Introduction To R

Dillip Kumar Majhi

Hadoop Architect

What is R?

- The R statistical programming language is a free open source package based on the S language developed by Bell Labs.
- The language is very powerful for writing programs.
- Many statistical functions are already built in.
- Contributed packages expand the functionality to cutting edge research.
- Since it is a programming language, generating computer code to complete tasks is required.

History of R

- S: language for data analysis developed at Bell Labs circa 1976
- Licensed by *AT&T/Lucent* to *Insightful Corp*. Product name: *S-plus*.
- R: initially written & released as an open source software by Ross Ihaka and Robert Gentleman at U Auckland during 90s (R plays on name “S”)
- Since 1997: international R-core team ~15 people & 1000s of code writers and statisticians happy to share their libraries! AWESOME!

"Open source"... that just means I don't have to pay for it, right?

- No. Much more:
  - Provides full access to algorithms and their implementation
  - Gives you the ability to fix bugs and extend software
  - Provides a forum allowing researchers to explore and expand the methods used to analyze data
  - Is the product of 1000s of leading experts in the fields they know best. It is CUTTING EDGE.
  - Ensures that scientists around the world - and not just ones in rich countries - are the co-owners to the software tools needed to carry out research
  - Promotes reproducible research by providing open and accessible tools
  - Most of R is written in… R! This makes it quite easy to see

What is it?

- R is an interpreted computer language.
  - Most user-visible functions are written in R itself, calling upon a smaller set of internal primitives.
  - It is possible to interface procedures written in C, C+, or FORTRAN languages for efficiency, and to write additional primitives.
  - System commands can be called from within R

- R is used for data manipulation, statistics, and graphics. It is made up of:
  - operators (+ - <- * %*% …) for calculations on arrays & matrices
  - large, coherent, integrated collection of functions
  - facilities for making unlimited types of publication quality graphics
  - user written functions & sets of functions (packages); 800+ contributed packages so far & growing

R

Advantages

- Fast and free.
- State of the art: Statistical researchers provide their methods as R packages. SPSS and SAS are years behind R!
- 2nd only to MATLAB for graphics.
- Mx, WinBugs, and other programs use or will use R.
- Active user community
- Excellent for simulation, programming, computer intensive analyses, etc.
- Forces you to *think* about your analysis.
- Interfaces with database storage software (SQL)

Disadvantages

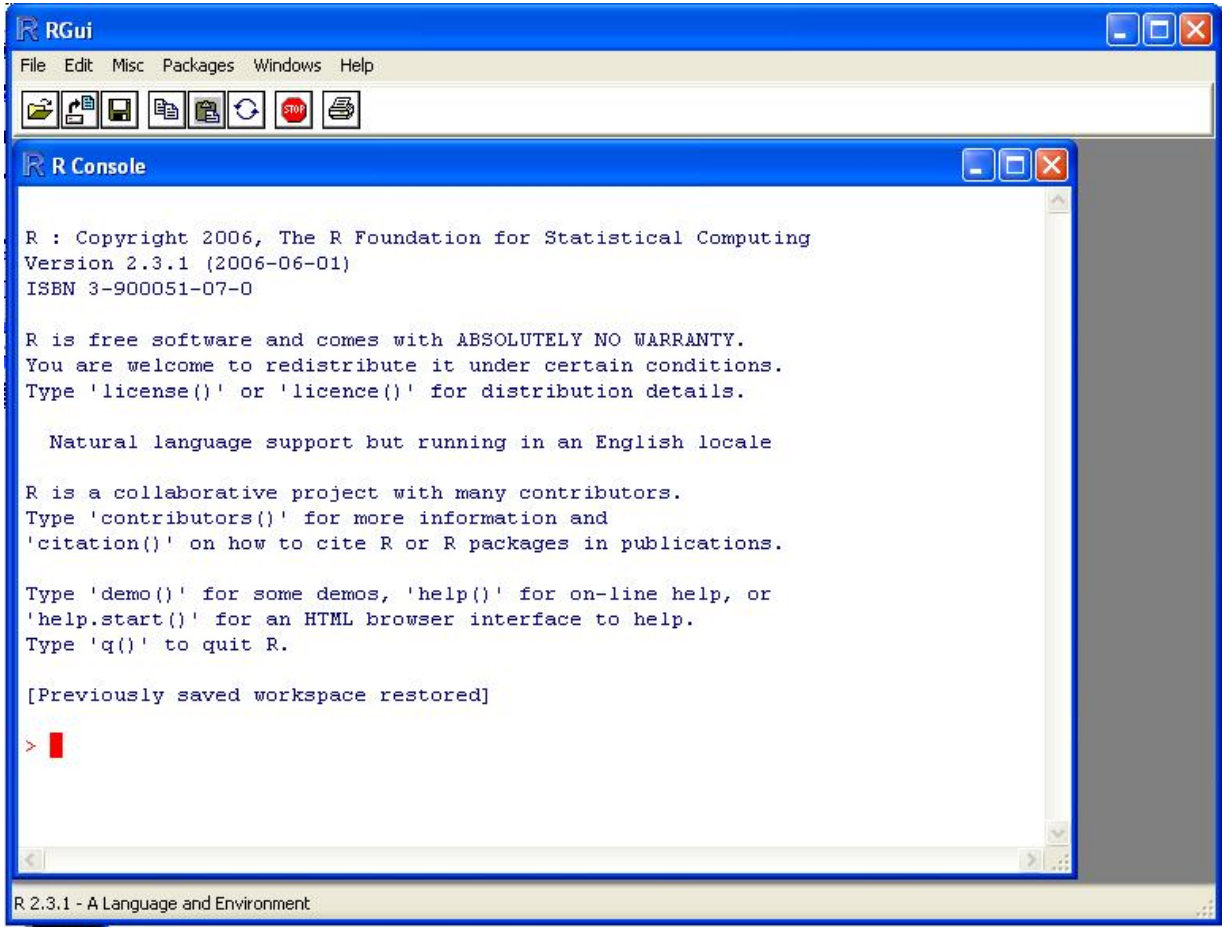
- Not user friendly @ start - steep learning curve, minimal GUI.
- No commercial support; figuring out correct methods or how to use a function on your own can be frustrating.
- Easy to make mistakes and not know.
- Working with large datasets is limited by RAM
- Data prep & cleaning can be messier & more mistake prone in R vs. SPSS or SAS
- Some users complain about hostility on the R listserve

Getting Started

- Where to get R?
- Go to www.r-project.org
- Downloads: CRAN
- Set your Mirror: Anyone in the USA is fine.
- Select Windows 95 or later.
- Select base.
- Select R-2.4.1-win32.exe
  - The others are if you are a developer and wish to change the source code.
- UNT course website for R:
  - http://www.unt.edu/rss/SPLUSclasslinks.html

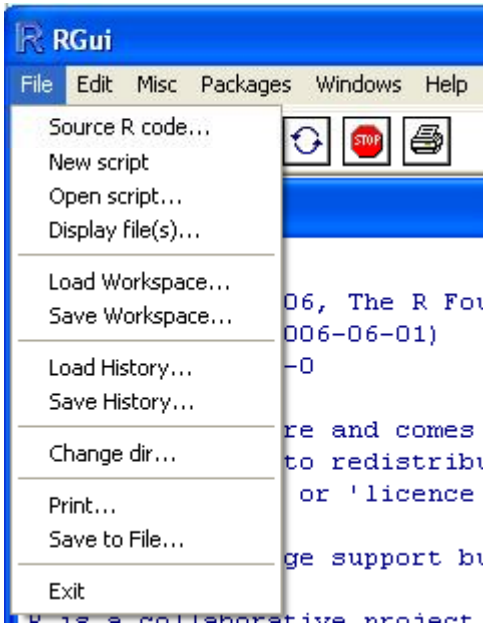
Getting Started

- The R GUI?



Getting Started

- Opening a script.
- This gives you a script window.



Getting Started

- Basic assignment and operations.
- Arithmetic Operations:
  - +, -, *, /, ^ are the standard arithmetic operators.
- Matrix Arithmetic.
  - * is element wise multiplication
  - %*% is matrix multiplication
- Assignment
  - To assign a value to a variable use “<-”

Getting Started

- How to use help in R?
  - R has a very good help system built in.
  - If you know which function you want help with simply use ?_____ with the function in the blank.
  - Ex: ?hist.
  - If you don't know which function to use, then use help.search("_____").
  - Ex: help.search("histogram").

Operators

- Mathematic operators: + - * / ^
  - Mod: %%
  - sqrt, exp, log, log10, sin, cos, tan, …..
- Other operators:
  - $ component selection HIGH
  - [, [[subscripts, elements
  - : sequence operator
  - %*% matrix algebra
  - <, >, <=, >= inequality
  - ==, != comparison
  - ! not
  - &, |, &&, || and, or
  - ~ formulas
  - <- assignment (or = 1.9.1 later)

Demo Algebra, Operators and Functions

```
> 1+2
[1] 3
> 1 > 2
[1] FALSE
> 1 > 2 | 2 > 1
[1] TRUE
> 1:3
[1] 1 2 3
> A = 1:3
> A
[1] 1 2 3
> A*6
[1]  6 12 18
> A/10
[1] 0.1 0.2 0.3
> A %% 2
[1] 1 0 1
```

```
> B=4:6
> A*B
[1]  4 10 18
> A%*%B
     [,1]
[1,]   32
> A %*% t(B)
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]    8   10   12
[3,]   12   15   18
> A / B
[1] 0.25 0.40 0.50
> sqrt(A)
[1] 1.000000 1.414214 1.732051
> log(A)
[1] 0.0000000 0.6931472 1.0986123
```

```
> round(sqrt(A),2)
[1] 1.00 1.41 1.73
> ceiling(sqrt(A))
[1] 1 2 2
> floor(sqrt(A))
[1] 1 1 1
> eigen( A%*% t(B))
$values
[1] 3.200000e+01 5.835176e-16 2.480655e-16

$vectors
          [,1]       [,2]       [,3]
[1,] 0.2672612  0.3273463 -0.8890009
[2,] 0.5345225 -0.8217055  0.2540003
[3,] 0.8017837  0.4665237  0.3810004
> eigen( A%*% t(B))$values
[1] 3.200000e+01 5.835176e-16 2.480655e-16
```

Importing Data

- How do we get data into R?
- Remember we have no point and click…
- First make sure your data is in an easy to read format such as CSV (Comma Separated Values).
- Use code:
  - `D <- read.table("path",sep=",",header=TRUE)`

Working with data.

