R Programming Language

林 建 甫
C.F. Jeff Lin, MD. PhD.

台北大學統計系助理教授
台北榮民總醫院生物統計顧問
美國密西根大學生物統計博士

What is R?

- Statistics package (a GNU project based on the S language)
  - Statistical environment
  - Graphics package
  - Programming language

What is R?

- R is object oriented
- Includes many statistical tools
- Can be used to share/reproduce analyses
- Many new packages being created - can be downloaded and installed easily
- Largely text-based interface
  - Note 1: > is the prompt
  - Note 2: R is case sensitive!!

What is R Not?

- a statistics software package
- menu-driven
- quick to learn
- a program with a complex graphical interface

Why R?

- R was written by scientists to be used by scientists.
- R is a great language.
- Runs on Windows, Mac, Unix, Linux, ...
- Easy to install.
- R is free.
- Hundreds(!) of packages available.
- Papers are often published with a R package.

Why R?

- Great help and documentation.
- online mail support.
- A great community – friendly and helpful people...
- Easy to call Fortran, C, Java, ... libraries
- Everyone else is and will be using R *(-Yes, I'm willing to bet!)*

S Programming Language

- The S programming language was developed at AT&T Bell Labs, for internal use by a group of statisticians who wanted an interactive, graphics environment that encouraged development of new statistical techniques.

S Programming Language

- As the program spread through universities, many people appreciated the same qualities as the AT&T researchers, and the language became quite popular among academic statisticians.

Strengths and Weaknesses of R

Strengths

- Highly extensible and flexible
- Implementation of modern statistical methods
- Strong user community
- Moderately flexible graphics with intelligent defaults

Strengths and Weaknesses of R

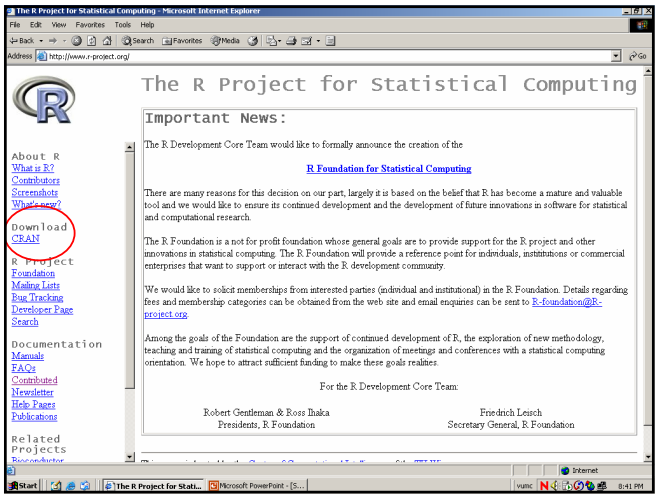
Weaknesses

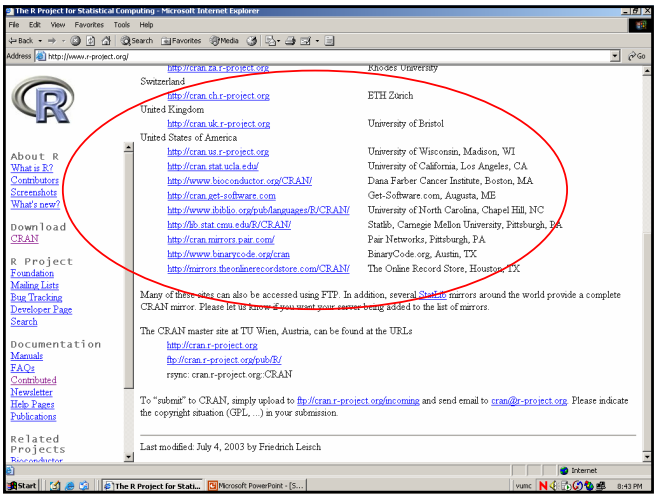
- Slow or impossible with large data sets
- Non-standard programming paradigms
- Runs on limited platforms
- S-Plus performs poorly in case of using extensive loop

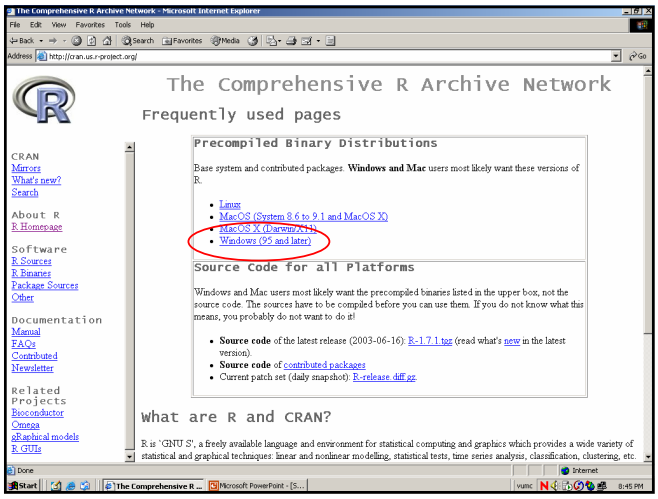
Let's look at R

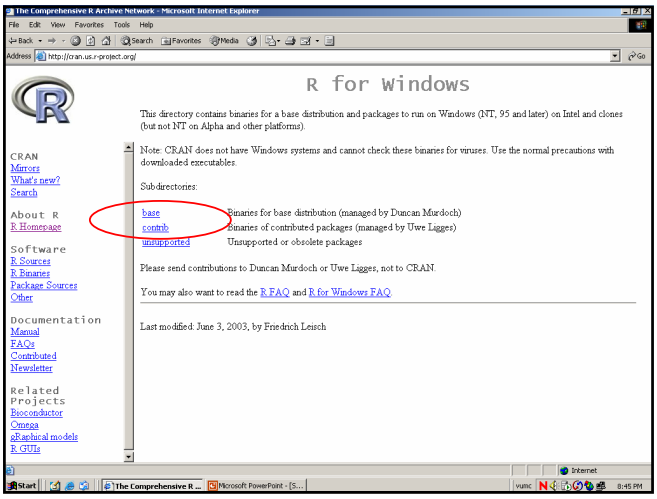
Installing R

- www.r-project.org/
- download from CRAN
- select a download site
- download the base package at a minimum
- download contributed packages as needed

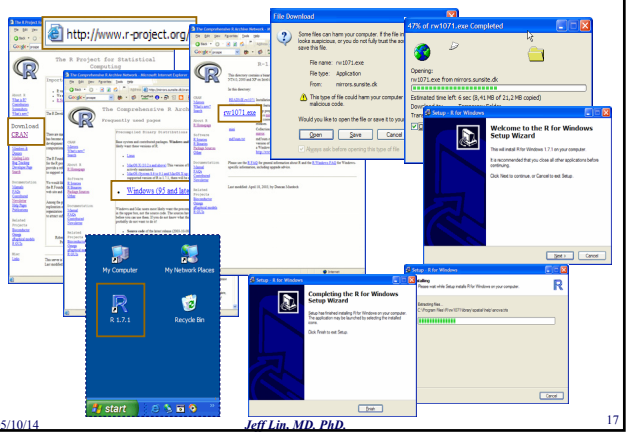






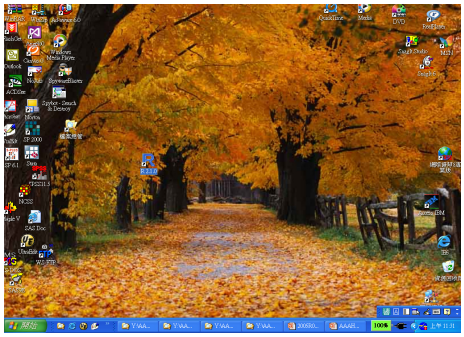


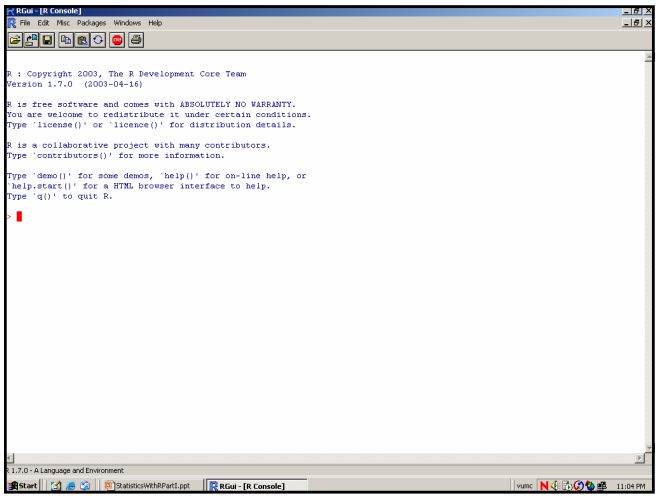
Installing R on Windows



Starting and Quitting R

- click on the R icon
- q()





