CWRU DSCI351-351M-451: Week01a-p R Intro, (CWRU, Pitt, UCF, UTRGV)

Profs: R. H. French, L. S. Bruckman, P. Leu, K. Davis, S. Cirlos

TAs: W. Oltjen, K. Hernandez, M. Li, M. Li, D. Colvin

06 September, 2022

Contents

2.1.3.0.1 Foundations, Practicum and Advanced

- Foundation Topics are the background fundamentals
- Practicum is real coding and data analysis
- Advanced are introductions to advanced topics

2.1.3.1 Syllabus

Day:Date	Foundation	Practicum	Reading	Due
w01a:Tu:8/30/22	ODS Tool Chain	R, Rstudio, Git		
w01b:Th:9/1/22	Setup ODS Tool Chain	Bash, Git, Slack, Agile	PRP4-33	LE1
w02a:Tu:9/6/22	What is Data Science	OIS:Intro2R, Git	PRP35-64	
w02b:Th:9/8/22	Summarizing Data	Intro2R	OIS1,2	
w02Pr:Fr:9/9/22			PRP65-93	**451 Update1**
w03a:Tu:9/13/22	Summarizing Data	Git, Rmds, Loops,	PRP94-116	LE2 **LE1 Due**
w03b:Th:9/15/22	Rand. Var. Normal Dist.	Data Analytic Style	OIS4	
w04a:Tu:9/20/22	Tidy Check Explore	Tidy GapMinder	EDA1-31	
w04b:Th:9/22/22	Inference, DSCI Process	Other Distrib. 7 ways	R4DS1-3	LE3 **LE2 Due**
w04Pr:Fr:9/23/22			EDA32-58	**451 Update2**
w05a:Tu:9/27/22	OIS4 Rand. Var.	EDA of PET Degr.	OIS5	
w05b:Th:9/29/22	OIS5 Found. of Infer.	Multivar Corr. Plot	R4DS4-6	
w05Pr:Fr:9/30/22				**451 RepOut1**
w06a:Tu:10/4/22	Pred., Algorithm, Model	Anscombe's Quartets	R4DS7-8	
w06b:Th:10/6/22	EDA stats, vis	Summ. Stats & Vis.	R4DS9-16	LE4 **LE3 Due**
w06Pr:Fr:10/7/22	Corr. Coeff. Pairs Plots			**451 Update3**
w07a:Tu:10/11/22	Confidence Intervals	Penguins	OIS6.1-2	**PeerRv1 Due**
w07b:Th:10/13/22	Midterm Rev.	Hypo.Test, Sampl. Dist.		
w08a:Tu:10/18/22	**MIDTERM**	**EXAM**		
w08b:Th:10/20/22	Programming & Coding	Coding Expect.		**LE4 Due**
w08Pr:Fr:10/21/22				**451 Update4**
Tu:10/24,25	**CWRU**	**FALL BREAK**	R4DS17-21	
w09b:Th:10/27/22	Cat. Inf. 1 & 2 propor.	Indep. Test,2-way tables	OIS6.3-4	LE5
w09Pr:Fr:10/28/22				**451 RepOut2**
w10a:Tu:11/1/22	Goodness of Fit, χ^2 test	t-tests 1&2 means	OIS7.1-4	
w10b:Th:11/3/22	Num. Infer, Cont. Tables	Stat. Power		
w10Pr:Fr:11/4/22				**451 Update5**
w11a:Tu:11/8/22	Sample & Effect Size	Stat. Power GGmap	OIS8	**PeerRv2 Due**
w11b:Th:11/10/22	Inf. 4 Regr, Test & Train	Curse of Dimen.	ISLR1,2.1,2	LE6 **LE5 Due**
w12a:Tu:11/15/22	Lin. Regr. Part 1	Residuals	OIS9	
w12b:Th:11/17/22	Lin. Regr. Part 2	Regr. Diagnostics		
w12Pr:Fr:11/18/22				**451 Update6**
w13a:Tu:11/22/22	Mult. Lin. Regr.	Var. & Mod. Selec.,	ISLR3.1	LE7 **LE6 due**
w13b:Th:11/24/22	Log. Regr.	GIS Trends	ISLR3.2	
w13Pr:Fr:11/25/22				**451 RepOut3**
w14a:Tu:11/23/22	Classificat., Sup. Lrning	Caret, Broom 4 modeling	ISLR4.1-3	
Th,Fr:11/24,25	**THANKSGIVIING**	**Vacation**		
w15a:Tu:11/29/22		Clustering		**PeerRv3 Due**
w15b:Th:12/1/22	Big Data Analytics	Dist. Comp., Hadoop		
w15SPr:Fr:12/2/22		Read Article by	Mirletz,2015	
w16a:Tu:12/6/22	Final Exam Review			
w15b:Th:12/8/22				**LE7 due**
Friday 12/12	**SemProj**	**Final Report**		**SemProj4 due**
Monday 12/19	**FINAL EXAM**	**12:00-3:00pm**	Nord 356	or remote