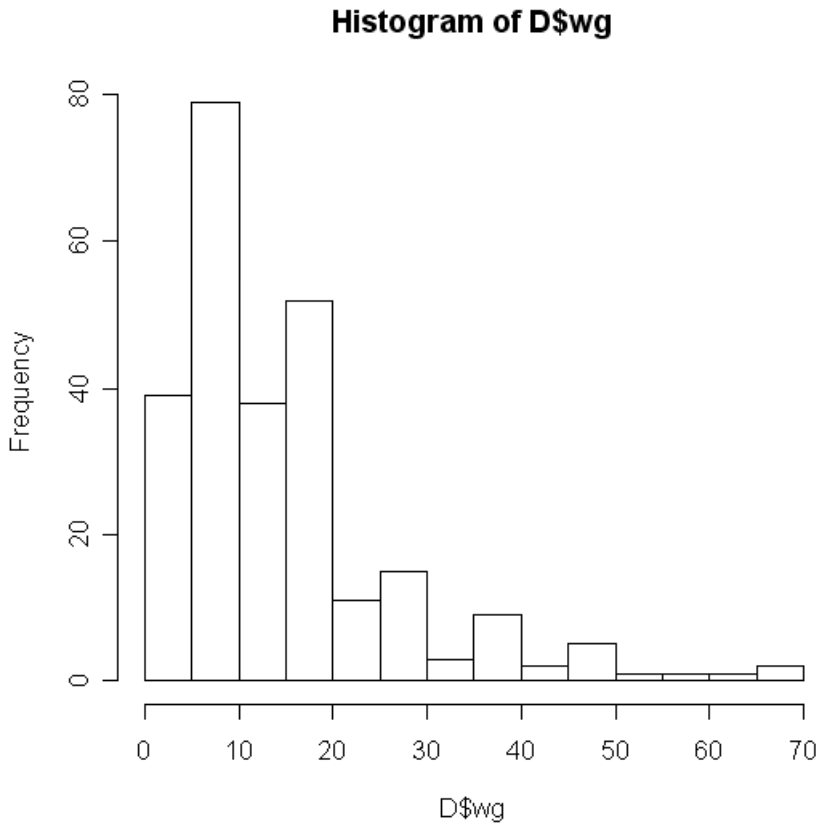
- Accessing columns.
- D has our data in it…. But you can't see it directly.
- To select a column use `D$column`.

Working with data.

- Subsetting data.
- Use a logical operator to do this.
  - ==, >, <, <=, >=, <> are all logical operators.
  - Note that the “equals” logical operator is two = signs.
- Example:
  - `D[D$Gender == “M”,]`
  - This will return the rows of D where Gender is “M”.
  - Remember R is case sensitive!
  - This code does nothing to the original dataset.
  - `D.M <- D[D$Gender == “M”,]` gives a dataset with the appropriate rows.

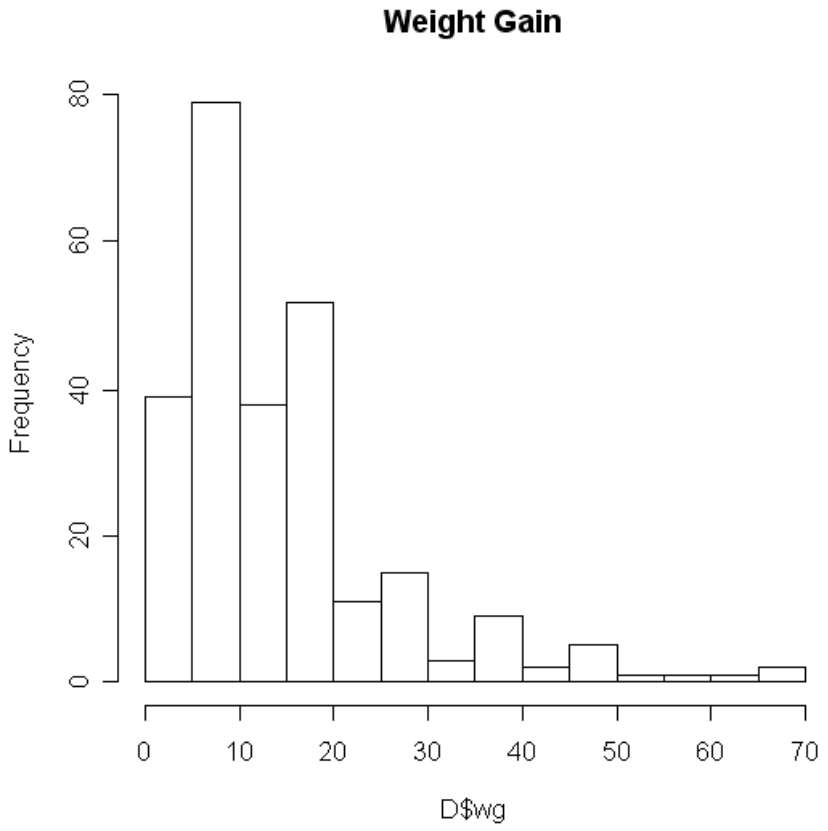
Basic Graphics

- Histogram
  - `hist(D$wg)`



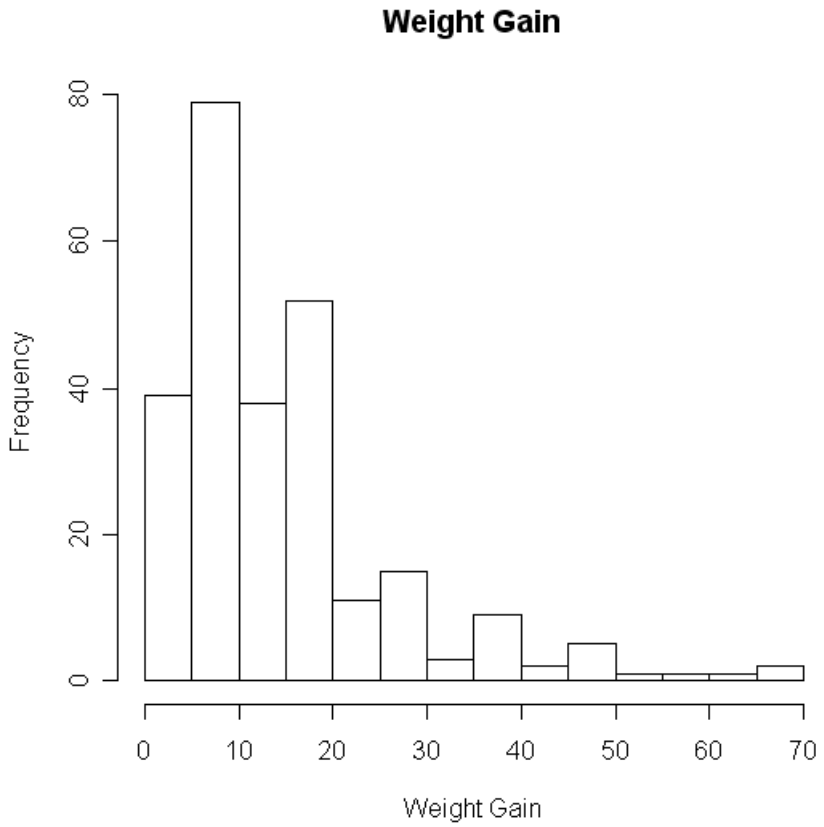
Basic Graphics

- Add a title…
  - The “main” statement will give the plot an overall heading.
  - `hist(D$wg , main='Weight Gain')`



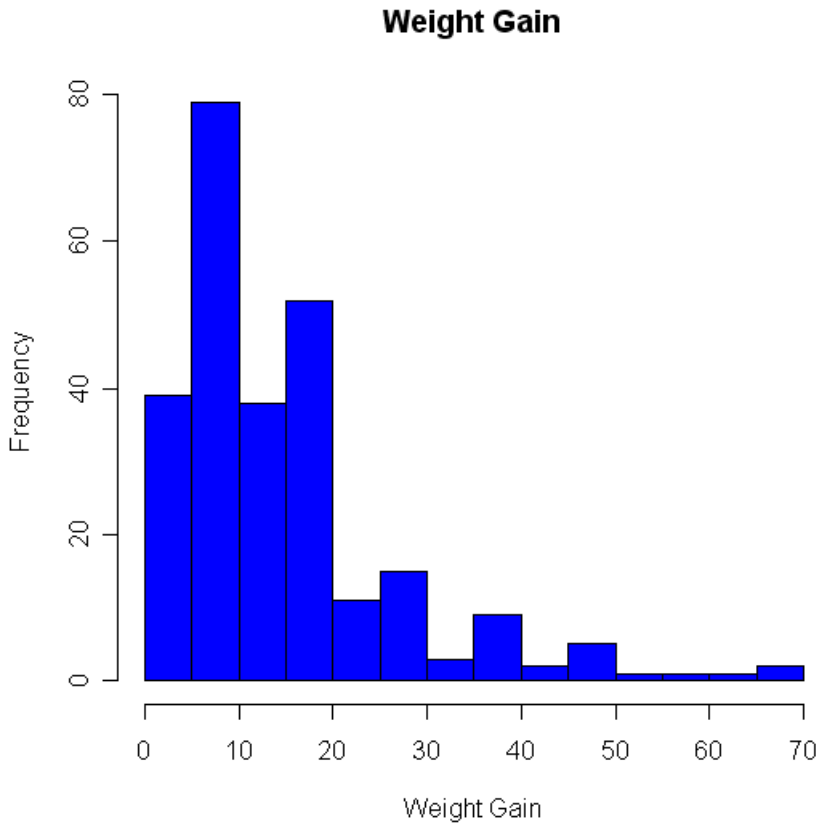
Basic Graphics

- Adding axis labels…
- Use “xlab” and “ylab” to label the X and Y axes, respectively.
- `hist(D$wg , main='Weight Gain',xlab='Weight Gain', ylab ='Frequency')`

