Intro to R & Reference

Joseph Paul Cohen
Henry Z. Lo

What is R?

A Recent Research Revolution™

Matrix Playground

Data Understanding

Functional language (effectively)
(variables can be functions)

Why R?

What R should be used for:

Statistics
Machine learning
Plotting
Concise syntax for common data tasks
Exploring data

Installing Packages

Full list of packages here:
http://cran.r-project.org/web/packages/available_packages_by_name.html

```
install.packages('ggplot2')
library(ggplot2)

install.packages('png')
library(png)

install.packages('vegan')
require(vegan)

library(lattice)
```

Help

```
> ?plot
Usage:
     plot(x, y, ...)
…
Examples:
     require(stats)
     plot(cars)
     lines(lowess(cars))
     plot(sin, -pi, 2*pi) # see ?plot.function
     ## Discrete Distribution Plot:
     plot(table(rpois(100, 5)), type = "h", col = "red", lwd =
          main = "rpois(100, lambda = 5)")
     ## Simple quantiles/ECDF, see ecdf() {library(stats)} for
     plot(x <- sort(rnorm(47)), type = "s", main = "plot(x, typ
     points(x, cex = .5, col = "dark red")
```

String Operations

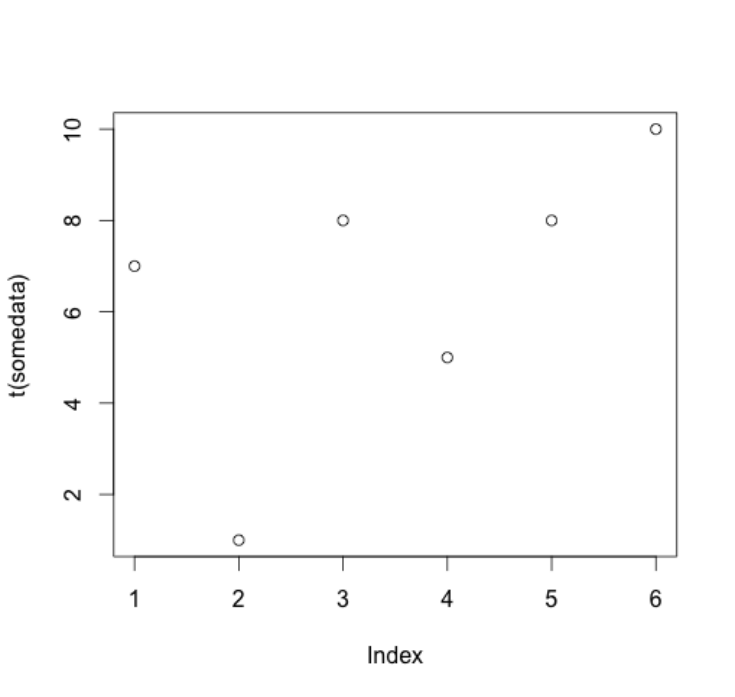
```
> w = c("Hello", "World")
> w[1]
[1] "Hello"
> w[2]
[1] "World"

> w = paste("Hello", "World")
> w
[1] "Hello World"

> w = paste("Hello", "World", sep="")
> w
[1] "HelloWorld"
```

Sorting Data

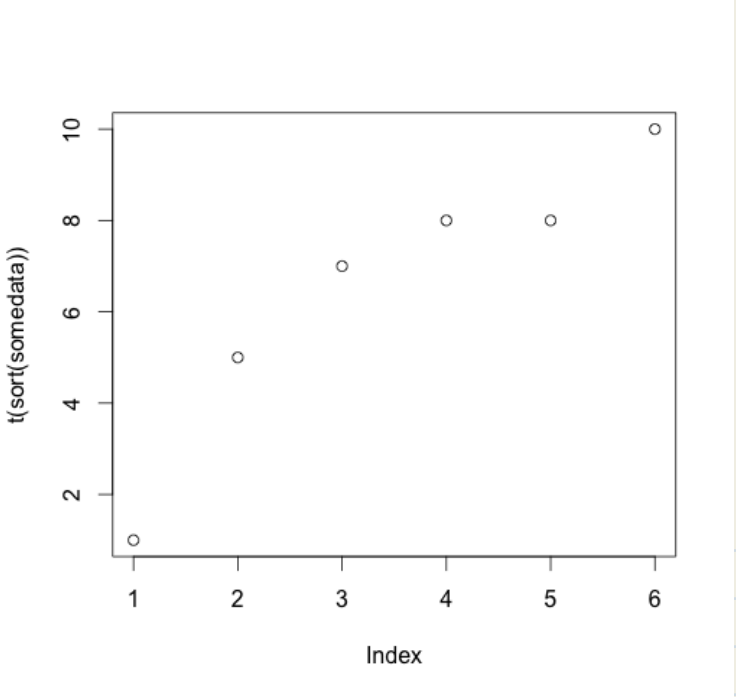
```
#make a random row dataframe
> s = sample(1:10,6,replace=T)
> somedata = as.data.frame(t(s))
#view it
> somedata
  V1 V2 V3 V4 V5 V6
1  7  1  8  5  8 10

> sort(somedata)
  V2 V4 V1 V3 V5 V6
1  1  5  7  8  8 10
> sort(somedata)[2]    # get second lowest value
  V4
1  5
> colnames(sort(somedata))[2]
[1] "V4"
```