Figure 1: DSCI351-351M-451 Syllabus

- License: CC-BY-SA 4.0

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

2.1.3.1.1 setup for r-code chunks

2.1.3.2 A Simple Overview of R

- We'll look in more detail, going forward

2.1.3.2.1 Intro to some R: Data Types

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

```
  x <- 1
  class(x)
## [1] "numeric"
  y <- "Hello World"
  class(y)
## [1] "character"
  z <- TRUE
  class(z)
## [1] "logical"
  as.integer(z)
## [1] 1
```

2.1.3.2.2 Simple Types, and their class

2.1.3.2.3 Simple Types - Vectors

- The basic type unit in R is a vector

```
  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
```

2.1.3.2.4 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
```

2.1.3.2.5 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
```

2.1.3.2.6 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
```

2.1.3.2.7 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
```

2.1.3.2.8 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
```

2.1.3.2.9 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] -0.839434  0.656447  0.458536  0.359397 -0.051743 -1.446275 -2.066105
##  [8] -0.581393 -0.622831  1.127231
```

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

```
##  [1]  99.730 106.243 100.606 106.878 100.956  98.014 100.600  98.513  93.416
## [10] 103.956
```

- Many different distributions available

2.1.3.3 Data Frames

2.1.3.3.1 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)

```
  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
```

2.1.3.3.2 Data Frames can be indexed

- Data frames can be indexed like a vector or matrix:

```
  # 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
```

2.1.3.3.3 Generate a Data Frame

- Let's generate some dummy data:
  - Using data.frame

```
  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 2022-09-06 12:45:59 AUDUSD 1.2353 1.2359  RTM
## 2 2022-09-06 12:46:00 AUDUSD 1.2342 1.2348  RTM
## 3 2022-09-06 12:46:01 AUDUSD 1.2336 1.2347  EBS
## 4 2022-09-06 12:46:02 AUDUSD 1.2344 1.2364  CNX
## 5 2022-09-06 12:46:03 AUDUSD 1.2335 1.2360  RTM
```

2.1.3.3.4 Data Frames

- We can add/remove columns on the fly:

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

```
##                  time    sym    bid    ask exch     spread    mid
## 1 2022-09-06 12:45:59 AUDUSD 1.2353 1.2359  RTM 0.00064997 1.2356
## 2 2022-09-06 12:46:00 AUDUSD 1.2342 1.2348  RTM 0.00066595 1.2345
## 3 2022-09-06 12:46:01 AUDUSD 1.2336 1.2347  EBS 0.00110140 1.2341
## 4 2022-09-06 12:46:02 AUDUSD 1.2344 1.2364  CNX 0.00205366 1.2354
## 5 2022-09-06 12:46:03 AUDUSD 1.2335 1.2360  RTM 0.00244609 1.2347
```

```
## 6 2022-09-06 12:46:04 AUDUSD 1.2336 1.2356  EBS 0.00198945 1.2346
```