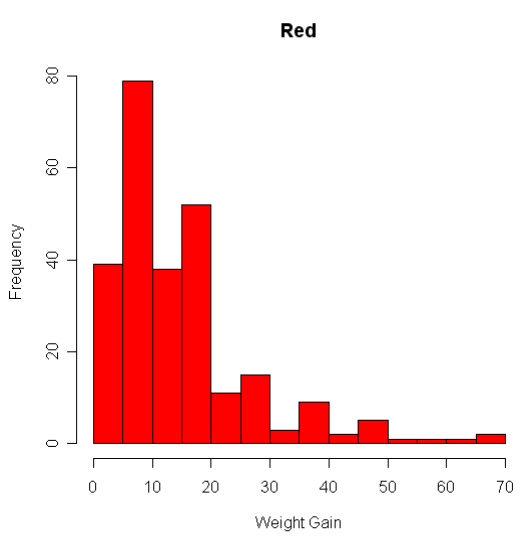


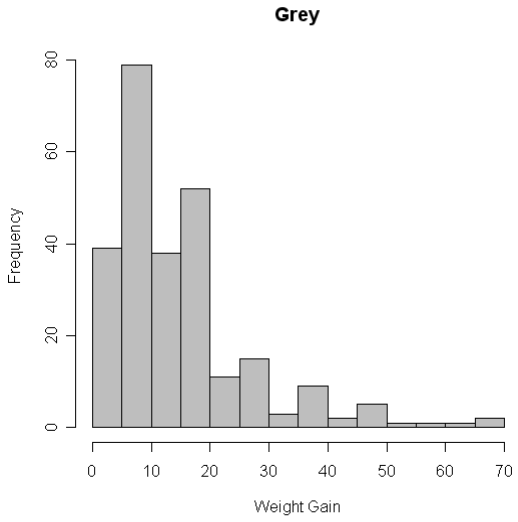
Basic Graphics

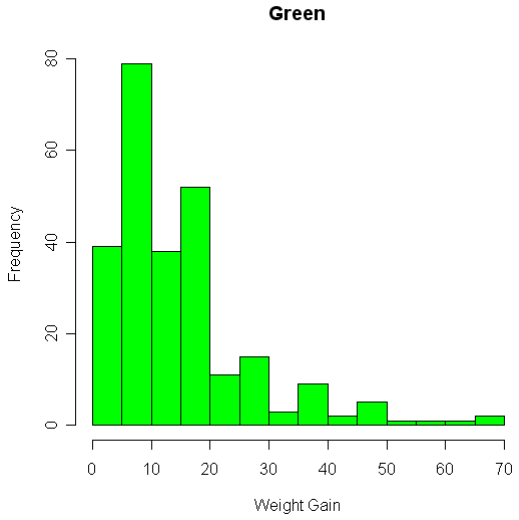
- Changing colors…
- Use the col statement.
  - ?colors will give you help on the colors.
  - Common colors may simply put in using the name.
  - `hist(D$wg, main="Weight Gain",xlab="Weight Gain", ylab ="Frequency", col="blue")`

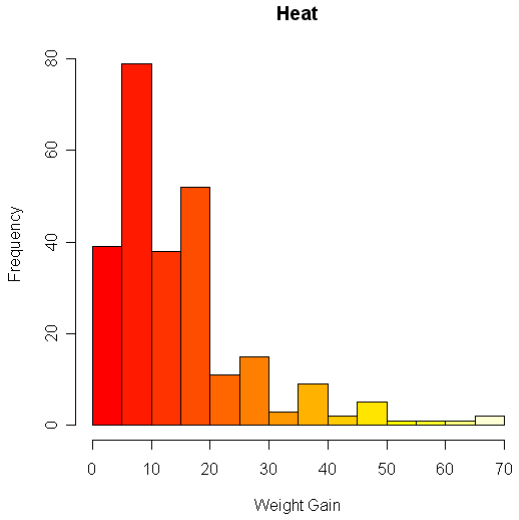


Basic Graphics – Colors



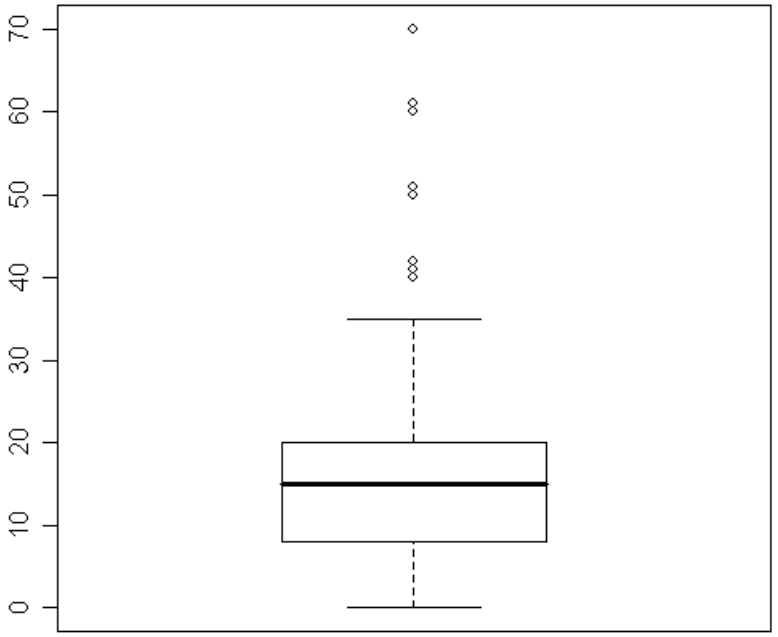






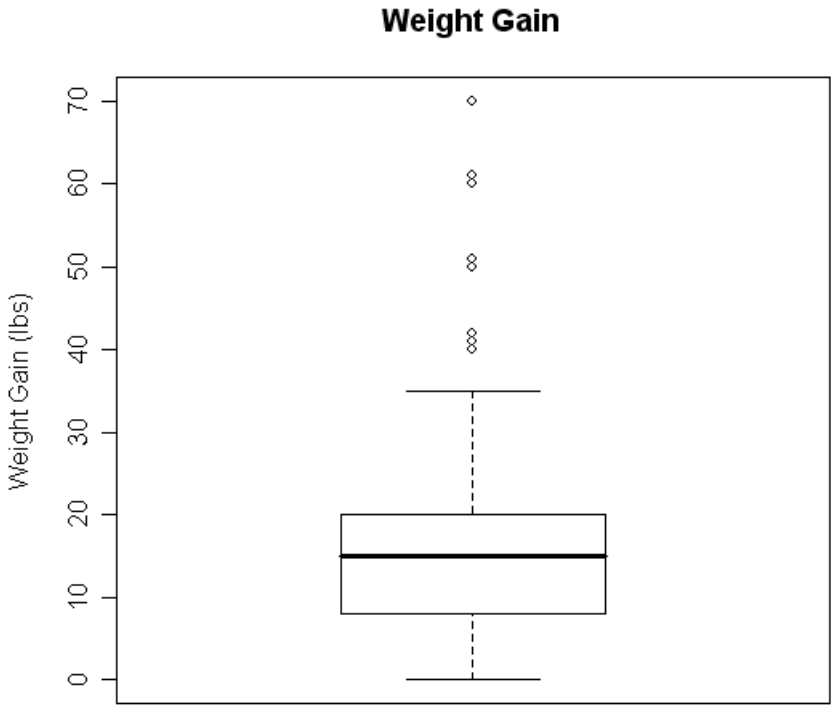
Basic Plots

- Box Plots
  - `boxplot(D$wg)`



Boxplots

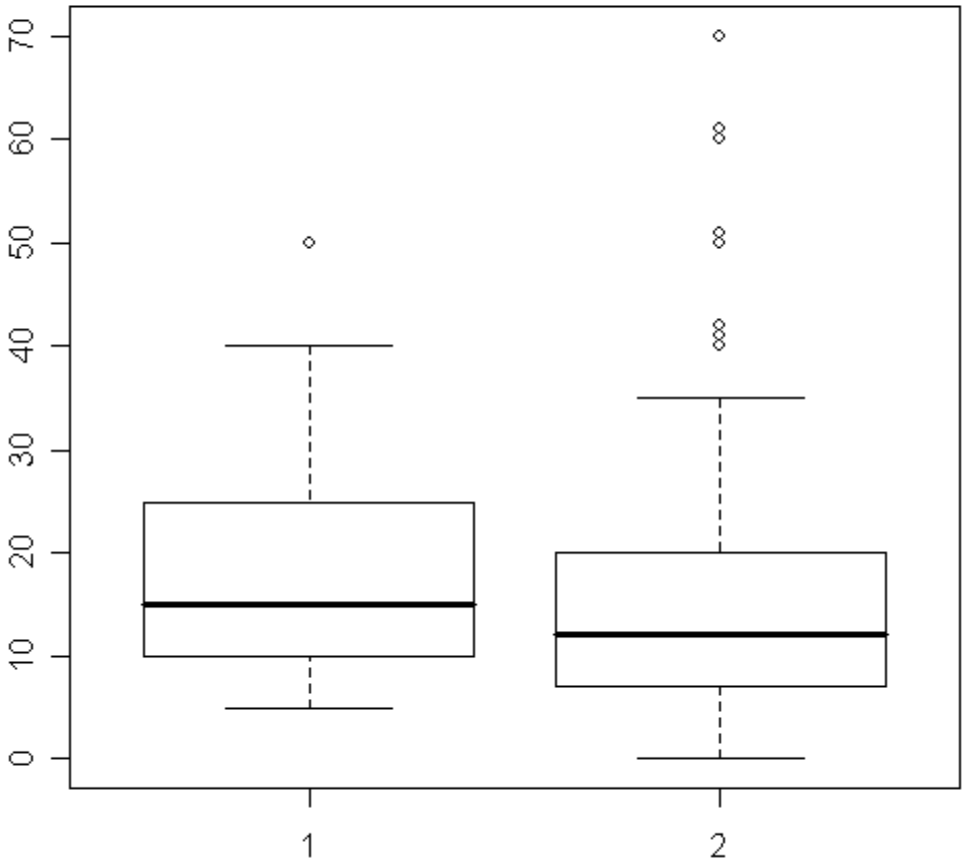
- Change it!
- `boxplot(D$wg,main='Weight Gain',ylab='Weight Gain (lbs)')`



