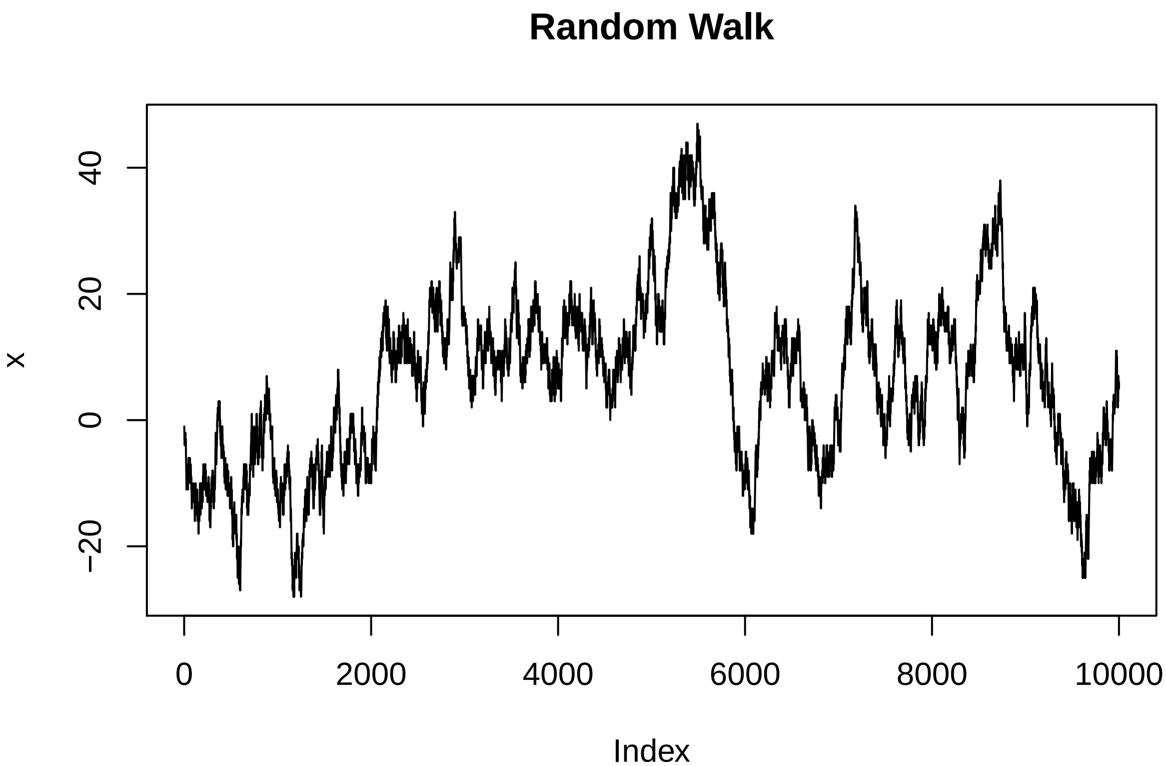
```
sum(replicate(100000,sum(rbinom(prob = 1/2, size = 10, 1)) == 3))/100000
```

```
## [1] 0.11802
```

1.1.2.5.3 Example: Random Walk

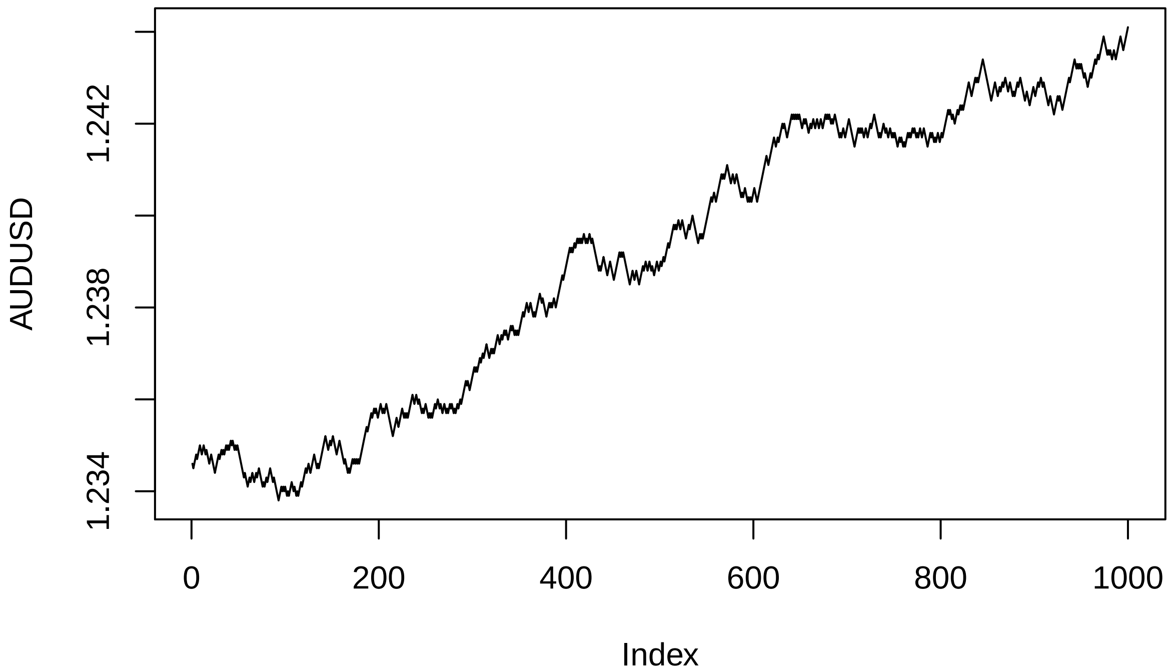
Generate 1000 up-down movements based on a fair coin toss and plot:

```
x <- (cumsum(ifelse(rbinom(prob = 0.5, size = 1, 10000) == 0,-1,1)))
plot(x, type = 'l', main = 'Random Walk')
```