2.1.3.3.5 Operations on Data Frames

- Some basic operations on data frames:

```
  names(prices)
## [1] "time"   "sym"    "bid"    "ask"    "exch"   "spread" "mid"
  table(prices$exch)
##
## CNX EBS RTM
##  21  11  18
  summary(prices$mid)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.23    1.23    1.24    1.24    1.24    1.24
```

```
  `(`
```

2.1.3.3.6 These Operators Are Functions

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

```
  (1 + 2)
```

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

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

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

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

2.1.3.4 Examples in R

2.1.3.4.1 Example: Median Absolute Deviation

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

```
  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: 0x562591a6a7a8>
## <environment: namespace:stats>
```

2.1.3.4.2 Example: Simulating Coin Tosses

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

Using binomial identity:

$$\binom{n}{k} p^k (1-p)^{(n-k)} = \binom{10}{3} \left(\frac{1}{2}\right)^3 \left(\frac{1}{2}\right)^7$$

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

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

Using binomial distribution density function:

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

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

Using simulation (100,000 tosses):

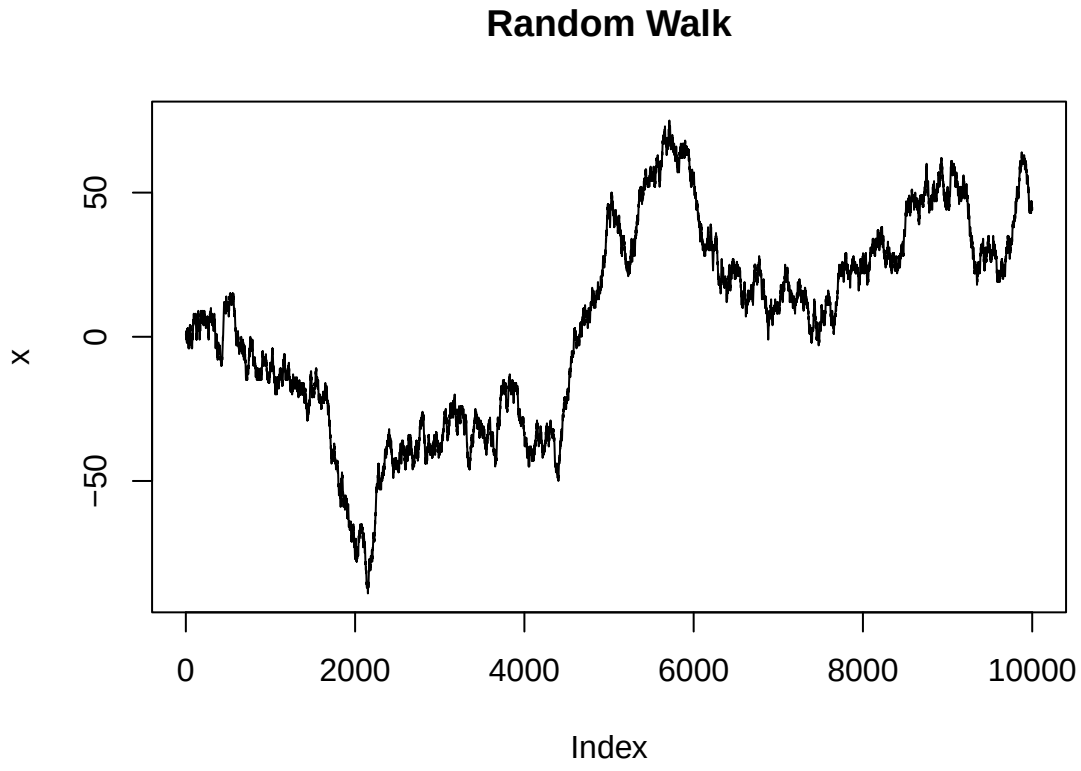
```
sum(replicate(100000,sum(rbinom(prob = 1/2, size = 10, 1)) == 3))/100000
```

```
## [1] 0.11854
```