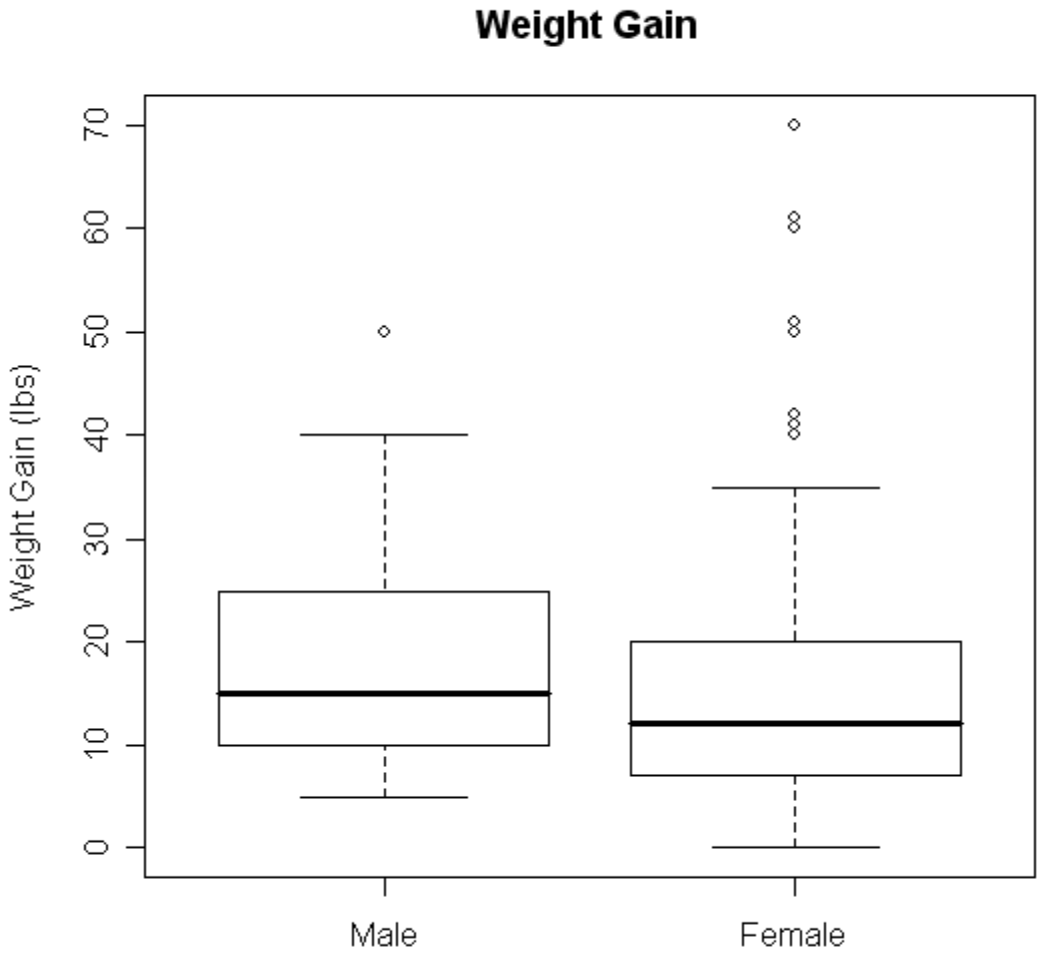
Box-Plots - Groupings

- What if we want several box plots side by side to be able to compare them.
- First Subset the Data into separate variables.
  - `wg.m <- D[D$Gender=="M",]`
  - `wg.f <- D[D$Gender=="F",]`
- Then Create the box plot.
  - `boxplot(wg.m$wg,wg.f$wg)`

Boxplots – Groupings



Boxplots - Groupings

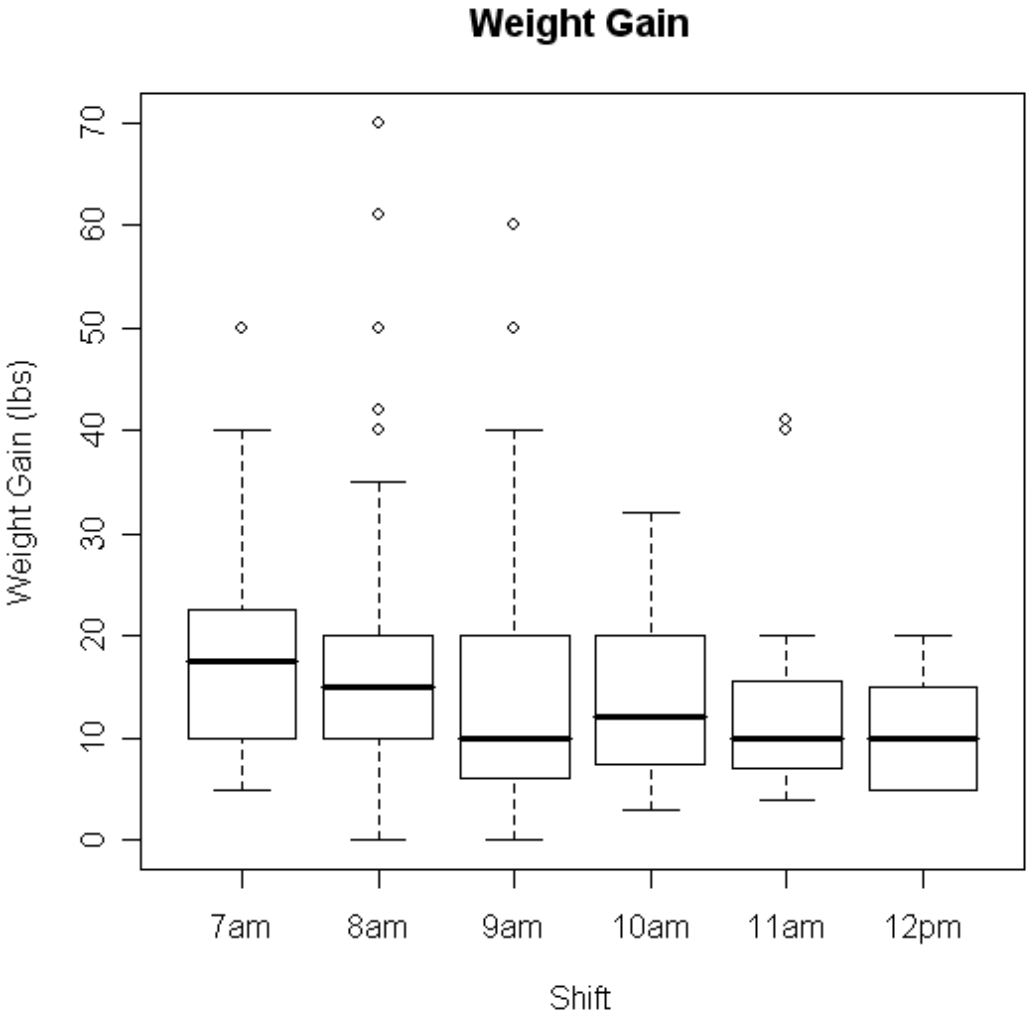



```
boxplot(wg.m$wg, wg.f$wg, main='Weight Gain (lbs)',
ylab='Weight Gain', names = c('Male','Female'))
```

Boxplot Groupings

- Do it by shift
  - `wg.7a <- D[D$Shift=="7am",]`
  - `wg.8a <- D[D$Shift=="8am",]`
  - `wg.9a <- D[D$Shift=="9am",]`
  - `wg.10a <- D[D$Shift=="10am",]`
  - `wg.11a <- D[D$Shift=="11am",]`
  - `wg.12p <- D[D$Shift=="12pm",]`
  - `boxplot(wg.7a$wg, wg.8a$wg, wg.9a$wg, wg.10a$wg, wg.11a$wg, wg.12p$wg, main='Weight Gain', ylab='Weight Gain (lbs)', xlab='Shift', names = c('7am','8am','9am','10am','11am','12pm'))`