Sorting Data

```
> plot(t(somedata))
```

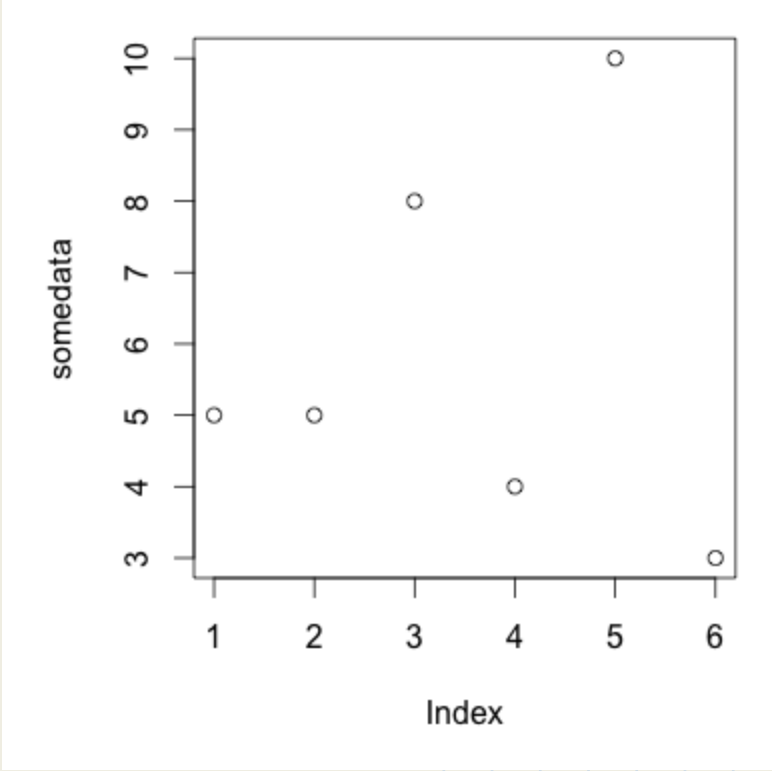



```
> plot(t(sort(somedata)))
```



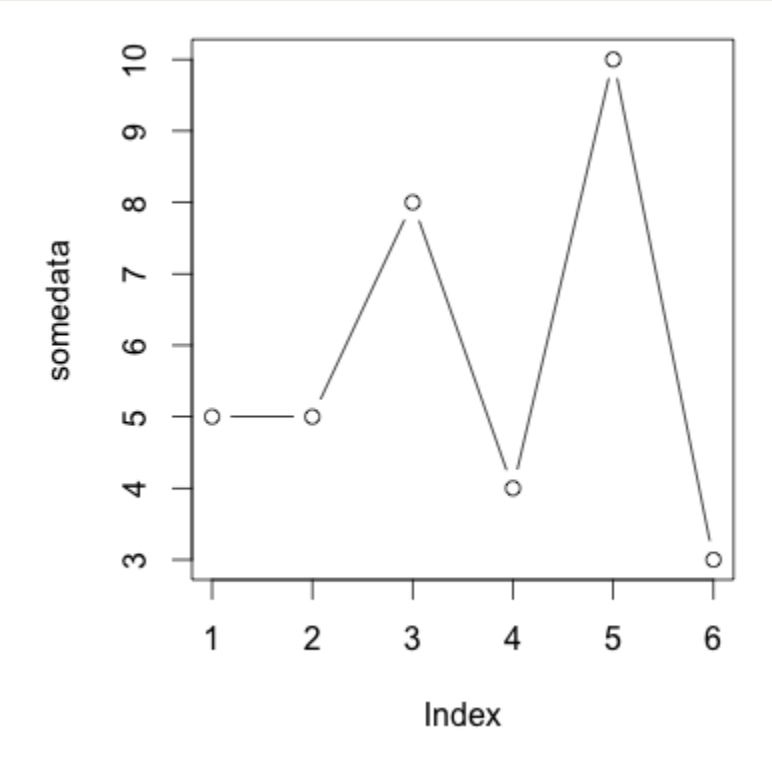
2-D Plots

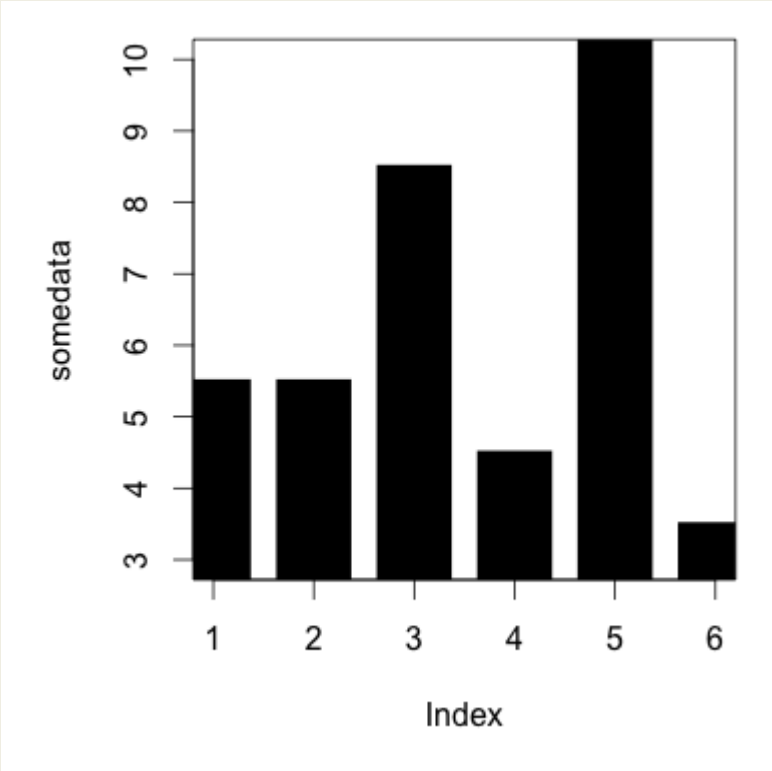
```
> somedata = sample(1:10,6,replace=T)
> somedata
[1]  5  5  8  4 10  3

#make it a vector
> somedata = as.matrix(somedata)
> somedata
     [,1]
[1,]    5
[2,]    5
[3,]    8
[4,]    4
[5,]   10
[6,]    3

#look at it
> plot(somedata)
```