1.1.2.5.4 Example: Generating Random Data

```
randomWalk <- function(N)(cumsum(ifelse(rbinom(prob = 0.5, size = 1, N) == 0,-1,1)))
AUDUSD <- 1.2345 + randomWalk(1000)*.0001
plot(AUDUSD, type = 'l')
```



1.1.2.5.5 Example: OANDA FX Data

```r
require(quantmod);require(TTR)
EURUSD <- getSymbols("EUR/USD", src="oanda", auto.assign=FALSE)
plot(EURUSD)
lines(EMA(EURUSD,10), col='red')
lines(EMA(EURUSD,30), col='blue')
```

1.1.2.5.6 Example: Connecting to kdb+

- kdb+ is an online database of open time-series data

```
Rorys-MacBook-Pro:kdb rorywinston$ <b>./rlwrap q/m32/q -p 5000</b>
KDB+ 3.1 2014.07.01 Copyright (C) 1993-2014 Kx Systems
m32/ 8()core 16384MB rorywinston rorys-macbook-pro.local 127.0.0.1 NONEXPIRE
Welcome to kdb+ 32bit edition
<b>q)\p</b>
5000i
<b>q) trades:([]time:100?.z.P;price:100?2.;
  side:100?`B`S;exch:100?`CNX`RTM`EBS;sym:100?`EURUSD`AUDUSD`GBPUSD)</b>
<b>q)10#trades</b>
time                          price     side exch sym
--------------------------------------------------------
2010.08.13D12:33:29.975458112 0.6019404 B    CNX  EURUSD
2001.11.24D20:53:58.972661440 0.7075032 S    CNX  EURUSD
2002.12.12D03:12:04.442386736 1.500898  S    CNX  GBPUSD
2002.02.12D21:48:33.887104336 0.6170263 S    EBS  AUDUSD
2014.05.01D06:59:44.647138496 0.8821325 S    EBS  GBPUSD
2010.12.06D15:30:14.928601664 1.094677  S    RTM  AUDUSD
2009.04.19D23:12:33.919967488 1.187474  B    RTM  GBPUSD
2008.07.18D22:13:25.681742656 0.1768144 B    EBS  GBPUSD
2010.08.22D10:16:15.261483520 0.3576458 S    EBS  AUDUSD
```

```
2010.02.28D13:49:33.686598976 1.526465  S    RTM  EURUSD
```

1.1.2.5.7 Example: Connecting to kdb+

```r
setwd("/Users/rorywinston/sandbox/kdb")
source("qserver.R")
open_connection("localhost", 5000)
trades <- k("select from trades")
head(trades)
                 time     price side exch    sym
1 2010-08-13 22:33:29 0.6019404    B  CNX EURUSD
2 2001-11-25 07:53:58 0.7075032    S  CNX EURUSD
3 2002-12-12 14:12:04 1.5008982    S  CNX GBPUSD
4 2002-02-13 08:48:33 0.6170263    S  EBS AUDUSD
5 2014-05-01 16:59:44 0.8821325    S  EBS GBPUSD
6 2010-12-07 02:30:14 1.0946771    S  RTM AUDUSD
```