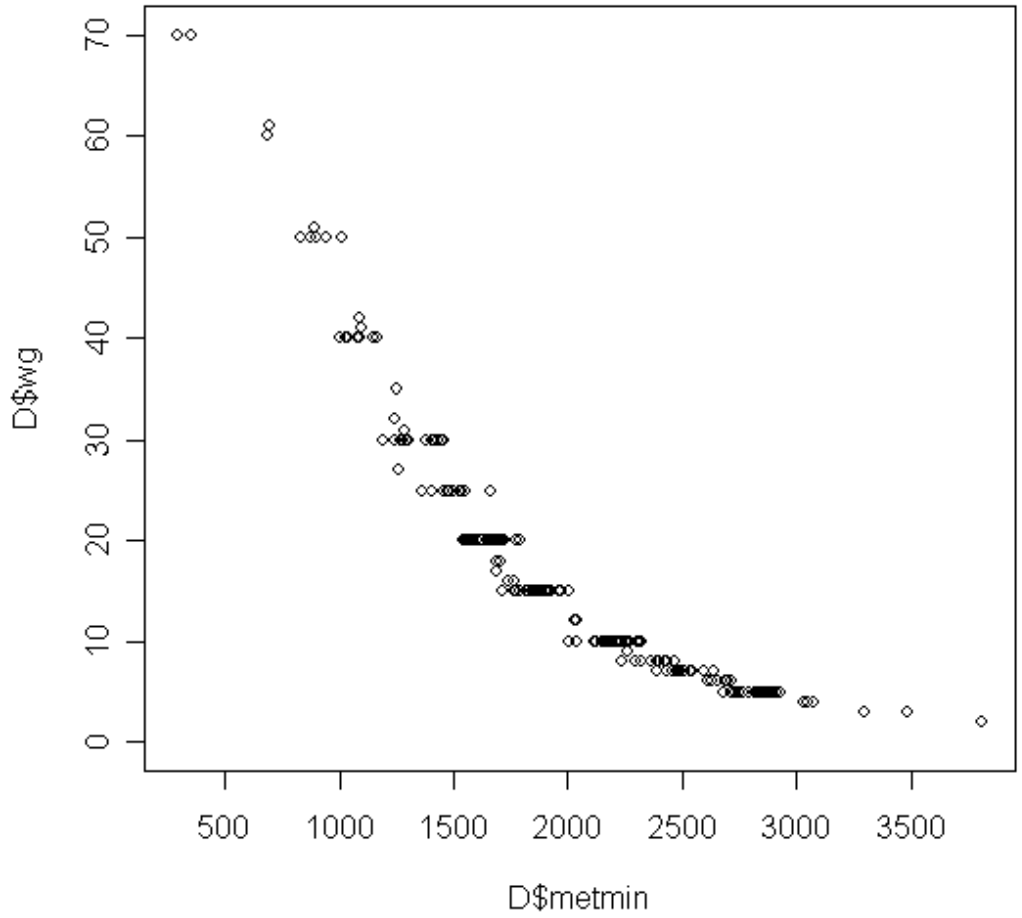
Boxplots Groupings



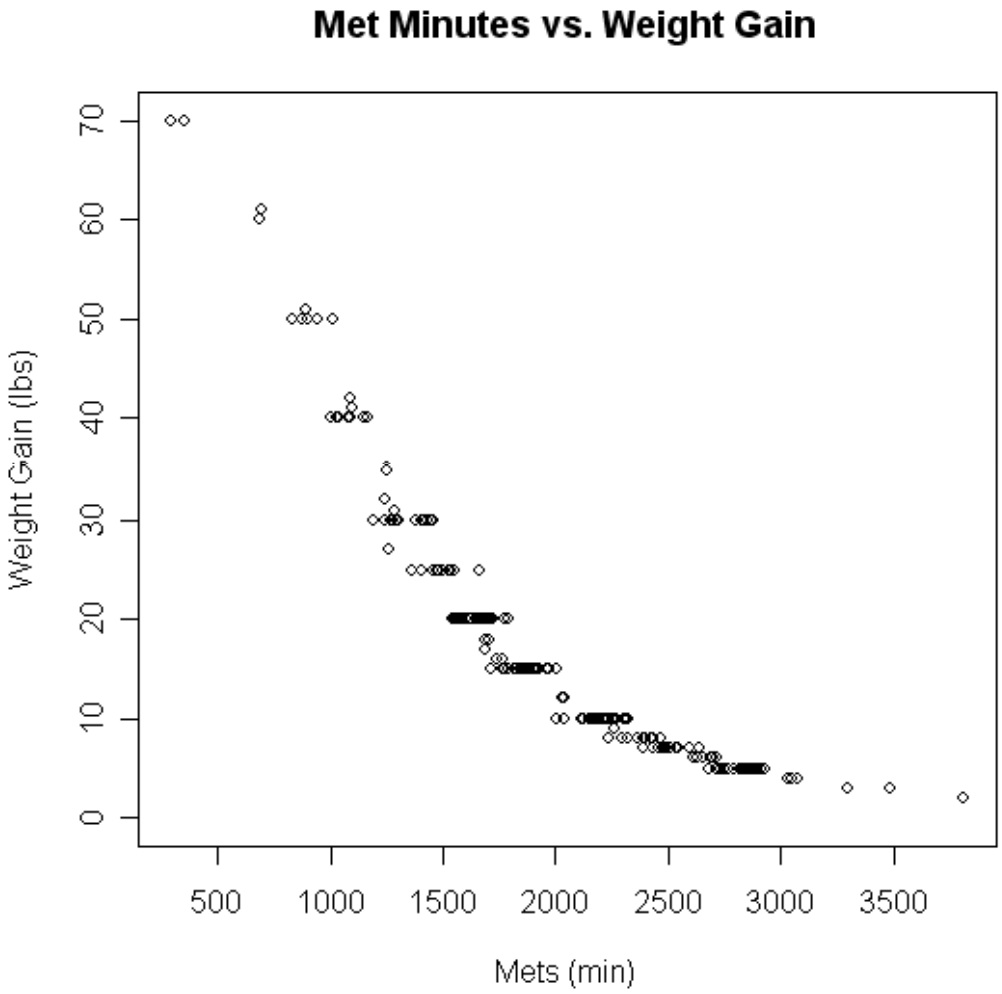
Scatter Plots

- Suppose we have two variables and we wish to see the relationship between them.
- A scatter plot works very well.
- R code:
  - `plot(x,y)`
- Example
  - `plot(D$metmin,D$wg)`

Scatterplots

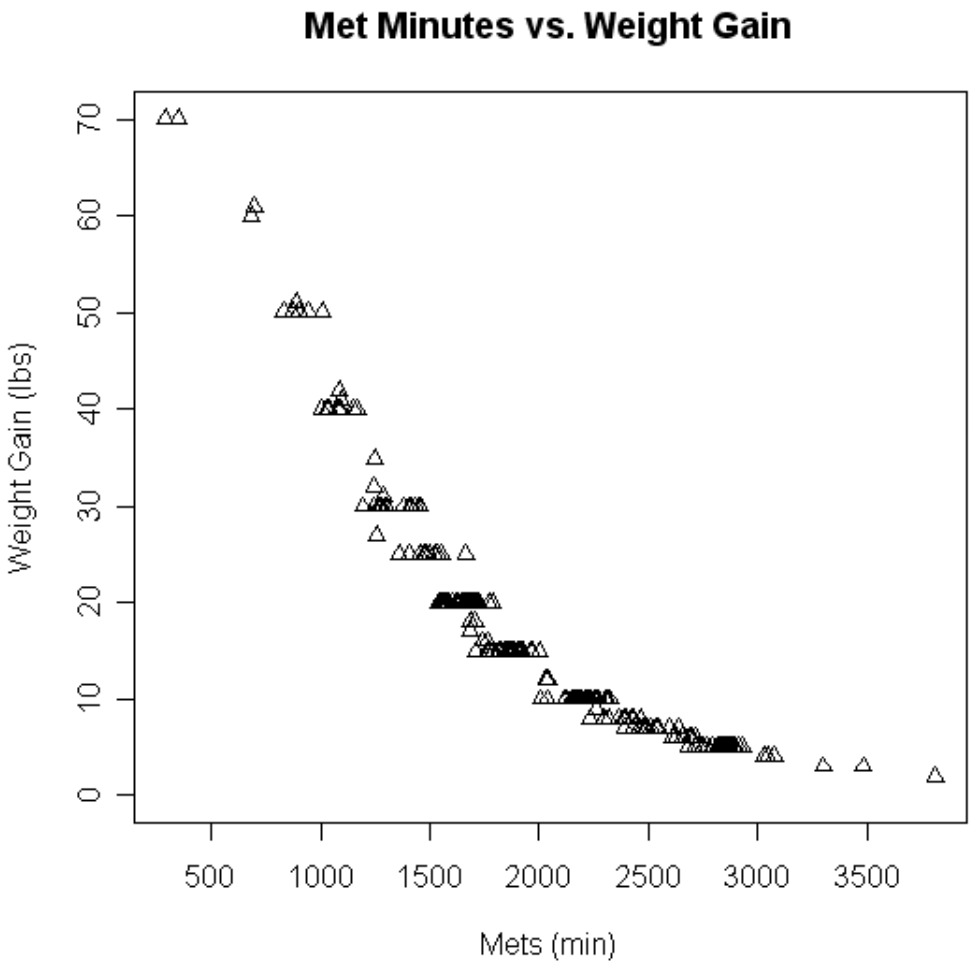


Scatterplots




```
plot(D$metmin,D$wg,main='Met Minutes vs. Weight Gain',
xlab='Mets (min)',ylab='Weight Gain (lbs)')
```