2-D Plots

```
#we want line charts!
> plot(somedata, type="b")
# and bar charts
> plot(somedata, type="h",lwd=50, lend="square")
```



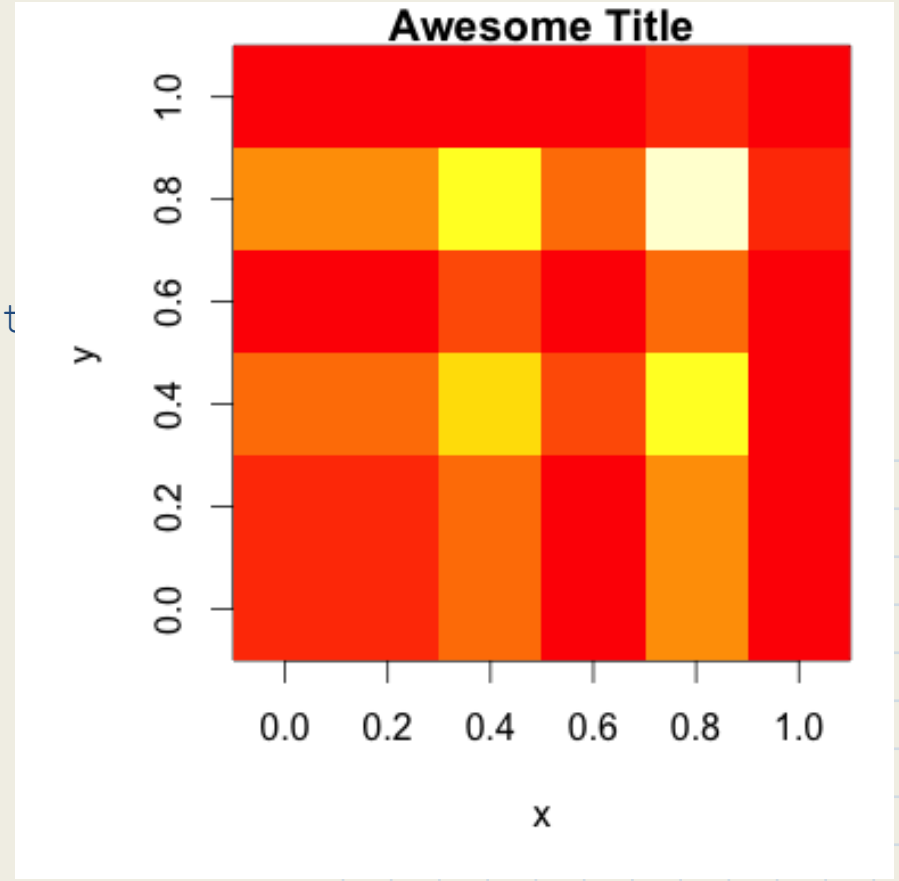


Matrix Operations

```
#transpose and multiply vectors
> t(somedata) %*% somedata
     [,1]
[1,]  239

# We really want a matrix though
> somedata %*% t(somedata)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   25   25   40   20   50   15
[2,]   25   25   40   20   50   15
[3,]   40   40   64   32   80   24
[4,]   20   20   32   16   40   12
[5,]   50   50   80   40  100   30
[6,]   15   15   24   12   30    9
```

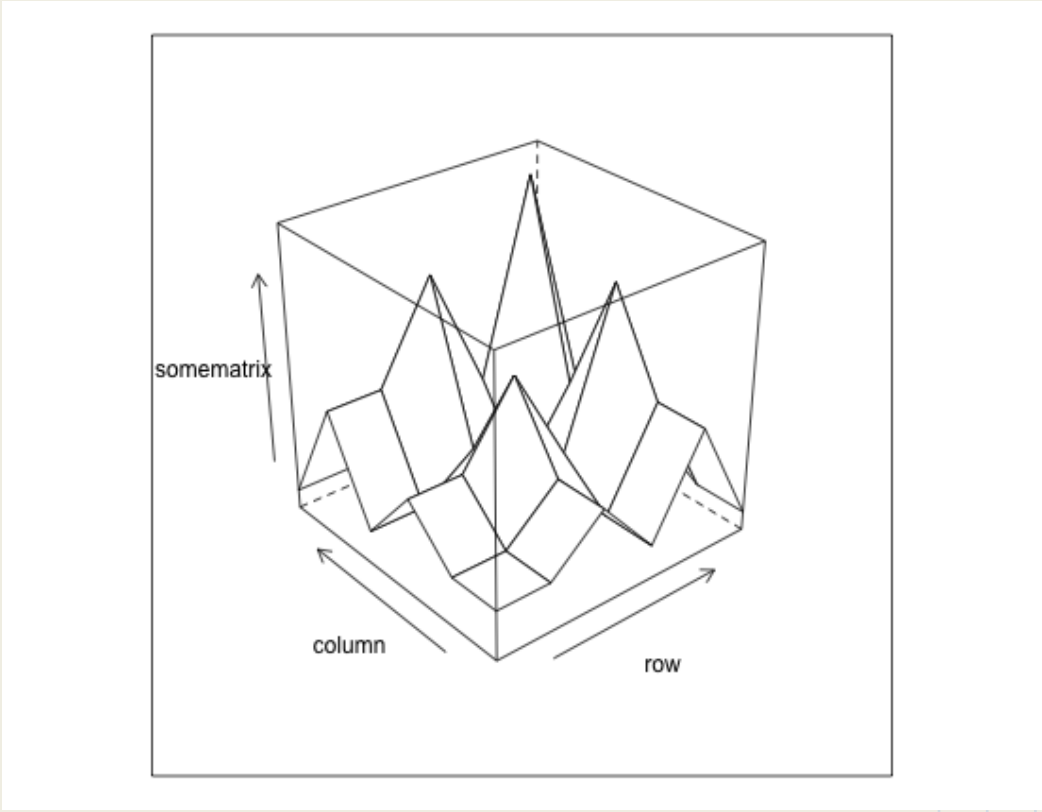
Heatmaps

```
#save this to a variable
> somematrix = somedata %*% t(somedata)

# lets see it
> image(somematrix)


# and then throw some text on it
> image(somematrix,
main="Awesome Title",
xlab="x",
ylab="y")
```