Getting Started

- Starting R
- Getting help
  - > help ( ) *# provides help on how to use 'help'*
  - > help (*topic*) *# provides help on a specific topic*
  - > help.start ( ) *# brings you to a web interface to the # R documentation*

Starting and Quitting R

```
R : Copyright 2003, The R Development Core Team
Version 1.7.1  (2003-06-16)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type `license()' or `licence()' for distribution details.

R is a collaborative project with many contributors.
Type `contributors()' for more information.

Type `demo()' for some demos, `help()' for on-line help, or
`help.start()' for a HTML browser interface to help.
Type `q()' to quit R.

> 1+2+3
[1] 6
> q()
Save workspace image? [y/n/c]: n
```


Change Working Directory

```
# Create a folder in C:\temp\Rdata first
setwd("C://temp//Rdata")

#Create a folder in C:\temp\Rdata\Biost first
setwd("C://temp//Rdata//Biost")

#Create a folder in C:\temp\Rdata\Reg first
setwd("C://temp//Rdata//Reg")
```


Help

help ( ) is an R function
help.start( )

R functions take arguments (pieces of information that you put into the function which go in between brackets and are separated by commas) and can perform a range of tasks. In the case of the 'help' function the task is to display information from the R documentation files.

Getting Help

- **Browse the help**

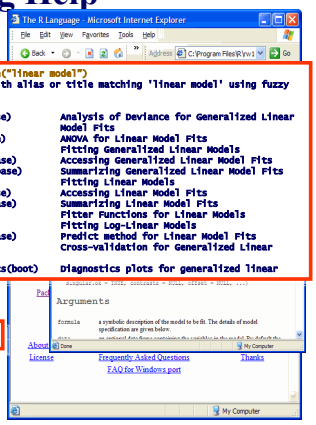
```
> help.start()
```

- **Getting help on a sp**

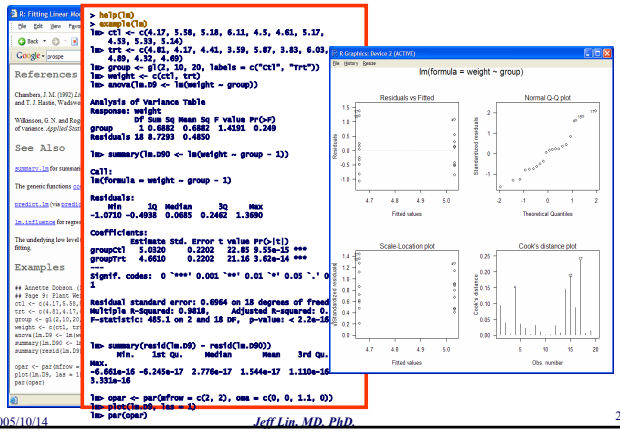
```
> help(lm)
> ?lm
```

- **Searching for help**

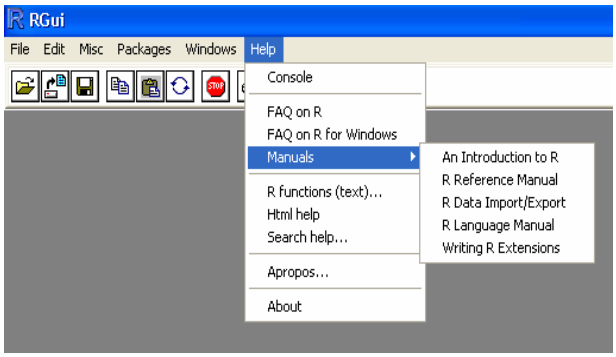
```
> help.search("linear model")
```



Running Help Examples



R References

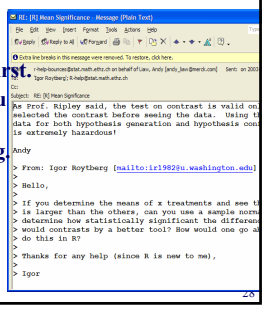


Online Helps

- R FAQ
- Mailing Lists (listserv)
  - Online r-help

r-help Mailing List

- **Ask questions on the r-help mailing list.**
- **Sign up on *http://www.r-project.org/***
- **Things to think of *before* sending a question:**
  - **Make sure you read the help/docs first.**
  - **Try to add a code example what you are trying to do.**
  - **Tell what version of R you are using**



Office Documents

- From R website under "Documentation"
  - "Manual" is the listing of official R documentation
    - An Introduction to R
    - R Language Definition
    - Writing R Extensions
    - R Data Import/Export
    - R Installation and Administration
    - The R Reference Index

Contributed Online Documents

Documents with greater than 100 pages:

- **"Using R for Data Analysis and Graphics - Introduction, Examples and Commentary"** by John Maindonald.
- **"Simple R"** by John Verzani.
- **"Practical Regression and Anova using R"** by Julian Faraway.
- **"An Introduction to S and the Hmisc and Design Libraries"** by Carlos Alzola and Frank E. Harrell.
- **"Statistical Computing and Graphics Course Notes"** by Frank E. Harrell.

Contributed Online Documents

Documents with less than 100 pages:

- **"R for Beginners"** by Emmanuel Paradis.
- **"Notes on the use of R for psychology experiments and questionnaires"** by Jonathan Baron and Yuelin Li.
- **"R for Windows Users (version 2.0)"** by Ko-Kang Wang.
- **"Building Microsoft Windows Versions of R and R packages under Intel Linux"** by Jun Yan and A. J. Rossini.
- **"A Guide for the Unwilling S User"** by Patrick Burns.
- **"The R language — a short companion"** by Marc Vandemeulebroecke.
- **"Fitting Distributions with R"** by Vito Ricci.
- **"Econometrics in R"** by Grant Farnsworth.
- **"The Friendly Beginners' R Course"** by Topy Marthews.

Contributed Online Documents

Short Documents and Reference Cards:

- **"R reference card"** by Jonathan Baron.
- **"R and Octave"** by Robin Hankin, a reference sheet translating between the most common Octave (or Matlab) and R commands.
- **"Time series reference card"** by Vito Ricci.
- **"R reference card"** by Tom Short.

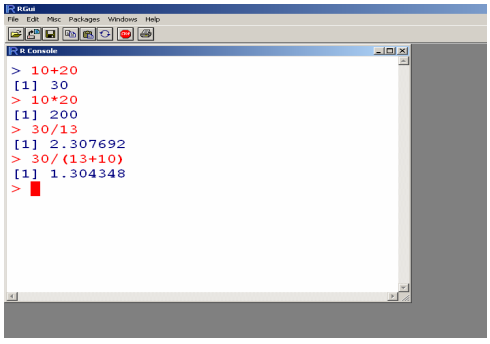
Reference Books

- John M. Chambers. *Programming with Data*. Springer, New York, 1998. ISBN 0-387-98503-4.
- Peter Dalgaard. *Introductory Statistics with R*. Springer, 2002. ISBN 0-387-95475-9.
- John Maindonald and John Braun. *Data Analysis and Graphics Using R*. Cambridge University Press, Cambridge, 2003. ISBN 0-521-81336-0.
- John Verzani. *Using R for Introductory Statistics*. Chapman & Hall/CRC, Boca Raton, FL, 2005. ISBN 1-584-88450-9.
- Michael J. Crawley. *Statistics: An Introduction using R*. Wiley, 2005. ISBN 0-470-02297-3.
  Venables & Ripley (2004) *Modern Applied Statistics with S.* New York: Springer-Verlag.
- Chambers (1998). *Programming With Data: A guide to the S language.* New York: Springer-Verlag.