Scatterplots

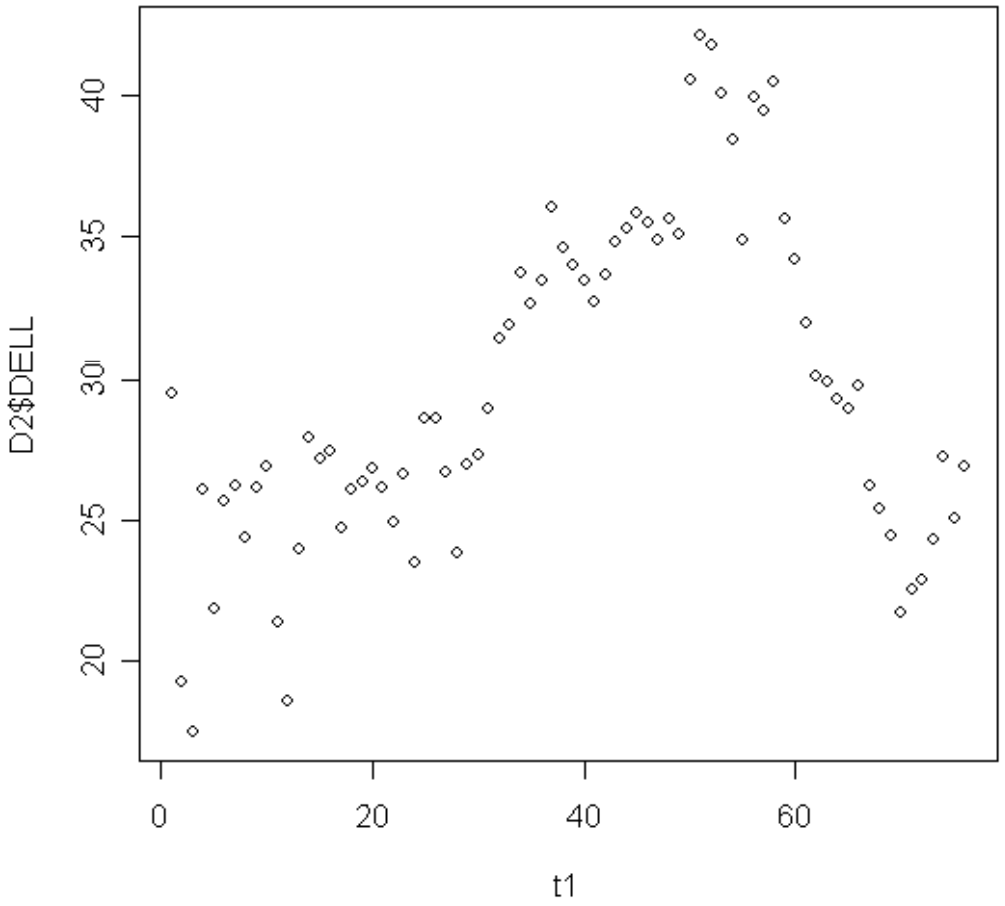



```
plot(D$metmin,D$wg,main='Met Minutes vs. Weight Gain',
 xlab='Mets (min)',ylab='Weight Gain (lbs)',pch=2)
```

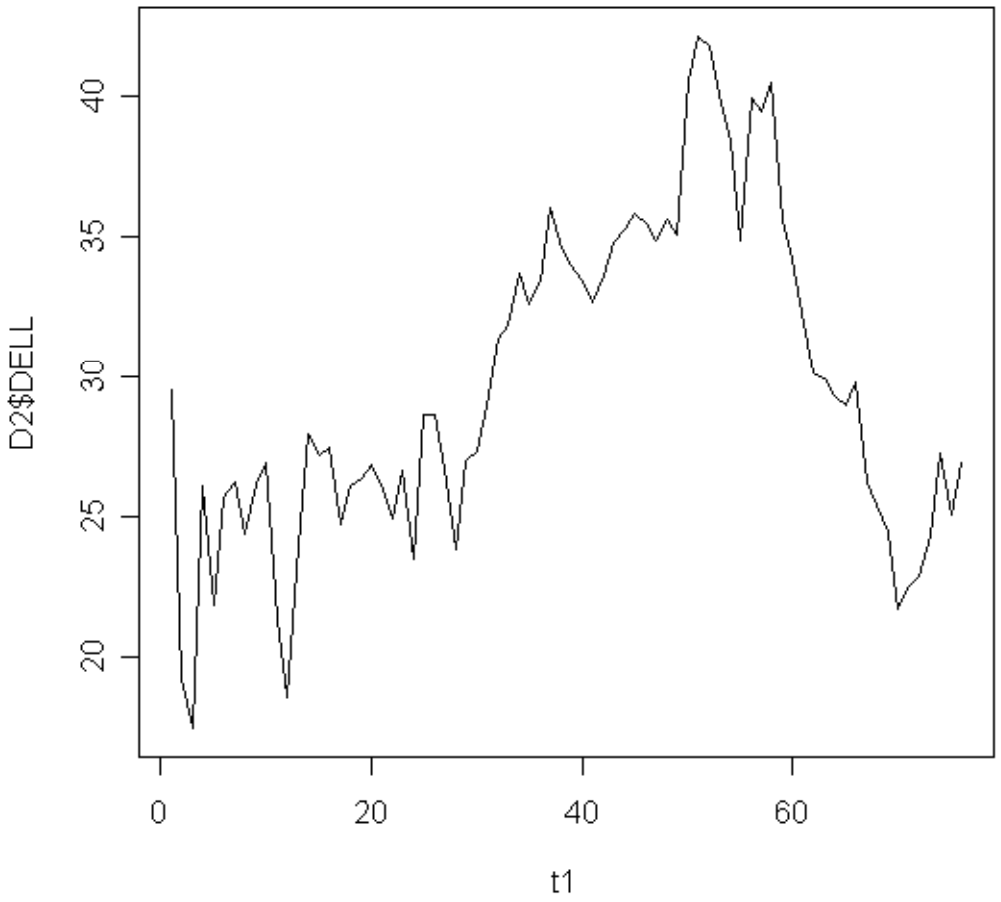
Line Plots

- Often data comes through time.
- Consider Dell stock
  - `D2 <- read.csv("H:\\Dell.csv",header=TRUE)`
  - `t1 <- 1:nrow(D2)`
  - `plot(t1,D2$DELL)`

Line Plots

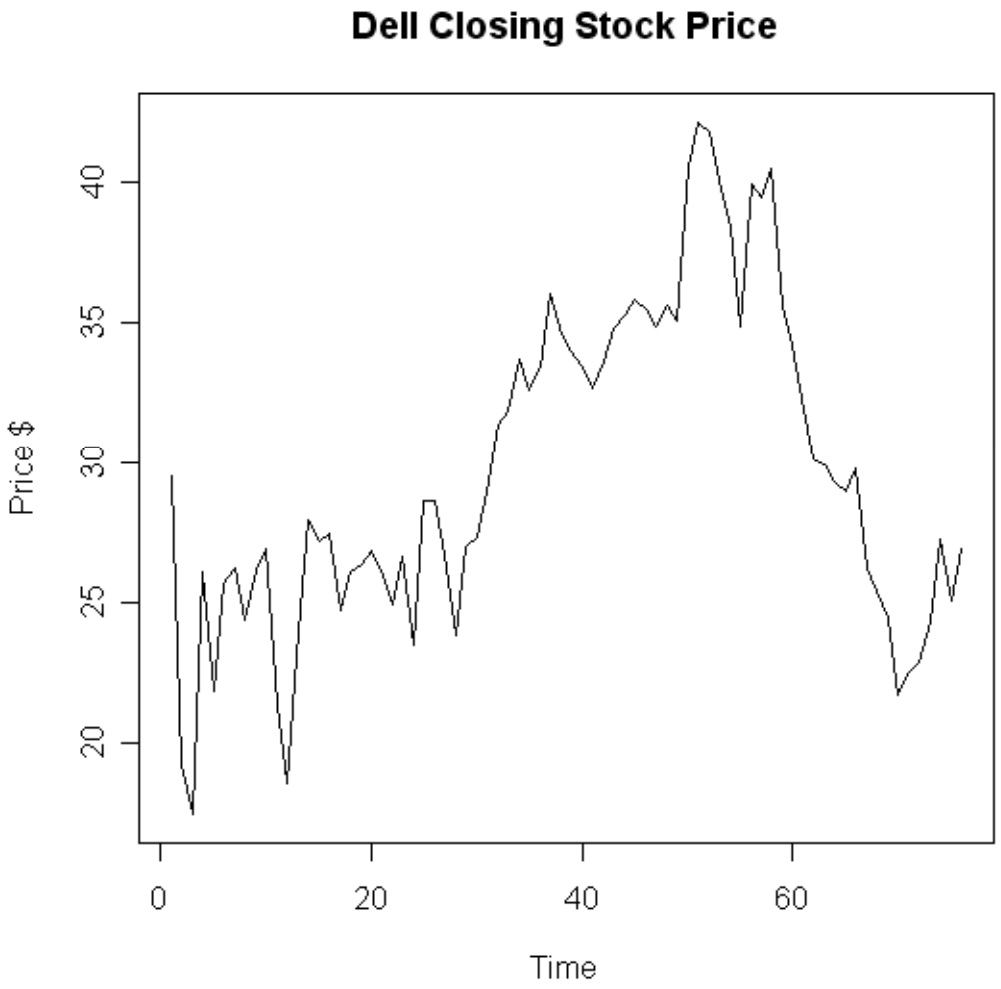


Line Plots




```
plot(t1,D2$DELL,type="l")
```

Line Plots



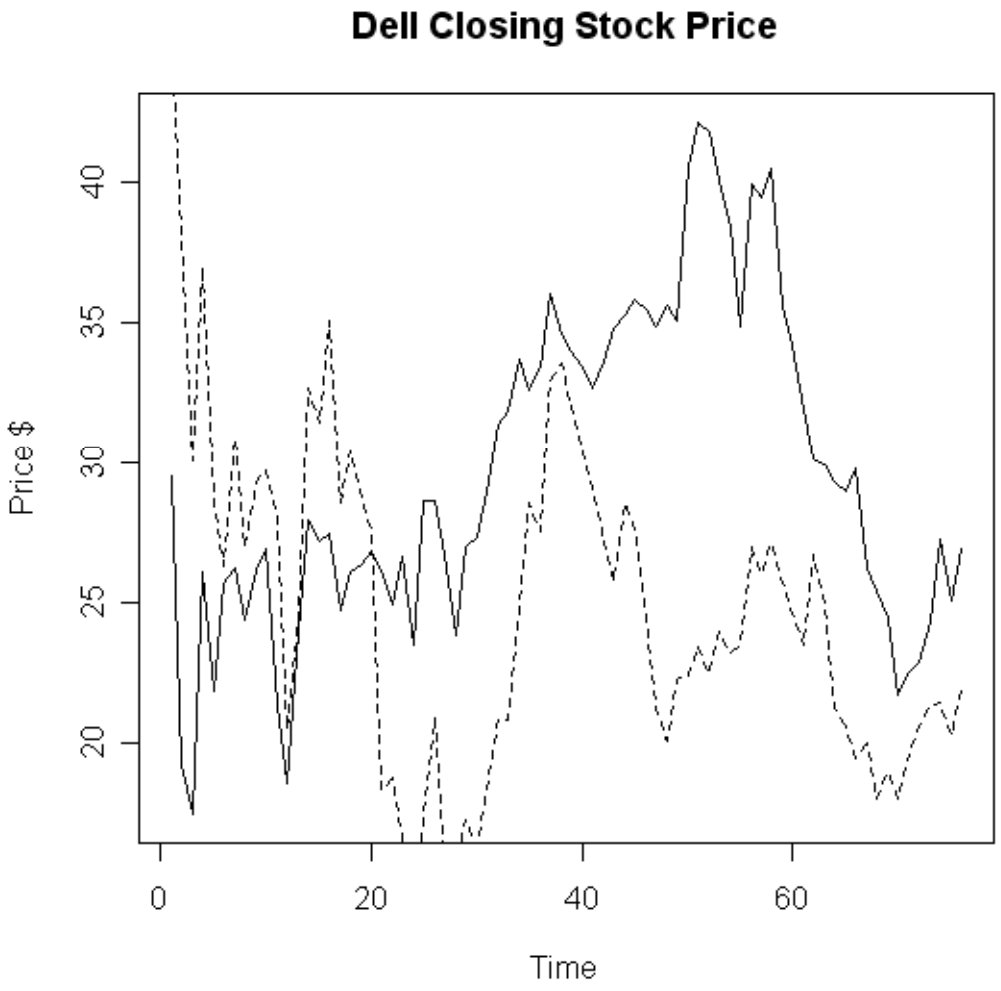

```
plot(t1,D2$DELL,type="l",main='Dell Closing Stock Price',
xlab='Time',ylab='Price $'))
```