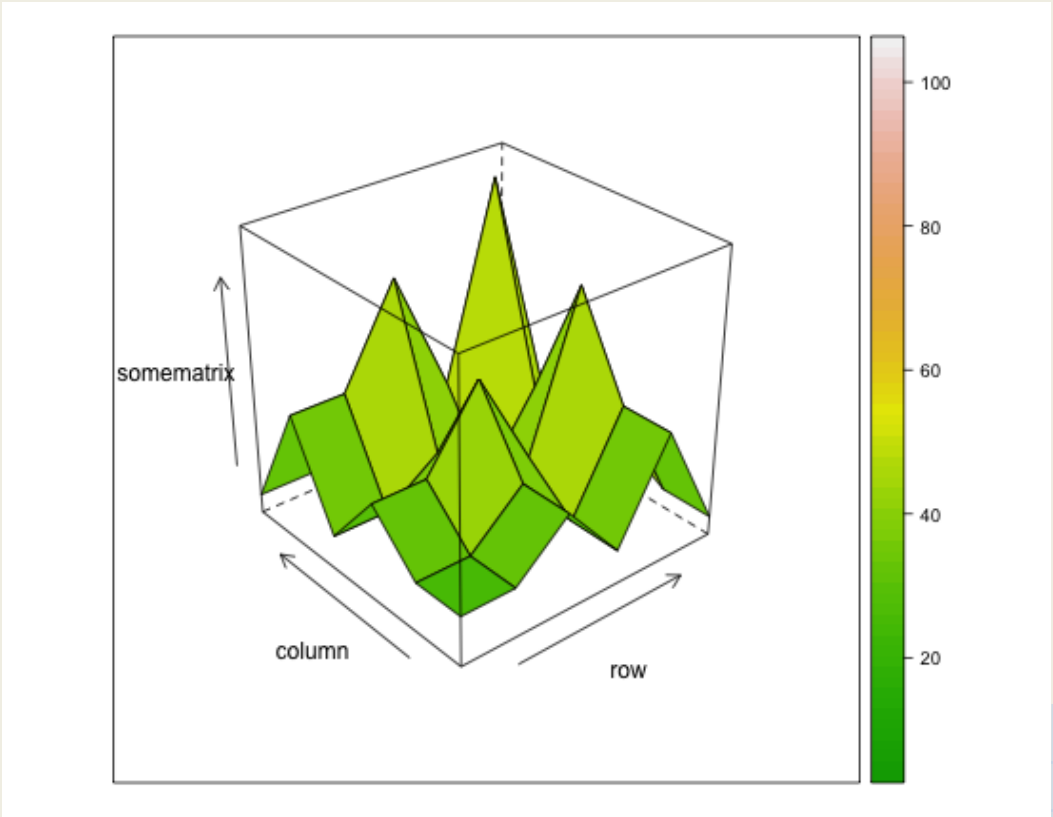
Wireframe Plots

```
# wireframe
> library(lattice)
> wireframe(somematrix)
```



Wireframe Plots (color)

```
# wireframe with color!
> wireframe(somematrix, drape=TRUE,
            col.regions=terrain.colors(100))
```



Figuring out Data types

```
> object = c(1,2)

> str(object)                  # Display R Object structure
 num [1:2] 1 2

> typeof(object)                  # The Type of an Object
[1] "double"

> dim(object)                     # Get dimensions of object
NULL                           # A list does not have dimensions

> object = matrix(object)
> dim(object)
[1] 2 1
```

Data Frames

```
#Fundamental data type in R

> iris <- read.csv('iris.csv')                    # or data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

> nrow(iris)
[1] 150

> ncol(iris)
[1] 5

# if data is incomplete, na.omit will remove incomplete rows
> iris = na.omit(iris)
```