2.1.3.4.3 Example: Random Walk

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

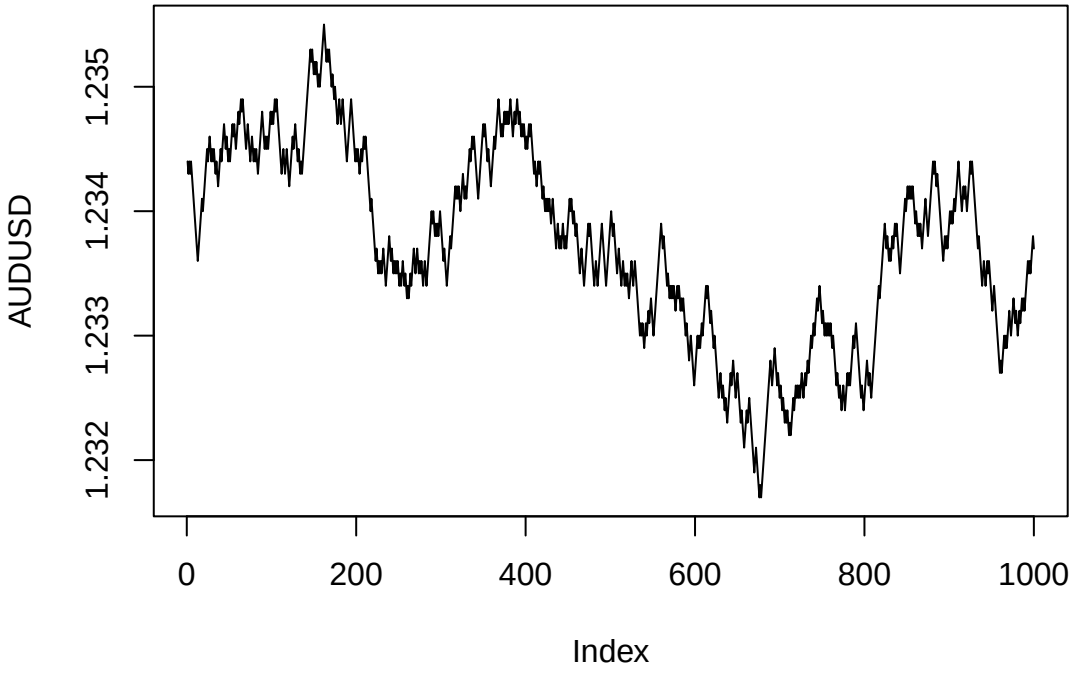
```
x <- (cumsum(ifelse(rbinom(prob = 0.5, size = 1, 10000) == 0,-1,1)))
plot(x, type = 'l', main = 'Random Walk')
```



```
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')
```