R Showcases

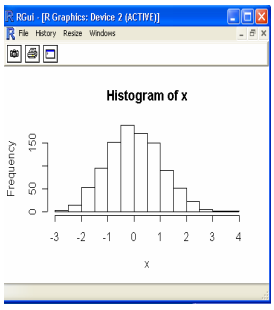
R as calculator

R will evaluate basic calculations which you type into the console (input window)

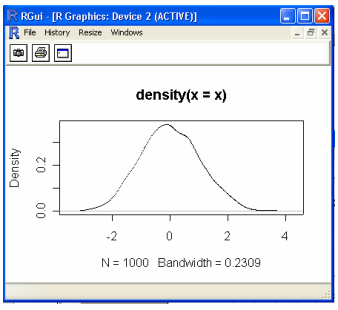


R is a Graphing Device

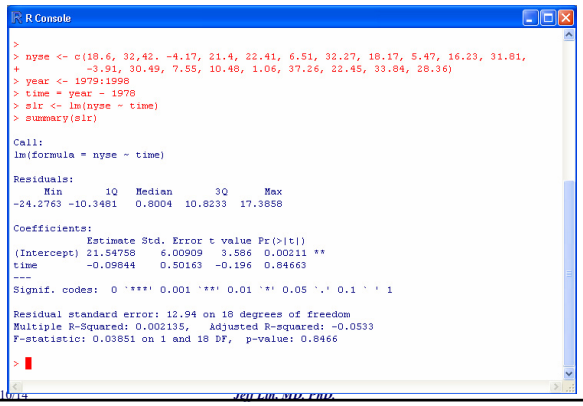
```
> x <- rnorm(1000, 0, 1)
> hist(x)
```




```
> plot(density(x))
>
```



R is a Statistics Package



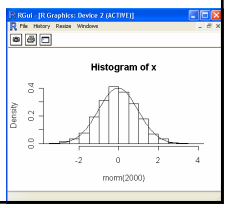

```
> nyse <- c(10.6, 32,42. -4.17, 21.4, 22.41, 6.51, 32.27, 18.17, 5.47, 16.23, 31.81,
+          -3.91, 30.49, 7.55, 10.48, 1.06, 37.26, 22.45, 33.84, 28.36)
> year <- 1979:1998
> time = year - 1976
> slr <- lm(nyse ~ time)
> summary(slr)

Call:
lm(formula = nyse ~ time)

Residuals:
     Min       1Q   Median       3Q      Max
-24.2763 -10.3481   0.8004  10.8233  17.3858

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 21.54758    6.00909   3.586  0.00211 **
time        -0.09844    0.50163  -0.196  0.84663
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 12.94 on 18 degrees of freedom
Multiple R-Squared: 0.002135,   Adjusted R-squared: -0.0533
F-statistic: 0.03851 on 1 and 18 DF,  p-value: 0.8466

>
```


R is a Simulator

```
> # rolls[1, ] first n tosses of two die
> # rolls[2, ] second n tosses of two die
> n <- 2000
> rolls <- matrix(ceiling(6*runif(2*n)), 2, n)
> # or rolls <- matrix(sample(1:6, 2*n, replace=T), 2, n)
> tosses <- apply(rolls, 2, sum)
> # frequency of observed outcomes
> o <- table(tosses) / n
> e <- expected frequency of outcomes
Error: syntax error
> e <- c(1:6, 5:1) / 36
> t(round(cbind(o, e), 4)
+ )
      2      3      4      5      6      7      8      9     10     11     12
o 0.0215 0.0625 0.0780 0.1025 0.1340 0.1650 0.1470 0.1130 0.0830 0.0620 0.0315
e 0.0278 0.0556 0.0833 0.1111 0.1389 0.1667 0.1389 0.1111 0.0833 0.0556 0.0278
>
```


R is a Programming Language

```
> hist.w.normal <-
+ function(x, xlab = deparse(substitute(x)), ...)
+ {
+     h = hist(x, plot=F, ...)
+     m = mean(x)
+     s = sd(x)
+     ylim = range(0, h$density, dnorm(0, sd=s))
+     hist(x, freq=F, ylim=ylim, xlab=xlab, ...)
+     curve(dnorm(x, m, s), add=T)
+ }
>
> hist.w.normal(rnorm(2000)
+ )
>
```



R as Objective-Oriented Language

R Basics

- objects
- naming convention
- assignment
- functions
- workspace
- history

Objects

- names
- types of objects: vector, factor, array, matrix, data.frame, ts, list
- attributes
  - mode: numeric, character, complex, logical
  - length: number of elements in object
- creation
  - assign a value
  - create a blank object

Naming Convention

- must start with a letter (A-Z or a-z)
- can contain letters, digits (0-9), and/or periods "."
- case-sensitive
  - `mydata` different from `MyData`
- do not use use underscore "_"

Caution

- Do not use the underscore "_" for names.
- R is case sensitive.
- Try not to use the "=" as it might be confusing and cause problem.
- Remember that if you use "return" command anywhere in your program then it will stop executing at there.

Assignment

- Simple operations

- Add: 10 + 20
- Multiply: 10 * 20
- Divide: 10/20
- Raise to a power: 10 ** 20
- Modulo: 10%%20
- Integer division: 10%/%4

R: Logical and Relational Operators

- `==` Equal to
- `!=` Not equal to
- `<` Less than
- `>` Greater than
- `<=` Less than or equal to
- `>=` Greater than or equal to
- `is.na(x)` Missing?
- `&` Logical AND
- `|` Logical OR
- `!` Logical NOT

Variables and Assignment

```
> a <- 49
> sqrt(a)
[1] 7
```

numeric

```
> b <- "The dog ate my homework"
> sub("dog","cat",b)
[1] "The cat ate my homework"
```

character string

```
> c <- (1+1==3)
> c
[1] FALSE
> as.character(b)
[1] "FALSE"
```

logical

Basic (Atomic) Data Types

- Logical

```
> x <- T; y <- F
> x; y
[1] TRUE
[1] FALSE
```

- Numerical

```
> a <- 5; b <- sqrt(2)
> a; b
[1] 5
[1] 1.414214
```

- Character

```
> a <- "1"; b <- 1
> a; b
[1] "1"
[1] 1
> a <- "character"
> b <- "a"; c <- a
> a; b; c
[1] "character"
[1] "a"
[1] "character"
```

NA, NaN, and Null

- NA or "Not Available"
  - Applies to many modes – character, numeric, etc.
- NaN or "Not a Number"
  - Applies only to numeric modes
- NULL
  - Lists with zero length

Missing Values

- R is designed to handle statistical data and therefore predestined to deal with missing values
- Numbers that are "not available"

```
> x <- c(1, 2, 3, NA)
> x + 3
[1] 4  5  6 NA
```

- "Not a number"

```
> log(c(0, 1, 2))
[1]      -Inf 0.0000000 0.6931472
> 0/0
[1] NaN
```


R: MissingValues

- Variables of each data type can also take the value NA (for Not Available)
  - NA is not the same as 0
  - NA is not the same as " " (blank, or empty string)
  - NA is not the same as FALSE
- Any computations involving NA *may or may not* produce NA as a result:

```
> 1+NA
[1] NA
> max(c(NA, 4, 7))
[1] NA
> max(c(NA, 4, 7), na.rm=T)
[1] 7
```


Getting Stuck at "+" Prompt

- If the "+" prompt continues after hitting return, then enter many ")" to get the ">" prompt
- Then start your expression again

```
> sqrt(
+
+ )))))
Error in parse(text = txt): Syntax error:
        No opening parenthesis, before
        ")" at this point:
sqrt(
))
Dumped
> sqrt(100)
```


A First Example

Vectors:

```
> x <- c(6,5,4,3,2,1)
> x
[1] 6 5 4 3 2 1
> sum(x)
[1] 21
> x[c(1,3,5)]
[1] 6 4 2
> x <- 6:1
> x
[1] 6 5 4 3 2 1
> x[1:3]
[1] 6 5 4
> x[1:3]+x[6:4]
[1] 7 7 7
> y <- c(0,10)
> x+y
[1]  6 15  4 13  2 11
```

Matrices (default is to fill column by column):

```
> x <- matrix(1:18, nrow=3)
> x
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7   10   13   16
[2,]    2    5    8   11   14   17
[3,]    3    6    9   12   15   18
> x[,3]
[1] 7 8 9
> x[-1,4:6]
     [,1] [,2] [,3]
[1,]   11   14   17
[2,]   12   15   18
> str(x)
 int [1:3, 1:6] 1 2 3 4 5 6 7 8 9 10
...
```


R is multiplying elementwise!