Data Frames

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

t-test

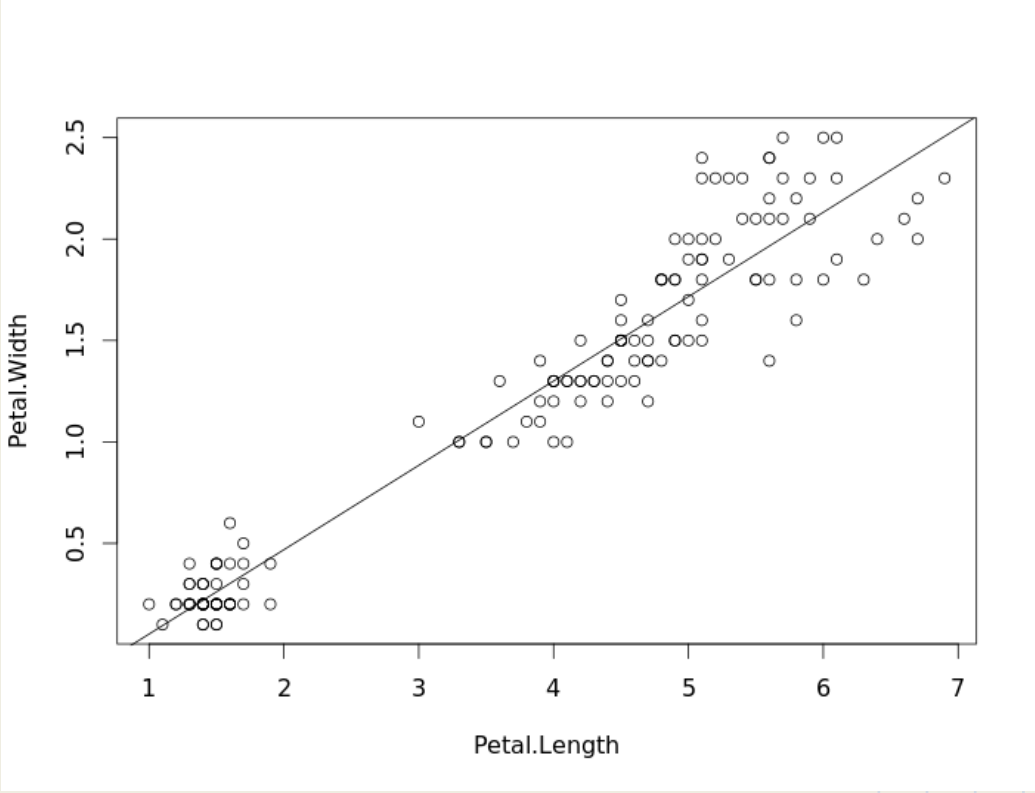
```
> setosas = iris[iris$Species == 'setosa',]
> versicolors = iris[iris$Species == 'versicolor',]
> t.test(setosas$Sepal.Width, versicolors$Sepal.Width)

    Welch Two Sample t-test

data:  setosas$Sepal.Width and versicolors$Sepal.Width
t = 9.455, df = 94.698, p-value = 2.484e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.5198348 0.7961652
sample estimates:
mean of x mean of y
    3.428     2.770
```

Linear Regression

```
> data(iris)

> summary(iris[,c('Petal.Width', 'Petal.Length')])
  Petal.Width     Petal.Length
 Min.   :0.100   Min.   :1.000
 1st Qu.:0.300   1st Qu.:1.600
 Median :1.300   Median :4.350
 Mean   :1.199   Mean   :3.758
 3rd Qu.:1.800   3rd Qu.:5.100
 Max.   :2.500   Max.   :6.900

> linear.equation = lm(data=iris, Petal.Width~Petal.Length)
> linear.equation
Coefficients:
 (Intercept)  Petal.Length
     -0.3631        0.4158
```

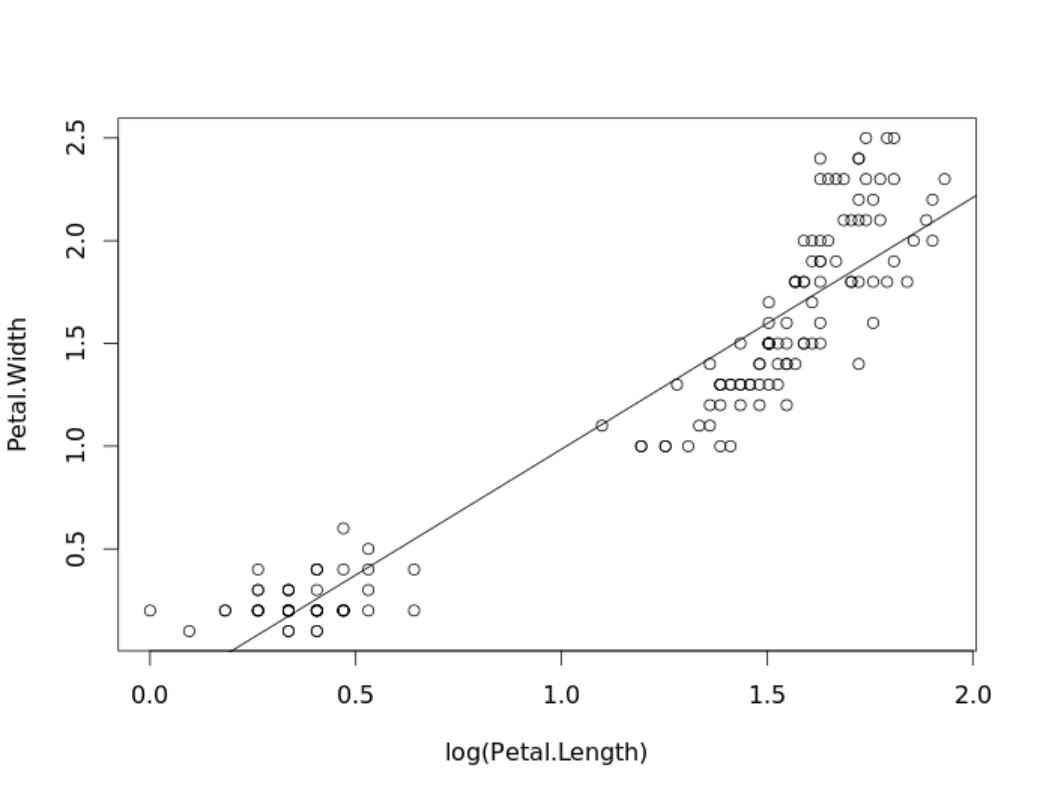
Linear Regression

```
# Petal.Width in terms of Petal.Length for iris dataset
> plot(data=iris, Petal.Width~Petal.Length)
> abline(linear.equation)
```