Overlaying Plots

- Often we have more than one variable measured against the same predictor (X).
  - `plot(t1,D2$DELL,type="l",main='Dell Closing Stock Price',xlab='Time',ylab='Price $'))`
  - `lines(t1,D2$Intel)`

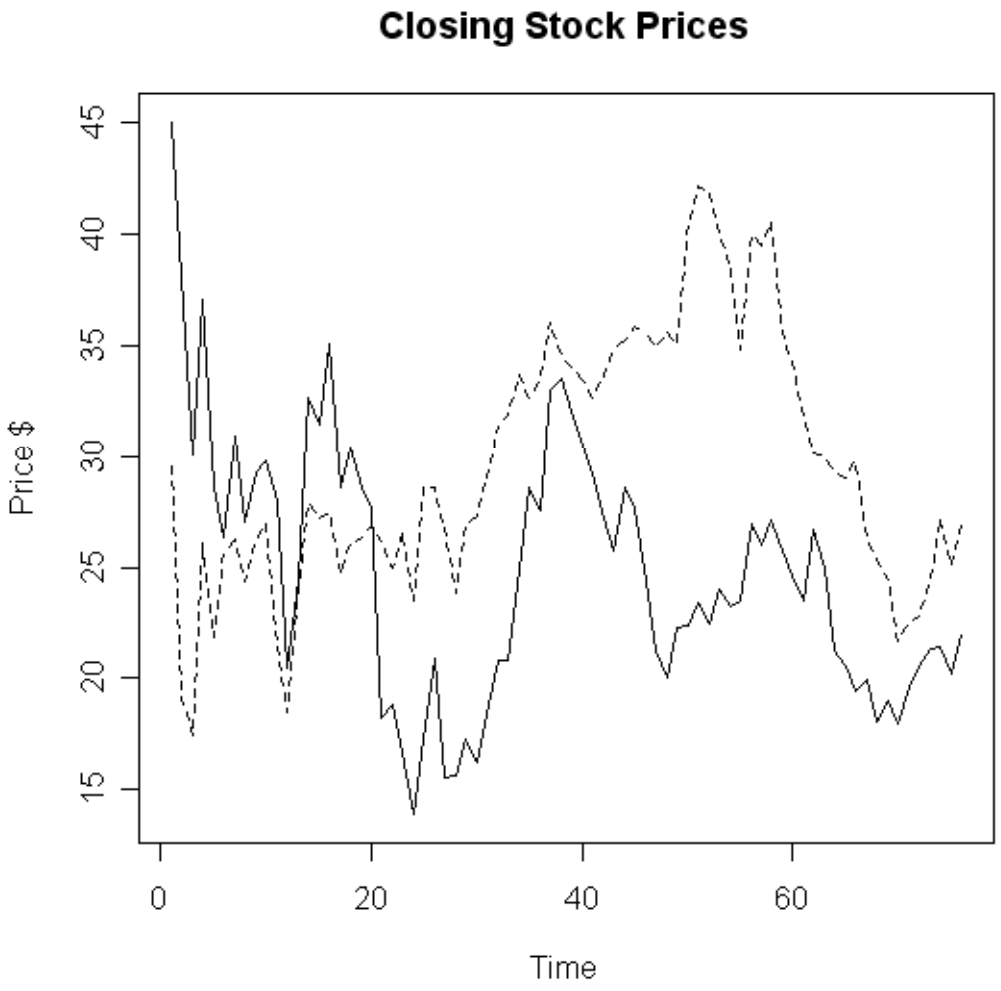
Overlaying Graphs

Overlaying Graphs



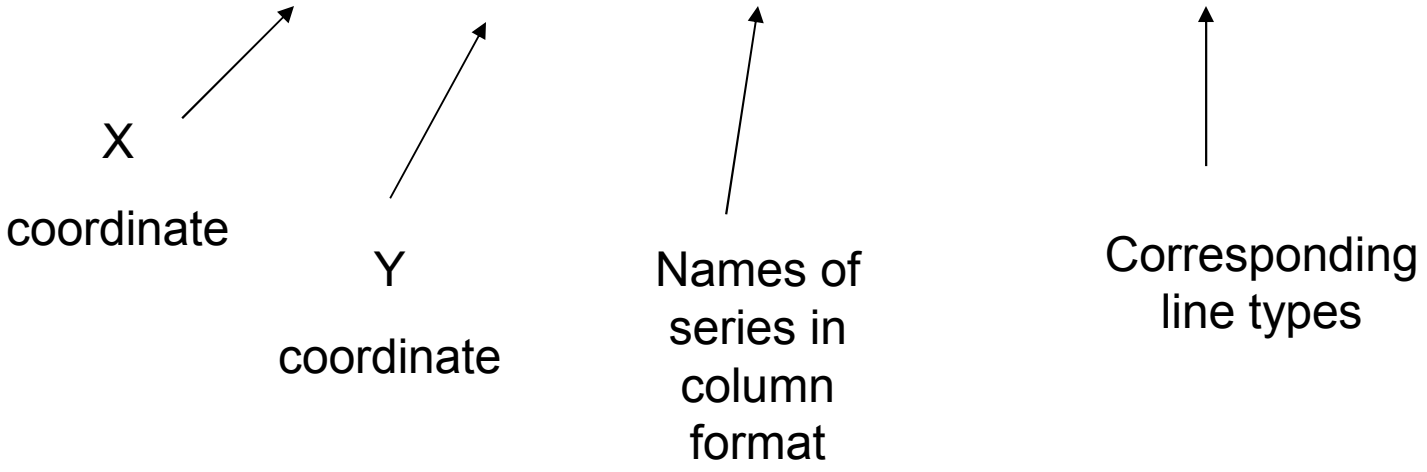

```
lines(t1,D2$Intel,lty=2)
```

Overlaying Graphs



Adding a Legend

- Adding a legend is a bit tricky in R.
- Syntax
- `legend( x, y, names, line types)`

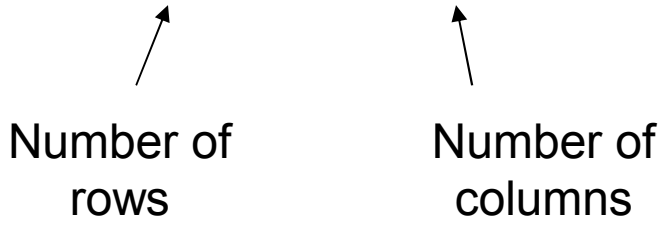


Adding a Legend


```
legend(60,45,c('Intel','Dell'),lty=c(1,2))
```

Paneling Graphics

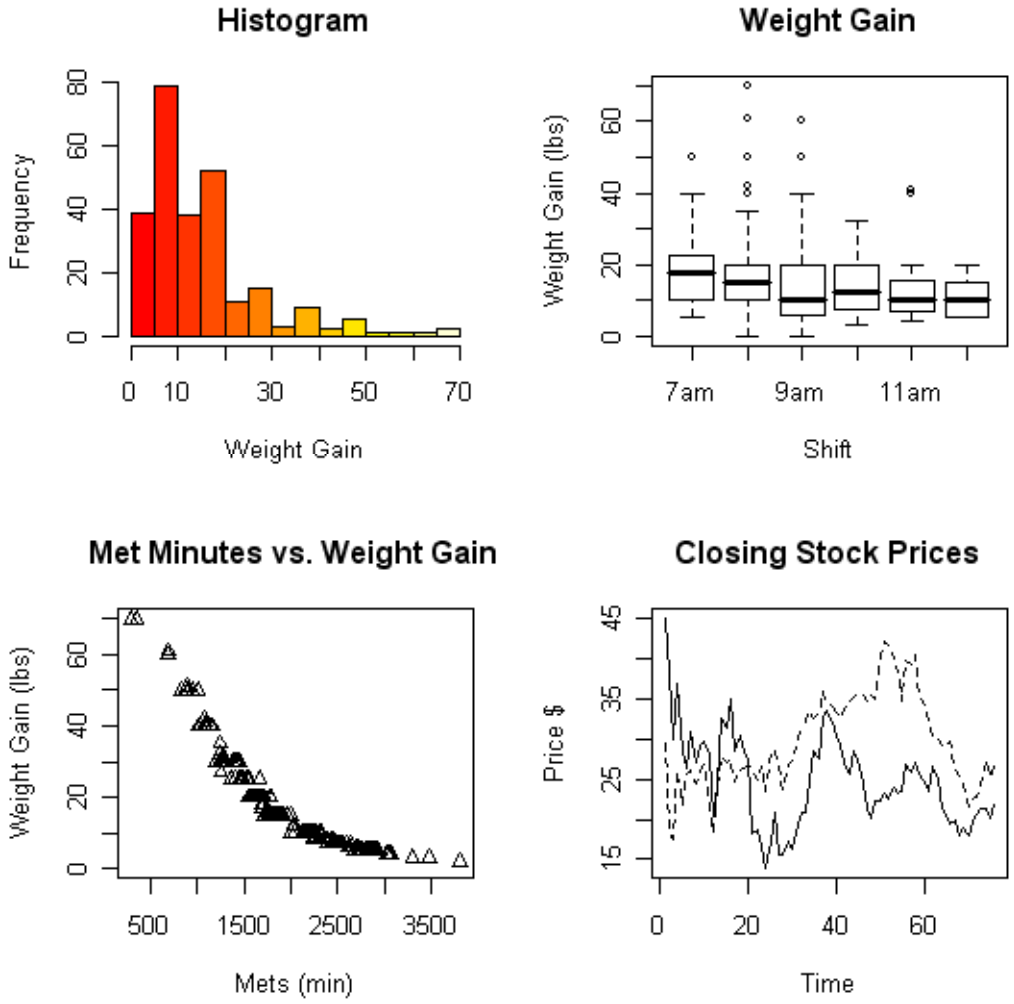
- Suppose we want more than one graphic on a panel.
- We can partition the graphics panel to give us a framework in which to panel our plots.
- `par(mfrow = c( nrow, ncol))`