Matrices:

```
> X <- matrix(1:9, ncol=3,
byrow=TRUE)
> X
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> I <- diag(1, nrow=3)
> I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> I * X
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    5    0
[3,]    0    0    9
> I %*% X
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Why? Because it turns out that it is more common than vector and matrix multiplications.

R is looping over missed values - really useful indeed!

Vectors:

```
> x <- 7:1
> x
[1] 7 6 5 4 3 2 1
> y <- 1:3
> y
[1] 1 2 3
> x + y
[1] 8 8 8 5 5 5 2
```

R will expand the shortest vector before adding!

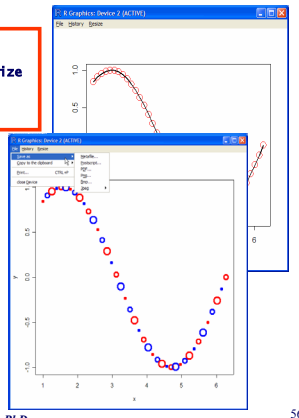
```
# Zero out every
2nd:
> x <- 3:8
> y <- c(1,0)
> x * y
[1] 3 0 5 0 7 0
```

Matrices:

```
> x <- matrix(1:16, nrow=3)
Warning message:
Replacement length not a multiple of
the elements to replace in matrix(...)
> x
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7   10   13   16
[2,]    2    5    8   11   14    1
[3,]    3    6    9   12   15    2
> y <- c(-1,1)
# Multiply ("moving down the columns")
> x * y
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   -1    4   -7   10  -13   16
[2,]    2   -5    8  -11   14   -1
[3,]   -3    6   -9   12  -15    2
```


A First Plot Example

```
> x <- seq(from=1, to=2*pi, length=41)
> y <- sin(x)
# Scatter plot, with red data points of size
2
> plot(x,y, col="red", cex=2)
# A line with double width through data
points
> lines(x,y, lwd=2)
```



Loop over plot attributes:

```
> col <- c("red", "blue")
> plot(x,y, col=col, cex=1:3,
lwd=4)
```

Write last plot to file:

```
> dev.print(png, "sin.png")
> dev.print(postscript,
"sin.ps")
```

(also in menus on Windows)

Questions?!

Putting Names on Things - helps you avoid stupid bugs etc.

Vectors:

```
> x <- c(87,76.3,1.67)
> x
[1] 87.0 76.3 1.67
> names(x) <- c("age", "weight",
"height")
> x
   age weight height
  87.0   76.3   1.67
# Alternatively
> x <- c(age=87, weight=76.3,
height=1.67)
```

Why?

```
> x["age"]
[1] 87.0
> bmi <- x["weight"]/x["height"]^2
```

cf.

```
> x[1]
[1] 87.0
> bmi <- x[2]/x[3]^2
```


Putting Names on Things

Matrices:

```
> x1 <- c(87,76.3,1.67)
> x2 <- c(78,96.3,1.84)
> x3 <- c(45,62.9,1.54)
> x <- matrix(c(x1,x2,x3), nrow=3, byrow=TRUE)
> x
     [,1] [,2] [,3]
[1,]   87 76.3 1.67
[2,]   78 96.3 1.84
[3,]   45 62.9 1.54
> colnames(x) <- c("age", "weight", "height")
> rownames(x) <- c("jon", "kim", "dan")
> x
    age weight height
jon  87   76.3   1.67
kim  78   96.3   1.84
dan  45   62.9   1.54
> x["jon",]
 age weight height
 87.0   76.3   1.67
> x[,c("weight","age")]
    weight age
jon   76.3  87
kim   96.3  78
dan   62.9  45
> bmi <- x[,"weight"]/x[,"height"]^2
> bmi
     jon      kim      dan
27.35846 28.44400 26.52218
```



```
> x <- c(a=1:2, b=3,
c=5:8)
> x
a1 a2  b c1 c2 c3 c4
 1  2  3  5  6  7  8
> x <- list(a=1:2, b=3,
c=5:8)
> x
$a
[1] 1 2
$b
[1] 3
$c
[1] 5 6 7 8
> x$a
[1] 1 2
> x[["a"]]
[1] 1 2
> x[2:3]
$b
[1] 3
$c
[1] 5 6 7 8
> x[c("b","c")]
$b
[1] 3
$c
[1] 5 6 7 8
```

Lists

Vectors and matrices can only contain *one* type of data *at the time*, e.g. either numbers or strings, but lists can carry mixed types:

```
> x <- list(a=1:4, b=4:6+2i,
src="GenePix")
> x
$a
[1] 1 2 3 4
$b
[1] 4+2i 5+2i 6+2i
$src
[1] "GenePix"
```

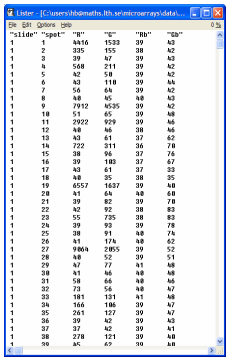
Data Frames

Data frames are very powerful. Simply speaking you can treat them as both lists and matrices:

```
> df <- data.frame(name=c("jon","kim","dan"), age=c(87,78,45),
weight=c(76.3,96.3,62.9), height=c(1.67,1.84,1.54))
> df
  name age weight height
1  jon  87   76.3   1.67
2  kim  78   96.3   1.84
3  dan  45   62.9   1.54
> df$weight
[1] 76.3 96.3 62.9
> df[,c("name","age")]
  name age
1  jon  87
2  kim  78
3  dan  45

> as.matrix(df)
  name  age  weight height
1 "jon" "87" "76.3" "1.67"
2 "kim" "78" "96.3" "1.84"
3 "dan" "45" "62.9" "1.54"
```


Importing Data

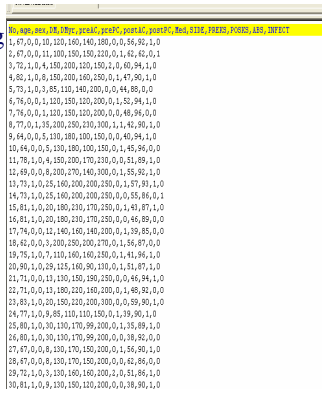
Data in text (ASCII) files can be imported using `read.table()`:



```
> df <- read.table("foo.txt", header=TRUE)
> dim(df)
[1] 221952      6
> summary(df)
     slide             spot              R
 Min.   :1.00   Min.   :    1   Min.   :   30.0
 1st Qu.:1.75   1st Qu.:13873   1st Qu.:   60.0
 Median :2.50   Median :27745   Median :  102.0
 Mean   :2.50   Mean   :27745   Mean   :  487.5
 3rd Qu.:3.25   3rd Qu.:41616   3rd Qu.:  233.0
 Max.   :4.00   Max.   :55488   Max.   :65211.0
       G                Rb                Gb
 Min.   :   24.0   Min.   :  32.00   Min.   : 24.00
 1st Qu.:   38.0   1st Qu.:  40.00   1st Qu.: 26.00
 Median :   59.0   Median :  49.00   Median : 32.00
 Mean   :  194.7   Mean   :  51.85   Mean   : 31.92
 3rd Qu.:  110.0   3rd Qu.:  64.00   3rd Qu.: 36.00
 Max.   :62341.0   Max.   :1234.00   Max.   :255.00
> str(df)
`data.frame':   221952 obs. of  6 variables:
 $ slide: int  1 1 1 1 1 1 1 1 1 1 ...
 $ spot : int  1 2 3 4 5 6 7 8 9 10 ...
 $ R    : int  4416 335 39 568 42 43 56 40 7912 51 ...
 $ G    : int  1533 155 47 211 50 110 64 45 4535 65 ...
 $ Rb   : int  39 38 39 39 39 39 39 40 39 39 ...
 $ Gb   : int  43 42 43 42 42 44 42 43 42 48 ...
```

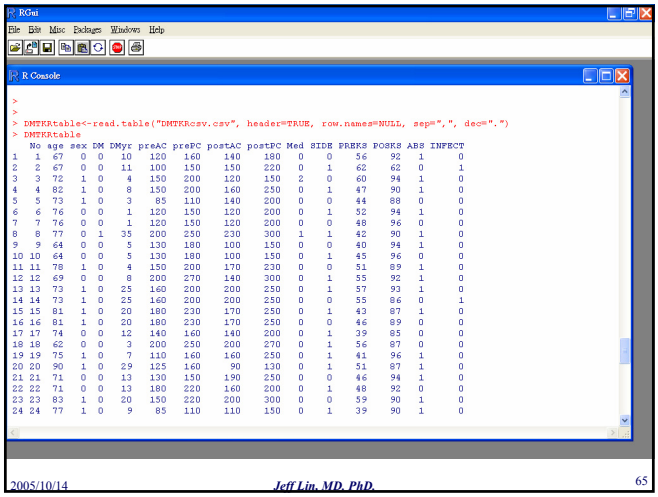

Importing Data

- Data file in the working directory



Importing Data

```
> # Data Managements
> setwd("C://temp//Rdata")
> DMTKRtable<-read.table("DMTKRcsv.csv",
      header=TRUE, row.names=NULL, sep=",", dec=".")
> DMTKRtable
```

```
> DMTKRtable<-read.table("DMTKRcsv.csv", header=TRUE, row.names=NULL, sep=",", dec=".")
> DMTKRtable
   No age sex DM DMyr preAC prePC postAC postPC Med SIDE PRRKS POSKS ABS INFECT
1   1  67   0  0   10   120   160   140   180   0    0    56    92   1      0
2   2  67   0  0   11   100   150   150   220   0    1    62    62   0      1
3   3  72   1  0    4   150   200   120   150   2    0    60    94   1      0
4   4  82   1  0    8   150   200   160   250   0    1    47    90   1      0
5   5  73   1  0    3    85   110   140   200   0    0    44    88   0      0
6   6  76   0  0    1   120   150   120   200   0    1    52    94   1      0
7   7  76   0  0    1   120   150   120   200   0    0    48    96   0      0
8   8  77   0  1   35   200   250   230   300   1    1    42    90   1      0
9   9  64   0  0    5   130   180   100   150   0    0    40    94   1      0
10 10  64   0  0    5   130   180   100   150   0    1    45    96   0      0
11 11  78   1  0    4   150   200   170   230   0    0    51    89   1      0
12 12  69   0  0    8   200   270   140   300   0    1    55    92   1      0
13 13  73   1  0   25   160   200   200   250   0    1    57    93   1      0
14 14  73   1  0   25   160   200   200   250   0    0    55    86   0      1
15 15  81   1  0   20   180   230   170   250   0    1    43    87   1      0
16 16  81   1  0   20   180   230   170   250   0    0    46    89   0      0
17 17  74   0  0   12   140   160   140   200   0    1    39    85   0      0
18 18  62   0  0    3   200   250   200   270   0    1    56    87   0      0
19 19  75   1  0    7   110   160   160   250   0    1    41    96   1      0
20 20  90   1  0   29   125   160    90   130   0    1    51    87   1      0
21 21  71   0  0   13   130   150   190   250   0    0    46    94   1      0
22 22  71   0  0   13   180   220   160   200   0    1    48    92   0      0
23 23  83   1  0   20   150   220   200   300   0    0    59    90   1      0
24 24  77   1  0    9    85   110   110   150   0    1    39    90   1      0
```


Importing Data

```
> setwd("C://temp//Rdata")
> DMTKRcsv<-read.csv("DMTKRcsv.csv",
      header = TRUE, sep = ",", dec=".")
> DMTKRcsv
> attach(DMTKRcsv)

> scan(file = "DMTKRcsv.csv", skip=1, sep = ",",
      dec = ".")
```

Exporting Data

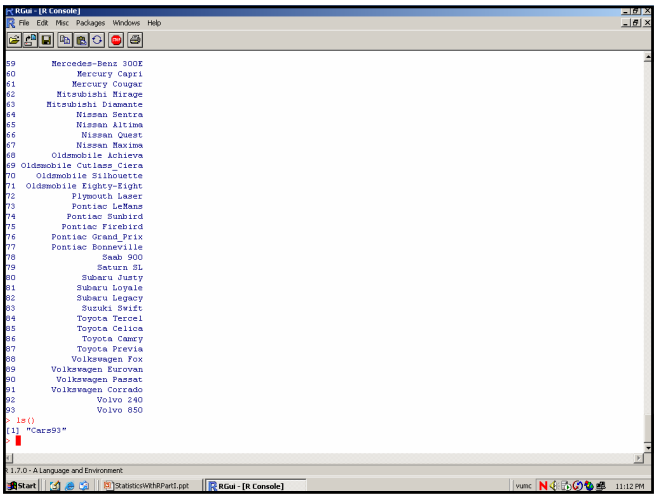
```
> #write data out
> cat("2 3 5 7", "11 13 17 19", file="ex.dat", sep="\n")
# Read in ex.dat again
> scan(file="ex.dat", what=list(x=0, y="", z=0),
      flush=TRUE)
```