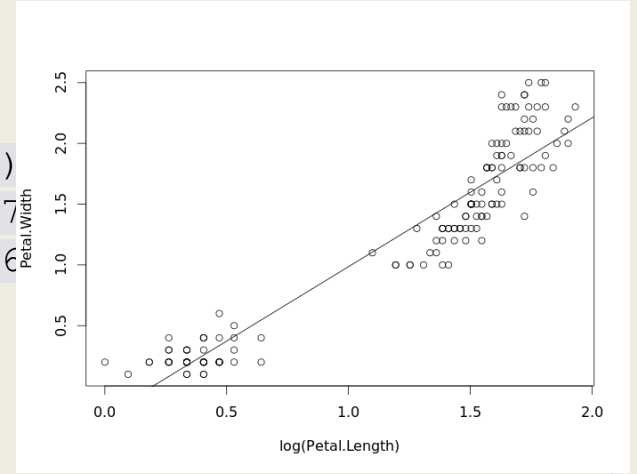
Logarithmic Regression

```
> log.equation = lm(data=iris, Petal.Width~log(Petal.Length))
> plot(data=iris, Petal.Width~log(Petal.Length))
> abline(log.equation)
```

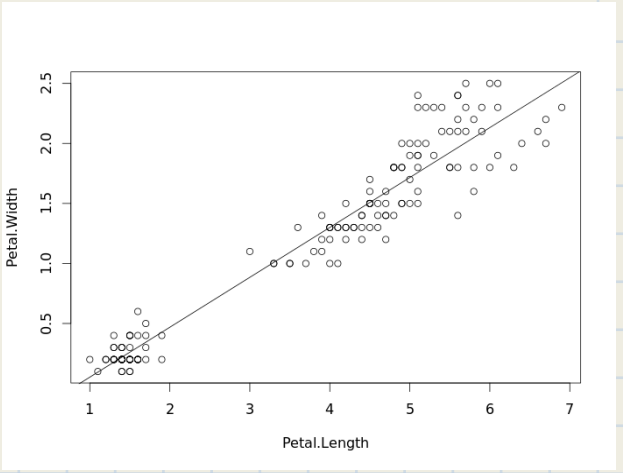


Comparing Regressions

```
> summary(log.equation)

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        -0.23947    0.04441  -5.393 2.69e-07
log(Petal.Length)   1.22448    0.03380  36.232  < 2e-16

Multiple R-squared:  0.8987,
Adjusted R-squared:  0.898
```

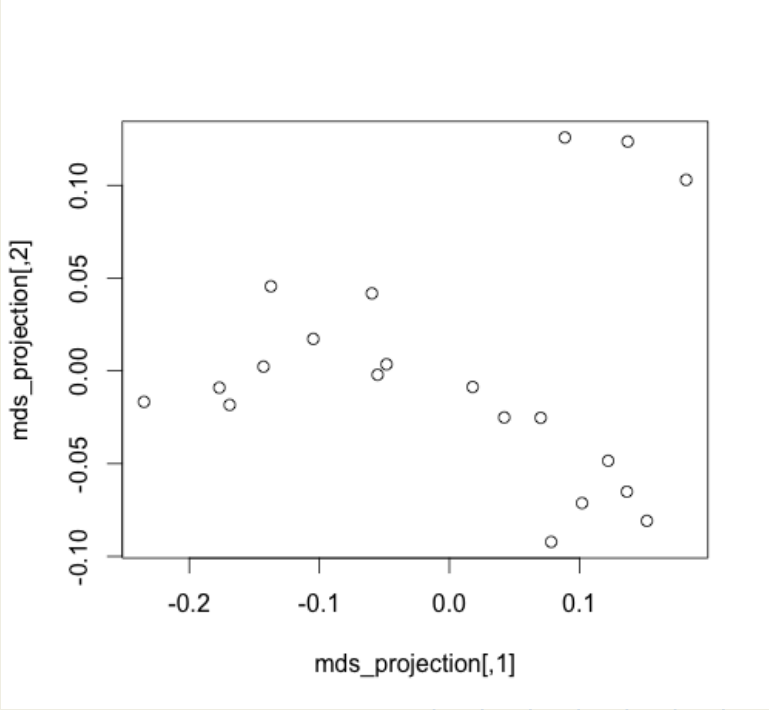



```
> summary(linear.equation)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.363076   0.039762  -9.131  4.7e-16
Petal.Length  0.415755   0.009582  43.387  < 2e-16

Multiple R-squared:  0.9271,
Adjusted R-squared:  0.9266
```