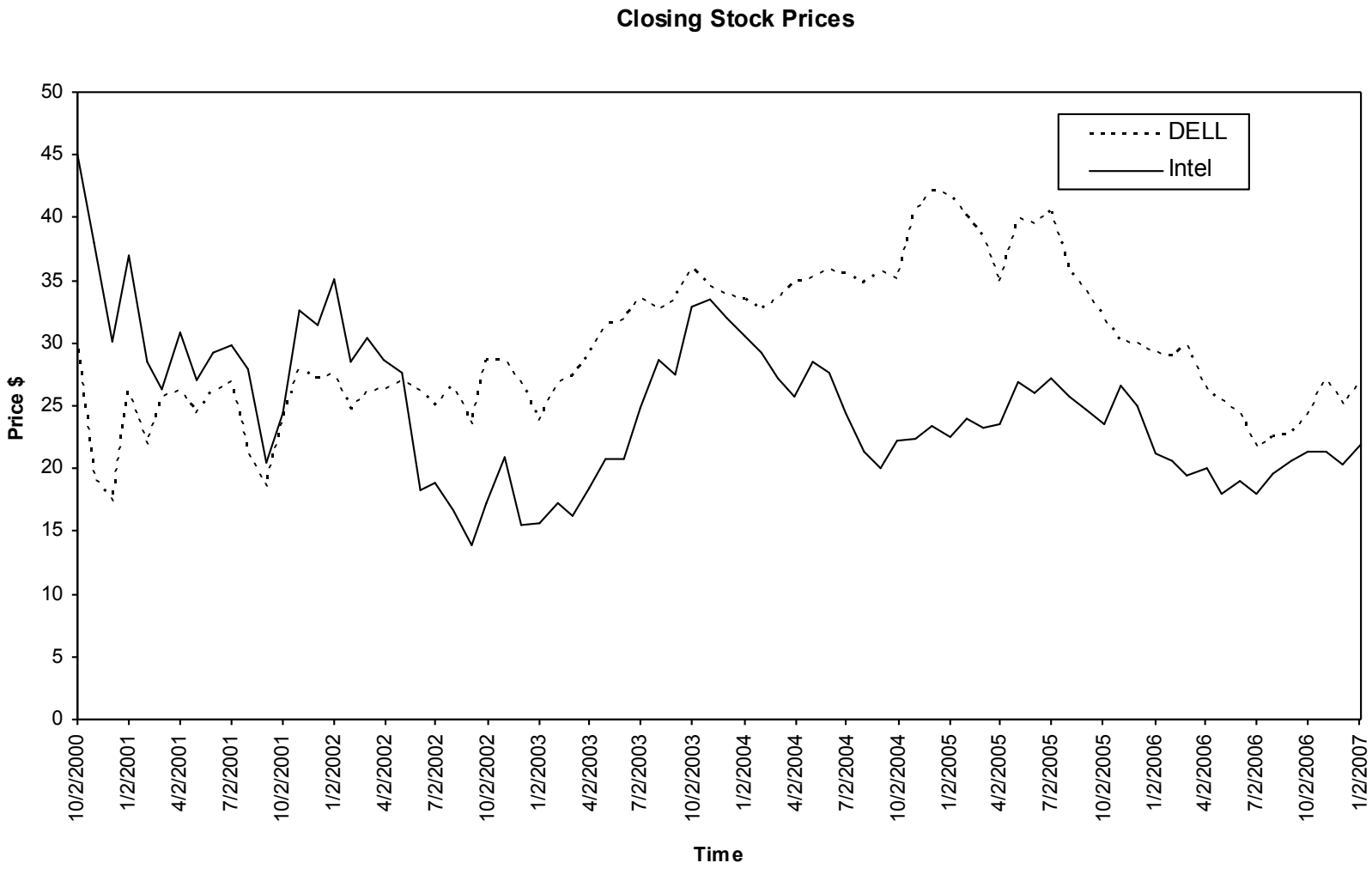
Paneling Graphics

- Consider the following
- `par(mfrow=c(2,2))`
- `hist(D$wg, main='Histogram',xlab='Weight Gain', ylab ='Frequency', col=heat.colors(14))`
- `boxplot(wg.7a$wg, wg.8a$wg, wg.9a$wg, wg.10a$wg, wg.11a$wg, wg.12p$wg, main='Weight Gain', ylab='Weight Gain (lbs)',`
- `xlab='Shift', names = c('7am','8am','9am','10am','11am','12pm'))`
- `plot(D$metmin,D$wg,main='Met Minutes vs. Weight Gain', xlab='Mets (min)',ylab='Weight Gain (lbs)',pch=2)`
- `plot(t1,D2$Intel,type="l",main='Closing Stock Prices',xlab='Time',ylab='Price $')`
- `lines(t1,D2$DELL,lty=2)`

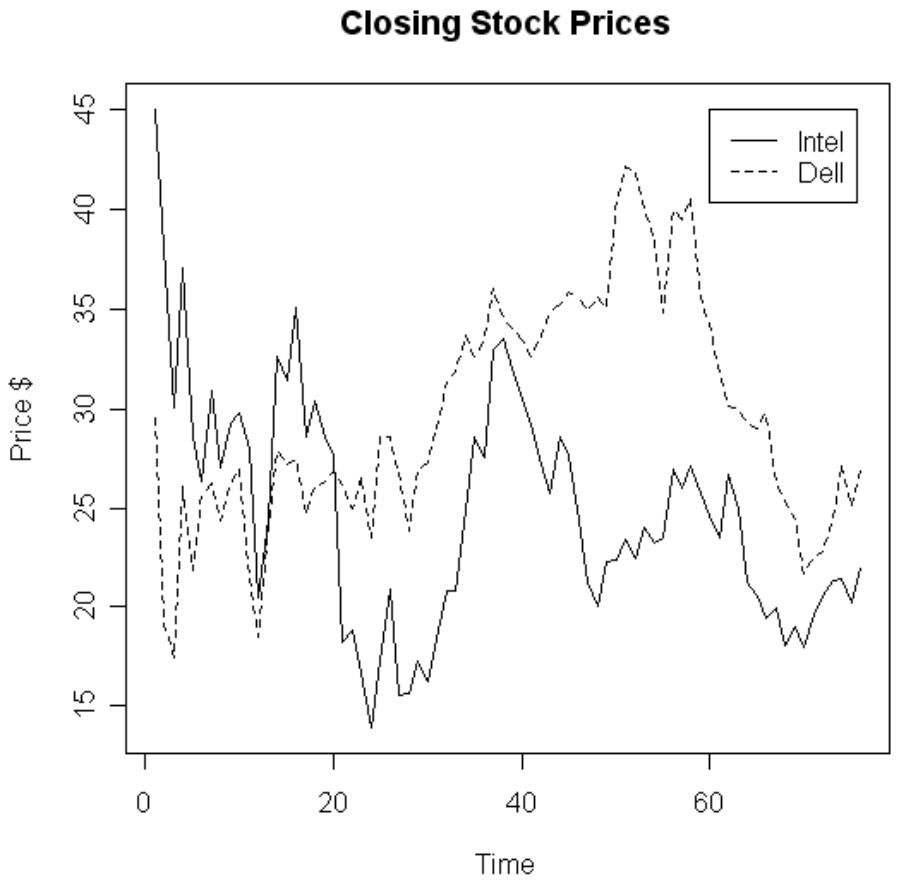
Paneling Graphics



Quality - Excel



Quality - R


Closing Stock Prices
Intel
Dell
Price $
15
20
25
30
35
40
45
0
20
40
60
Time

Statistical Functions

Excel	R
NORMSDIST	pnorm(7.2,mean=5,sd=2)
NORMSINV	qnorm(0.9,mean=5,sd=2)
LOGNORMDIST	plnorm(7.2,meanlog=5,sdlog=2)
LOGINV	qlnorm(0.9,meanlog=5,sdlog=2)
GAMMADIST	pgamma(31, shape=3, scale =5)
GAMMAINV	qgamma(0.95, shape=3, scale =5)
GAMMALN	lgamma(4)
WEIBULL	pweibull(6, shape=3, scale =5)
BINOMDIST	pbinom(2,size=20,p=0.3)
POISSON	ppois(2, lambda =3)

Summary

- All of the R code and files can be found at:
- http://www.cran.r-project.org/