2.1.3.4.4 Example: Generating Random Data



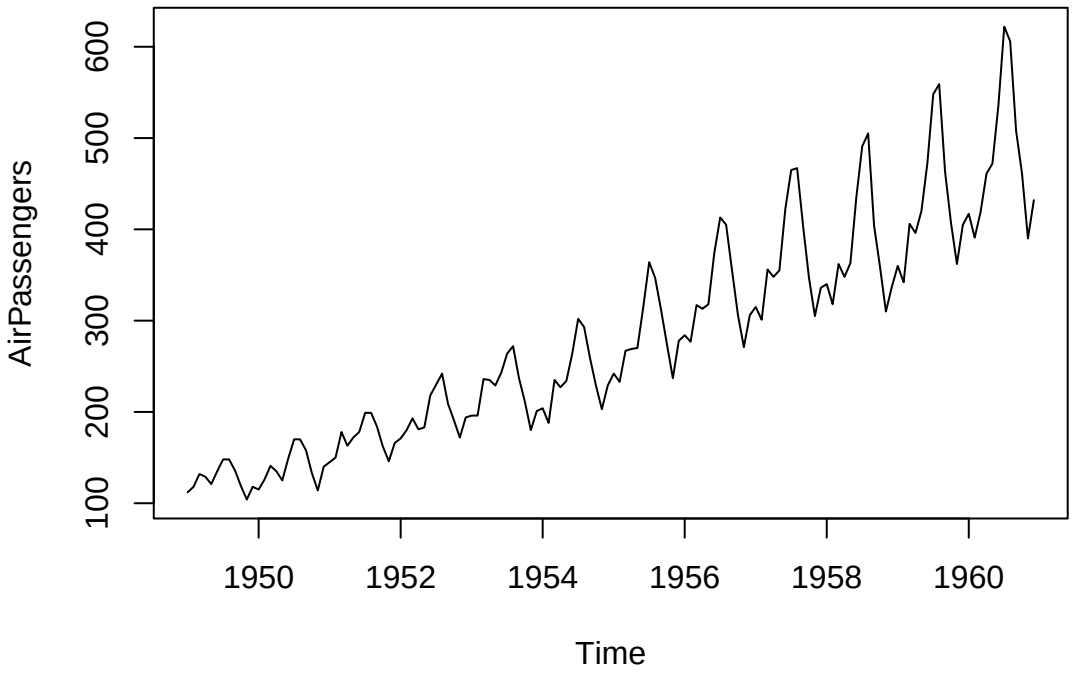

```
# Read file into data frame
logfile <- read.csv("/tmp/application.log", sep=",", header=FALSE)
# Set column descriptors
colnames(logfile) <- c("time","message","severity")
# Convert to native date/time
logfile$time <- as.POSIXct(strptime
                 (logfile$time, "%Y-%m-%d %H:%M:%OS"), tz="GMT")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

2.1.3.4.5 Example: Reading Log Data From File

2.1.3.4.6 Example: Using already existing Datasets

- The famous 'Air passengers' dataset

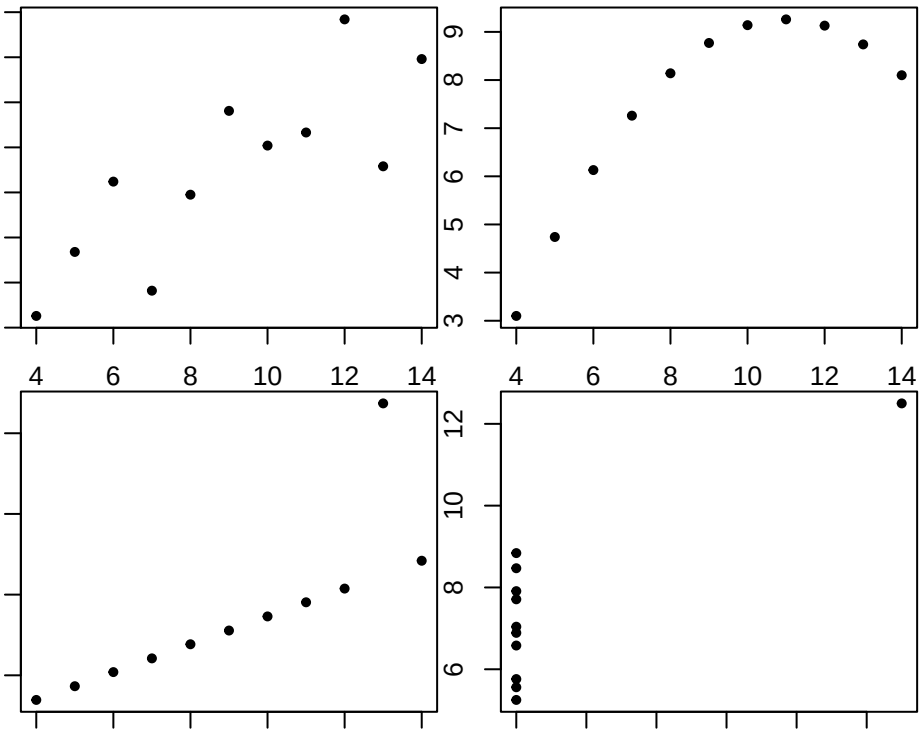
```
plot(AirPassengers)
```



2.1.3.4.7 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)})
```




```
par(op)
```

2.1.3.4.8 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