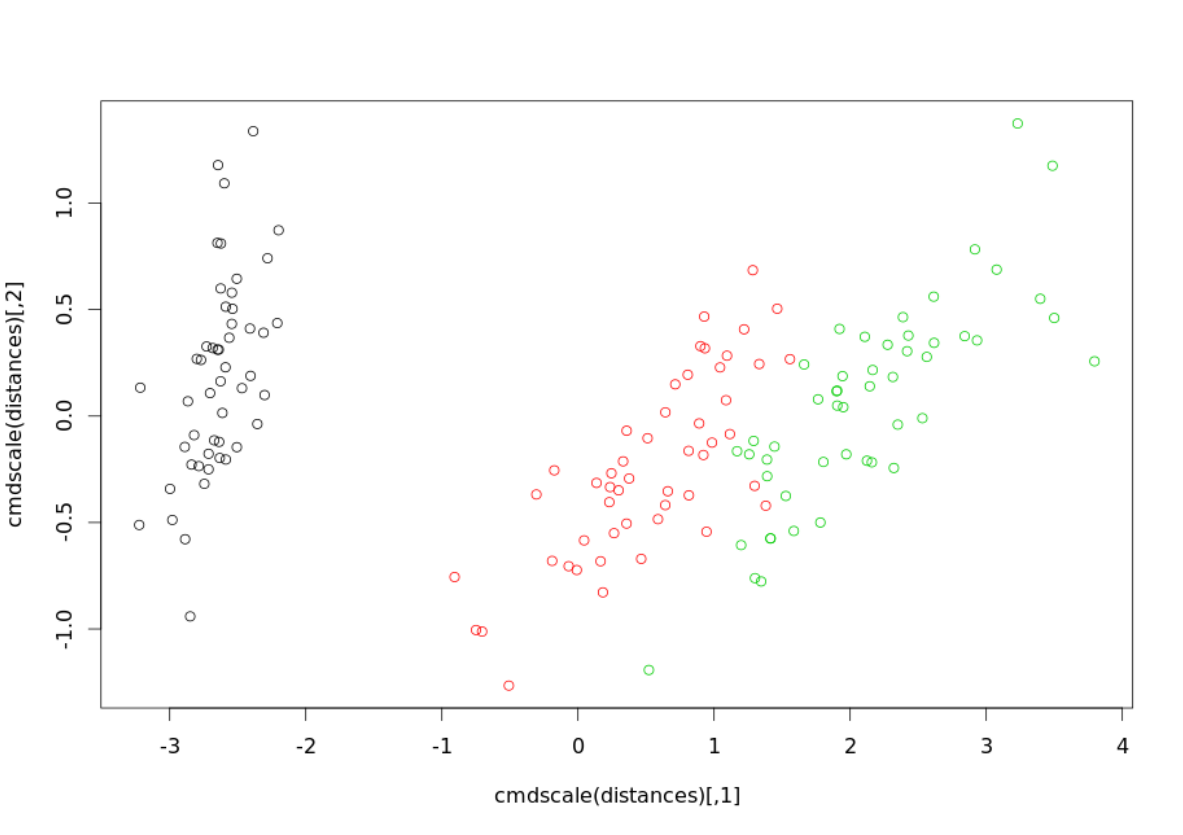
Some MDS/PCA

```
# Get a distance matrix
distance_matrix = read.csv(file="filename.csv",header=FALSE);

mds_projection <- cmdscale(distance_matrix , k=2)#Do MDS/PCA

# Plot it
plot(mds_projection)
```



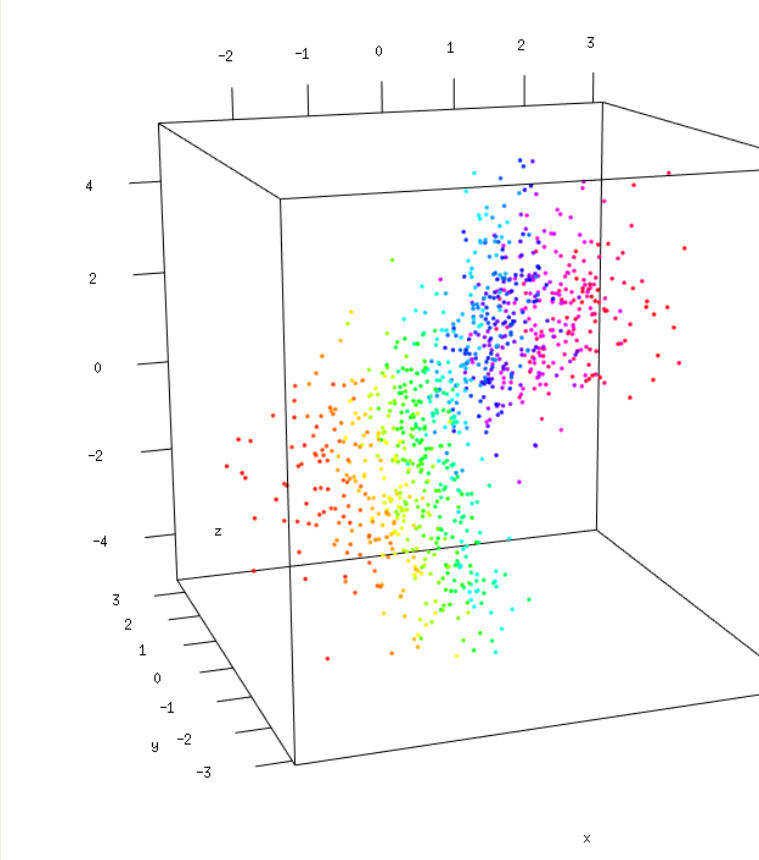
Another MDS Example

```
> distances = dist(iris[,-5])
> projected.distances = cmdscale(distances)
> plot(projected.distances, col=iris$Species)
```