```
df <- data.frame(a = I("a \" quote"))
write.table(df)
write.table(df, qmethod = "double")
write.table(df, quote = FALSE, sep = ",")
```


Workspace & History

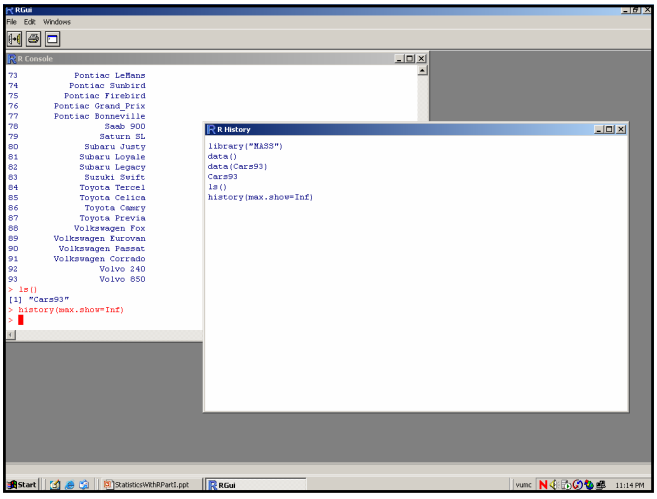
Workspace

- during an R session, all objects are stored in a temporary, working memory
- list objects
  - `ls()`
- remove objects
  - `rm()`
- objects that you want to access later must be saved in a "workspace"
  - from the menu bar: File->save workspace
  - from the command line:
    `save(x,file="MyData.Rdata")`



History

- command line history
- can be saved, loaded, or displayed
  - `savehistory(file="MyData.Rhistory)`
  - `loadhistory(file="MyData.Rhistory)`
  - `history(max.show=Inf)`
- during a session you can use the arrow keys to review the command history



R: Session Management

- Your R objects are stored in a *workspace*
- To list the objects in your workspace: `> ls()`
- To remove objects you no longer need:
  `> rm(weight, height, bmi)`
- To remove ALL objects in your workspace:
  `> rm(list=ls())` or use Remove all objects in the Misc menu
- To save your workspace to a file, you may type
  `> save.image()` or use Save Workspace... in the File menu
- The default workspace file is called .RData

R: Saving Your Work and Quitting

- You may also save your command history by using Save History... in the File menu
- When you have finished your R session, you can quit by typing the R command `> q()` or by clicking on the X to close the window
- Don't forget the parentheses!
- You will be asked if you want to save the workspace image; generally, you will say 'yes' so that R will save the data there for you (for these practice sessions, you can say no)

Standard Units

- Standard units (SUs), also sometimes called *z-scores*, tell how many SDs above or below the mean (average) a particular observation is
- To convert a value x into standard units z, subtract the mean from the value, then divide that result by the SD:

$$z = (x - \text{mean})/SD$$

- Subtracting the average from each variable value x has the effect of making the average of the z's be 0; dividing by the SD makes the SD of the z's be 1.

Why Standard Units?

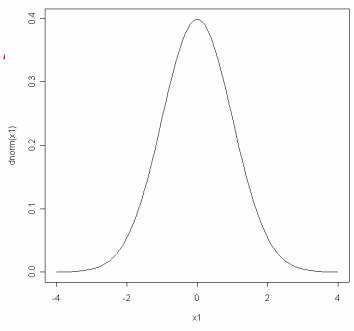
- For *comparing* two (or more) sets of data, it is often useful that values be expressed in the same units
- Detection of suspected *outliers* is often carried out in terms of standard units
- Standard units are important for using the *normal distribution*

R: Functions for Normals

- Generate pseudo-random normals: `> rnorm(…)`
- Probability to the *left* of a value: `> pnorm(…)`
- Quantiles: `> qnorm(…)`
- (Height of the curve: `> dnorm(…)`)
- These 4 fundamental items can be computed for a number of common distributions (e.g. binomial, t, chi-square, etc.): `rbinom()`, `qt()`, `pchisq()`…

R: Normal Curve Plot

```
> x1<-seq(-4,4,.1)
> plot(x1,dnorm(x1),
    type="l")
```



Exercises: Experiment with These 4 Functions

- simulate (and save) 200 standard normals
- simulate 200 normals with mean 30 and SD 4
- find the chance that a standard normal is less than 1.5; bigger than 1.5; less than -5; between -2 and 1
- find the 30th and 75th percentiles for the standard normal distribution; for a normal distribution with mean 30 and SD 4
- guess how you might be able to make a plot of the normal curve

Questions?!

Introduction to Basic Descriptive Statistics

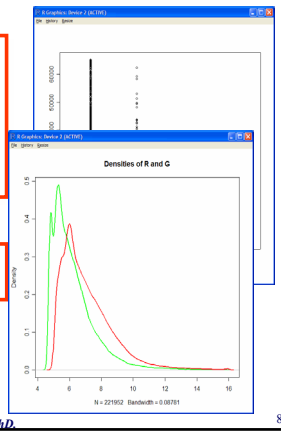
Importing Data

```
> setwd("C://temp//Rdata")
> DMTKRcsv<-read.csv("DMTKRcsv.csv",
      header = TRUE, sep = ",", dec=".")
> DMTKRcsv
> attach(DMTKRcsv)

> scan(file = "DMTKRcsv.csv", skip=1, sep = ",",
      dec = ".")
```


Boxplots and Density Plots

```
> df <- read.table("foo.txt", header=TRUE)
> dim(df)
[1] 221952      6
> boxplot(df[,c("R","G","Rb","Gb")],
col=c("red","green"))
# Ooops
> boxplot(df[,c("R","G","Rb","Gb")],
col=c("red","green"), ylim=c(0,300))
# Better!
```

```
> plot(density(log(df$G,base=2)),
col="green", main="Densities of R and G")
> lines(density(log(df$R,base=2)),
col="red")
```



Summary of "data types"

Vectors (only one type at the time):

```
> x <- 7:1
> x
[1] 7 6 5 4 3 2 1
> x <- c("a",3,"e")
> x
[1] "a" "3" "e"
```

Matrices (rectangular, only one type at the time):

```
> x <- matrix(1:18, nrow=3)
> x
     [,1] [,2] [,3] [,4] [,5]
[,6]
[1,]    1    4    7   10   13
16
[2,]    2    5    8   11   14
17
[3,]    3    6    9   12   15
18
> x <- matrix(letters[1:12],
nrow=3)
> x
     [,1] [,2] [,3] [,4]
[1,] "a"  "d"  "g"  "j"
[2,] "b"  "e"  "h"  "k"
[3,] "c"  "f"  "i"  "l"
```

Lists (any shape, mixed type):

```
> x <- list(a=1:4, b=4:6+2i,
src="GenePix")
> x
$a
[1] 1 2 3 4
$b
[1] 4+2i 5+2i 6+2i
$src
[1] "GenePix"
```

Data frames (rectangular, mixed type):

```
> x <-
data.frame(name=c("jon","kim","dan"),
age=c(87,78,45),
weight=c(76.3,96.3,62.9),
height=c(1.67,1.84,1.54))
> x
  name age weight height
1  jon  87   76.3   1.67
2  kim  78   96.3   1.84
3  dan  45   62.9   1.54
```

Bar Plot, Pie Plot, Table

```
Med.table<-table(Med)
pie(Med.table)
barplot(Med.table)

table(sex, Med)
```


Stem-and-Leaf Plot

```
stem(PREKS)
```


Box Plots and Histograms

```
boxplot(PREKS)
boxplot(PREKS,POSKS)
hist(PREKS)
hist(PREKS,freq=FALSE)
hist(PREKS, breaks=seq(33,63,3), freq=FALSE)
```


Relative Frequency Polygon

```
preks.hist<- hist(PREKS, breaks=seq(30,70,1),
    freq=FALSE, border="white")
lines(preks.hist$mid,preks.hist$intensities)
abline(h=0)
```


Relative Cumulative Frequency Plots

```
preks.hist<- hist(PREKS, breaks=seq(30,70,1),
    freq=FALSE, border="white")
preks.int<-seq(31,70,1)
preks.rcf<-cumsum(preks.hist$intensities)
plot(preks.int, preks.rcf, type="l") # "l"ittle
```


Summary

```
summary(PREKS)
mean(PREKS)
median(PREKS)
var(PREKS)
sd(PREKS)
```

Quantiles

```
quantile(PREKS)
quantile(PREKS, seq(0,1,0.25))
quantile(PREKS, seq(0,1,0.20))
quantile(PREKS, seq(0,1,0.1))
quantile(PREKS, seq(0,1,0.05))

qqnorm(PREKS)
qqline(PREKS, col = 2)
qqplot(PREKS,rnorm(300))
```


Bivariates

```
plot(age, PREKS)
plot(PREKS, POSKS)

boxplot(PREKS~Med)
boxplot(POSKS~ABS)

table(sex, Med)
table(sex,Med,ABS)
```


Questions?!

Exploratory Data Analysis (EDA)

- Also called *descriptive statistics*, this term is used to describe the process of 'looking at the data' prior to formal analysis
- In this phase of analysis, data are examined for quality and 'cleaned' as well as displayed to provide an overall impression of results
- We will look at two types of summaries:
  - Graphical summaries
  - Numerical summaries

Graphical Data Summaries

- For a single categorical variable:
  - Bar plot, dot plot (not covered here)
- For a single numerical variable:
  - Histogram (next)
  - Boxplot (a little later)
- For two numerical variables:
  - Scatterplot

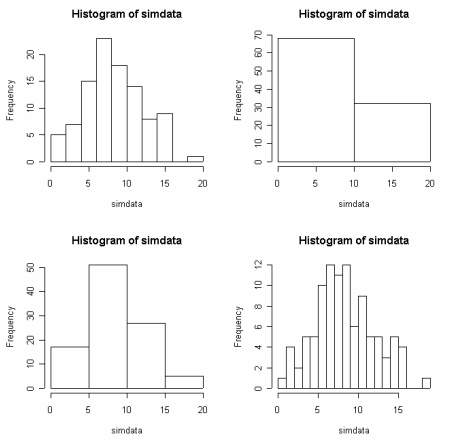
Histogram

- A *histogram* is a special kind of bar plot
- It allows you to visualize the *distribution* of values for a numerical variable
- When drawn with a *density scale:*
  - the *AREA* (NOT height) of each bar is the proportion of observations in the interval
  - the *TOTAL AREA* is 100% (or 1)

R: Making a Histogram

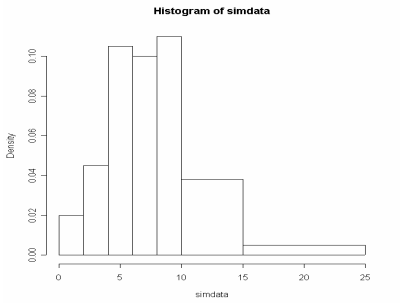
- Type `?hist` to view the help file
  - Note some important arguments, esp **breaks**
- Simulate some data, make histograms varying the number of bars (also called 'bins' or 'cells'), e.g.