kdb+ datatypes are converted to native R types

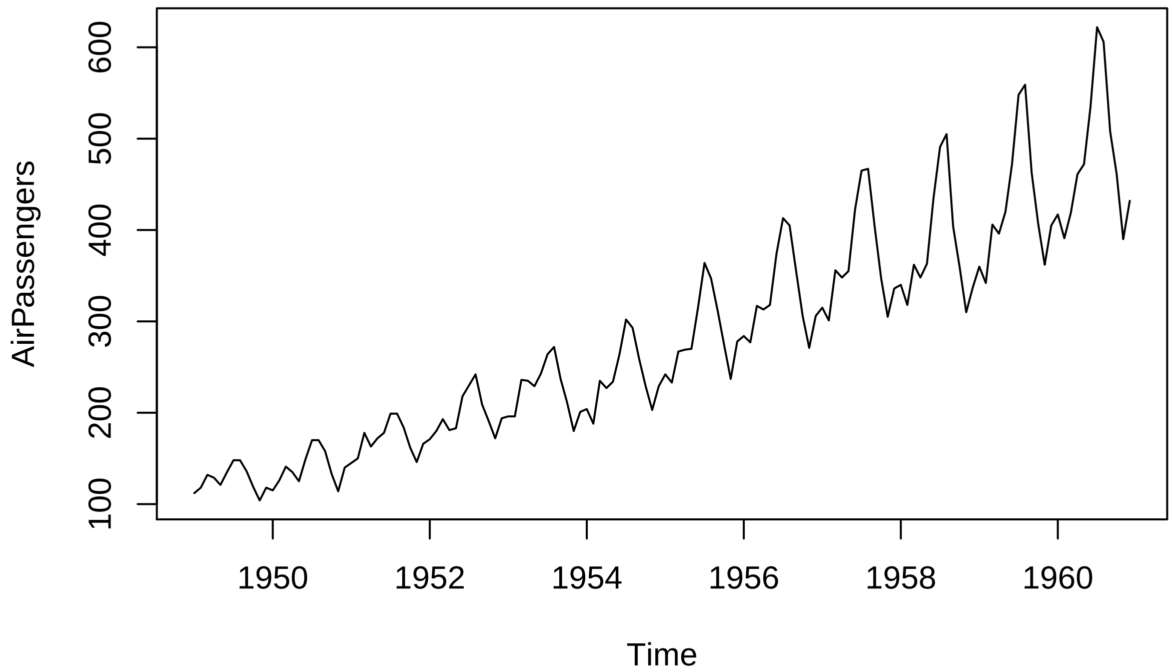
1.1.2.5.8 Example: Reading Log Data From File

```r
# Read file into data frame
logfile <- read.csv("/tmp/application.log", sep=",", header=FALSE)
# Set column descriptors
colnames(logfile) <- c("time","message","severity")
# Convert to native date/time
logfile$time <- as.POSIXct(strptime
                  (logfile$time, "%Y-%m-%d %H:%M:%OS"), tz="GMT")
```

1.1.2.5.9 Example: Using already existing Datasets

The famous 'Air passengers' dataset

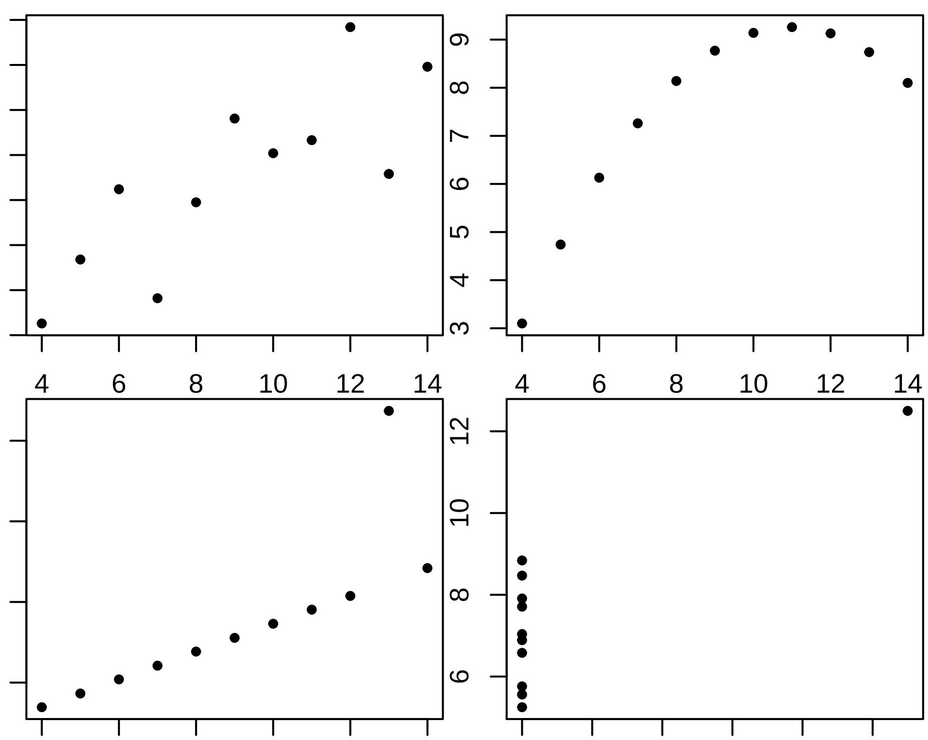
```r
plot(AirPassengers)
```



1.1.2.5.10 Example: Using Datasets:

- 'Anscombe Quartet':4 x-y datasets, same stats props, Yet quite different.

```
op <- par(mfrow = c(2,2),mar = rep(1,4))
with(anscombe,{plot(x1,y1,pch = 20);plot(x2,y2,pch = 20);
               plot(x3,y3,pch = 20);plot(x4,y4,pch = 20)})
```



```r
par(op)
```

1.1.2.5.11 Links

- http://www.r-project.org
- Rory Winston, for the Learning R intro
  - http://www.theresearchkitchen.com/archives/1017
- kdb+ is an online database of open time-series data
  - http://kx.com/kdb-plus.php