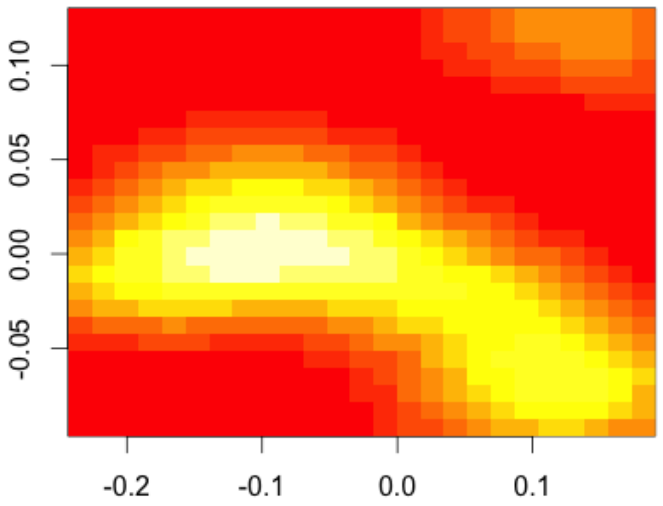
3D Plots

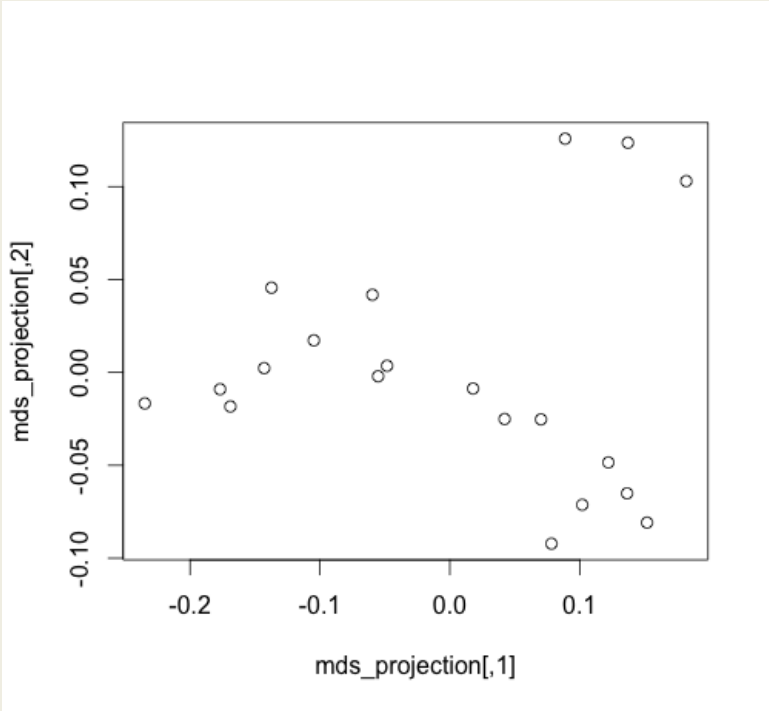
```
#On Ubuntu:
sudo apt-get install freeglut3-dev

> install.packges('rgl')
> library('rgl')
> open3d()
> x <- sort(rnorm(1000))
> y <- rnorm(1000)
> z <- rnorm(1000) + atan2(x,y)
> plot3d(x, y, z, col=rainbow(1000))
```



Gaussian Density Plots

```
> library(MASS)
> x = mds_projection[,1]
> y = mds_projection[,2]
> density = kde2d(x,y) # kernel density estimation, 2-d
> image(density)
```





grep arrays

```
#Filter lists
#Lets make a list
> data = c("aaa","abc","bbc","bba","abb")
> data
[1] "aaa" "abc" "bbc" "bba" "abb"

#Now substring match for a string with a in it
> data[grepl("a",data)]
[1] "aaa" "abc" "bba" "abb"

#Now use a perl regex to match an a at the end of the string
> data[grepl("a$",data,perl=TRUE)]
[1] "aaa" "bba"

#Now at the start
> data[grepl("^a",data,perl=TRUE)]
[1] "aaa" "abc" "abb"
```

Writing a plot to a file

```
# Set the output filename
png("outputfile.png")

# Get a fresh plot
plot.new()

# Do something that writes to a plot
plot(someAwesomeDataset, ...)

# Flush Changes to Disk
dev.off()
```

Functions in R

```
#Make 'addnumbers' a function
addnumbers = function (num1, num2){

   result = num1+num2
   return (result)
}


> addnumbers(1,2)
[1] 3
```

Functions in R

```
# Default values
inc = function (num, by=1){

    result = num+by
    return (result)
}

> inc(1)
[1] 2

> inc(1,10)
[1] 11

> inc(by=10,num=1)
[1] 11
```

Functions on Arrays

```
# add 10 to one element
> inc(4,10)
[1] 14

# add 10 to each element of list
> inc(c(4,4,5,1,3),10)
[1] 14 14 15 11 13

# same thing
> c(4,4,5,1,3) + 10
[1] 14 14 15 11 13

# apply logarithm to each element of list
> log(c(4,4,5,1,3))
[1] 1.386294 1.386294 1.609438 0.000000 1.098612
```

Vectors

Arguments are different in R:

```
2 + 3 == 5
2 + c(1,2,3) == c(3,4,5)
c(2,4) + c(1,3,4,5) == c(3,7,6,9)
```

Subsetting vectors is more powerful than C-like languages

```
a <- c(3,5,7,8)
a[c(TRUE, FALSE, TRUE, FALSE)]  == c(5,8)
a[a>5] == c(7,8)
```

plyr

```
> library(plyr)
> summarise(iris,
            sw.variance = var(Sepal.Width),
            sl.variance = var(Sepal.Length))

         sw.variance       sl.variance
1          0.1899794         0.6856935

> ddply(iris, 'Species', summarise
        sw.variance = var(Sepal.Width),
        sl.variance = var(Sepal.Length))

     Species       sw.variance      sl.variance
1     setosa        0.14368980        0.1242490
2 versicolor        0.09846939        0.2664327
3  virginica        0.10400408        0.4043429
```

Web Links

Presentation will be hosted at:
http://josephpcohen.com

R-Project main website
http://www.r-project.org/

R Development Environment
http://www.rstudio.com/