```
> par(mfrow=c(2,2))    # set up multiple plots
> simdata <-rchisq(100,8)
> hist(simdata)  # default number of bins
> hist(simdata,breaks=2)  # etc,4,20
```

R: Setting Your Own Breakpoints

```
> bps <- c(0,2,4,6,8,10,15,25)
> hist(simdata,breaks=bps)
```



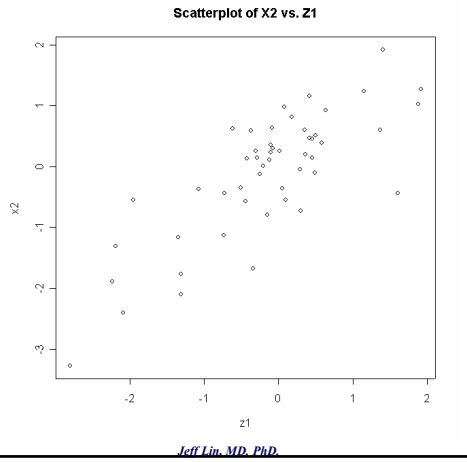
Scatterplot

- A scatterplot is a standard two-dimensional (X,Y) plot
- Used to examine the relationship between two (continuous) variables
- It is often useful to plot values for a single variable against the order or time the values were obtained

R: Making a Scatterplot

- Type `?plot` to view the help file
  - For now we will focus on simple plots, but R allows extensive user control for highly customized plots
- Simulate a bivariate data set:

```
> z1 <- rnorm(50)
> z2 <- rnorm(50)
> rho <- .75          # (or any number between -1 and 1)
> x2<- rho*z1+sqrt(1-rho^2)*z2
> plot(z1,x2)
```

Numerical Summaries

- Categorical/Qualitative variables
  - frequency table (not covered here)
- Numerical/Quantitative variables
  - *center*
  - *spread*

Measures of Center: Mean

- The *mean* value of a variable is obtained by computing the total of the values divided by the number of values
- Appropriate for distributions that are fairly symmetrical
- It is sensitive to presence of outliers, since all values contribute equally
- In R: `> mean(z1)`

Measures of Center: Median

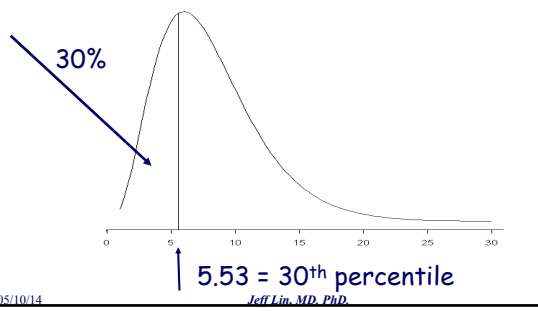
- The *median* value of a variable is the number having 50% (half) of the values smaller than it (and the other half bigger)
- It is NOT sensitive to presence of outliers, since it 'ignores' almost all of the data values
- The median is thus usually a more appropriate summary for skewed distributions
- In R: `> median(z1)`

Measures of Spread: SD

- The *standard deviation (SD)* of a variable is the square root of the average* of squared deviations from the mean (*for uninteresting technical reasons, instead of dividing by the number of values n, you usually divide by n-1)
- The *SD* is an appropriate measure of spread when center is measured with the *mean*
- In R: `> sd(z1)`

Slight Digression: Quantiles

- The p^{th} *quantile* is the number that has the proportion p of the data values smaller than it



Measures of Spread: IQR

- The 25th (Q_1), 50th (median), and 75th (Q_3) percentiles divide the data into 4 equal parts; these special percentiles are called *quartiles*
- The *interquartile range (IQR)* of a variable is the distance between Q_1 and Q_3:

$$IQR = Q_3 - Q_1$$

- The *IQR* is one way to measure spread when center is measured with the *median*
- In R: `> IQR(z1)  # note CAPITALS here`

Five-Number Summary and Boxplot

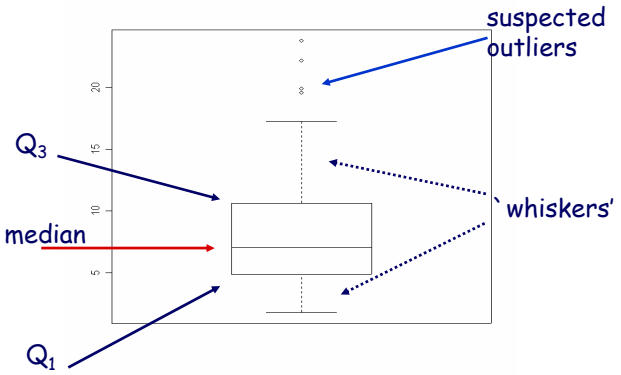
- An overall summary of the distribution of variable values is given by the five values:

  Min, Q_1, Median, Q_3, and Max

- In R, this summary can be obtained with the function **`quantile()`** (or the function **`summary()`**, which also includes the mean)
- A *boxplot* provides a visual summary of this five-number summary

Boxplot of Simdata

```
simdata <-rchisq(100,8)
```



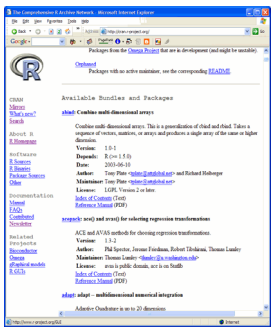
Measures of Spread: MAD

- The *median absolute deviation (MAD)* of a variable is obtained by
  1) getting the absolute values of the deviations between data values and the median, and then
  2) taking the median of those absolute deviations.
- MAD is a more robust measure of spread than the SD
- The *MAD* is another way (besides IQR) to measure spread when center is measured with the *median*
- In R: **`> mad(z1)`**

Introduction to Packages and Libraries

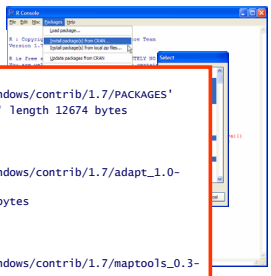
Packages

- **On CRAN - Comprehensive R Archive Network – there are today 300+ packages published!**
- **Browse CRAN at** ***http://www.r-project.org/***
- **Find what you want.**
- **Dirt simple to install package!**
- **At the Centre for Mathematical Sciences we try to keep install and update all package on our system.**



Install a Package

- On Windows extremely easy!
- On all systems:



```
> install.packages(c("adapt","maptools"))
trying URL `http://cran.r-project.org/bin/windows/contrib/1.7/PACKAGES'
Content type `text/plain; charset=iso-8859-1' length 12674 bytes
opened URL
downloaded 12Kb

trying URL `http://cran.r-project.org/bin/windows/contrib/1.7/adapt_1.0-
    3.zip'
Content type `application/zip' length 39304 bytes
opened URL
downloaded 38Kb

trying URL `http://cran.r-project.org/bin/windows/contrib/1.7/maptools_0.3-
    2.zip
'
Content type `application/zip' length 129634 bytes
opened URL
downloaded 126Kb

Delete downloaded files (y/N)? y
updating HTML package descriptions
> library(maptools)
```

Update Packages

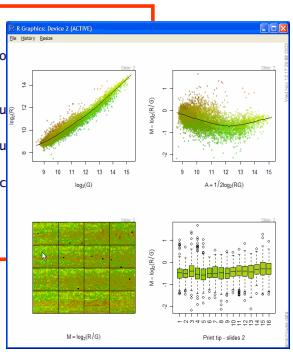
- On all systems:

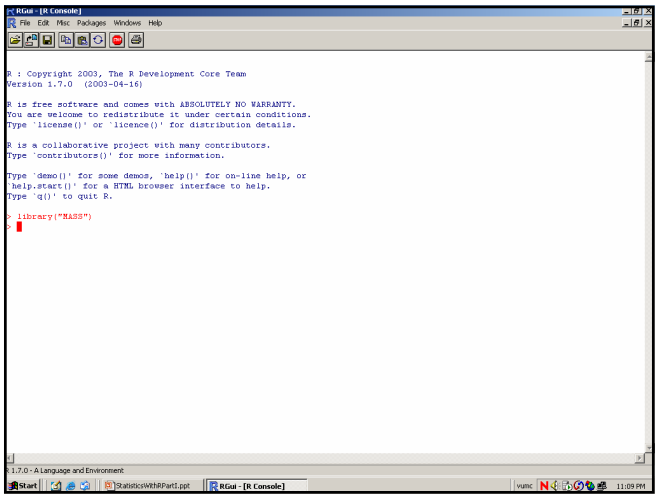
```
> update.packages()
trying URL `http://cran.r-project.org/bin/windows/contrib/1.7/PACKAGES'
Content type `text/plain; charset=iso-8859-1' length 12674 bytes
opened URL
downloaded 12Kb
cluster :
 Version 1.7.3 in c:/PROGRA~1/R/rw1071/library
 Version 1.7.6 on CRAN
Update (y/N)?  y
foreign :
 Version 0.6-1 in c:/PROGRA~1/R/rw1071/library
 Version 0.6-3 on CRAN
Update (y/N)?  y
...
trying URL `http://cran.r-project.org/bin/windows/contrib/1.7/foreign_0.6-
    3.zip'
Content type `application/zip' length 109855 bytes
opened URL
downloaded 107Kb

Delete downloaded files (y/N)? y
updating HTML package descriptions
>
```

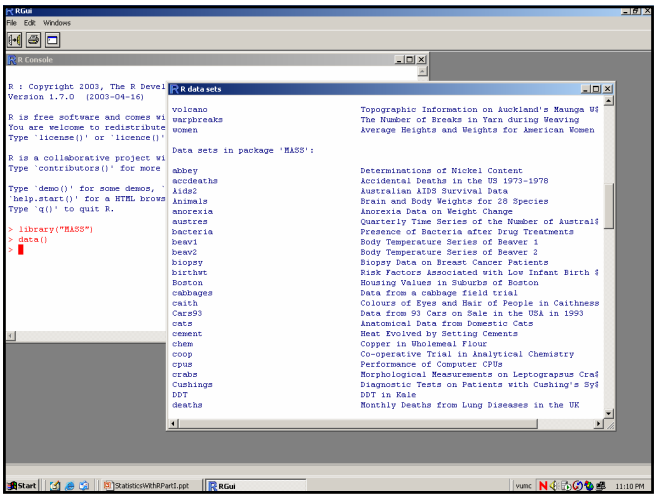

Using/Loading a Package

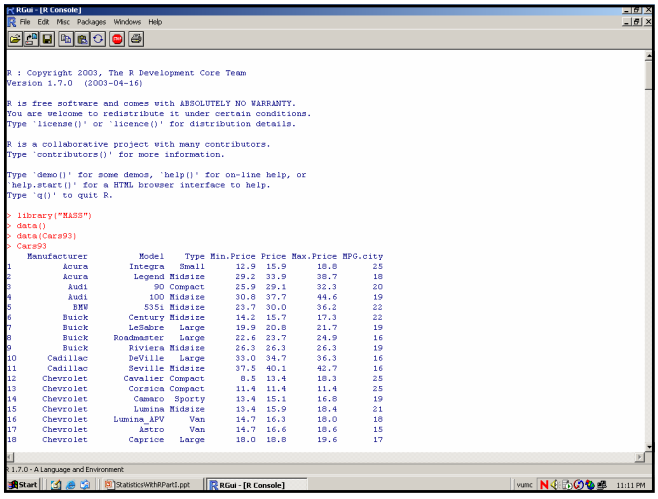
- On all systems:

```
> library(maptools)

# Some package loads other package to
> library(com.braju.sma)
Loading required package: R.oo
R.oo v0.44 (2003/10/29) was successfu
Loading required package: R.io
R.io v0.44 (2003/10/29) was successfu
Loading required package: R.graphics
R.graphics v0.44 (2003/10/29) was suc
com.braju.sma v0.64 (2003/10/31) was
    loaded.

> example(MAData)
```









Questions?!

Introduction to R Programming

Probability Distributions

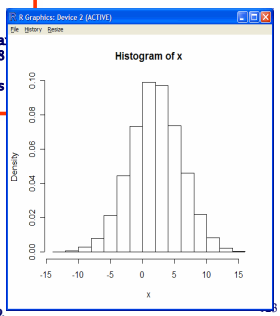
- Cumulative distribution function $P(X \leq x)$: 'p' for the CDF
- Probability density function: 'd' for the density,,
- Quantile function (given q, the smallest x such that $P(X \leq x) > q$): 'q' for the quantile
- simulate from the distribution: 'r

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df1, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog

logistic logis; negative binomial nbinom; normal norm; Poisson pois; Student's t t ; uniform unif; Weibull weibull; Wilcoxon wilcox

Random Numbers

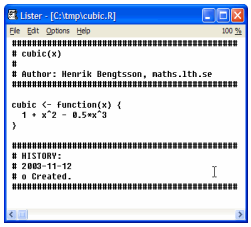
```
> x <- rnorm(10000, mean=2, sd=4)
> length(x)
[1] 10000
> mean(x)
[1] 2.007904
> sd(x)
[1] 3.969784
> summary(x)
    Min. 1st Qu.  Median  Mean  3rd Qu.    Ma
-12.670   -0.607   2.000  2.008   4.701  15.8
> hist(x)
# or use probabilities (not counts) on y-axis
> hist(x, probability=TRUE)
```



Defining Functions and Scripts

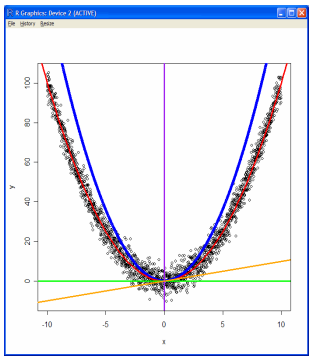
```
> cubic <- function(x) 1 + x^2 -
0.5*x^3
> cubic(0:4)
[1]   1.0   1.5   1.0  -3.5 -15.0
```

```
# Read R code from file
> source("cubic.R")
> cubic(0:4)
[1]   1.0   1.5   1.0  -3.5 -15.0
> cubic
function(x) {
  1 + x^2 - 0.5*x^3
}
```



Adding Data Points to an Existing Plot

```
> f <- function(x) x^2

> x <- seq(-10,10, by=0.01)
> eps <- rnorm(length(x), sd=4)
> y <- f(x) + eps

# plot() creates a new plot
> plot(x,y)

# points() add data points to
# an existing plot
> points(x,f(1.1*x), col="blue")

# Same for abline() and others
> abline(h=0, col="green")
> abline(v=0, col="purple")
> abline(a=0, b=1, col="orange")

# Same for curve() with add=TRUE
> curve(f, col="red", add=TRUE)

(On request by Linda)
```



Questions?!

Thanks !