Introduction to Machine Learning

Augustin Luna
15 January, 2016

Research Fellow
Department of Biostatistics and Computational Biology
Dana-Farber Cancer Institute

Topics to be Covered

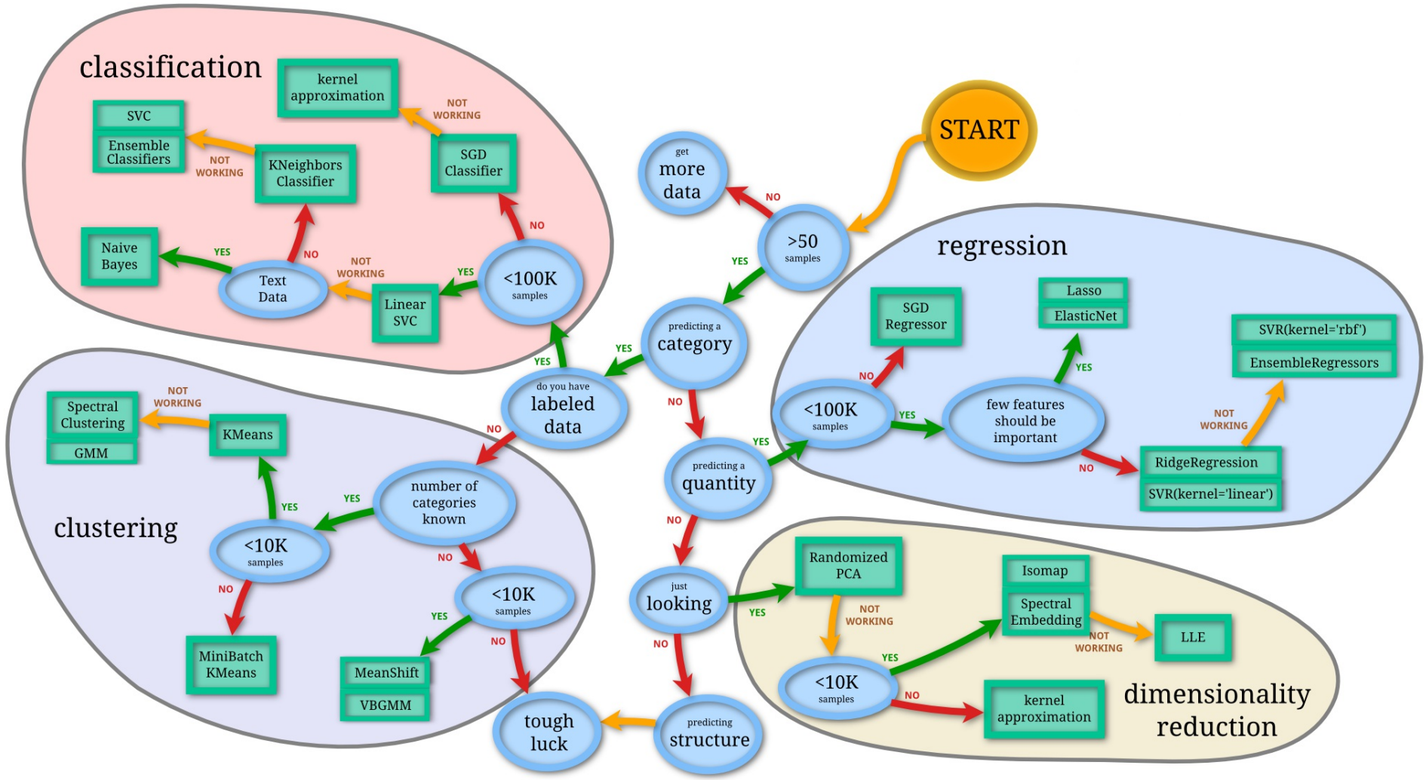
- Correlation Test
- Enrichment Analyses
- Multiple Testing Correction
- Regression
- Clustering
- Dimensionality Reduction

What is Machine Learning?

- Machine learning has the general goal of X
  - Related to the areas of informatics, data mining, data science, machine learning, and statistics
- Concepts covered here are important to the areas of above

Machine Learning Techniques

- The flowchart below helps find the right method for a given problem
  - http://scikit-learn.org/stable/tutorial/machine_learning_map/



Methods to be Covered

- Basic techniques will be covered to build confidence with R and the general concepts
  - Dimensionality Reduction: Principal Component Analysis (PCA)
  - Regression: Linear regression
  - Clustering: Hierarchical and K-means clustering
  - Classification will **NOT** be covered.

Correlating Two Vectors

```r
# Make sure the random numbers are always the same
set.seed(1)

# Generate two sets of 20 random numbers
a <- runif(20); b <- runif(20)

# Calulate the correlation of the two sets
cor.test(a, b)
```

```
    Pearson's product-moment correlation

data:  a and b
t = -1.0368, df = 18, p-value = 0.3136
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.6152730  0.2292138
sample estimates:
       cor
-0.2373854
```

Extracting Values From Results

- Values in results are described in the help `?cor.test`
- p-value Extracted
  - A p-value is the probability of seeing results as extreme as the ones produced in an analysis.

```
set.seed(1)

a <- runif(20)
b <- runif(20)

results <- cor.test(a, b, method="pearson")
names(results)
```

```
[1] "statistic"   "parameter"   "p.value"     "estimate"
"null.value"
[6] "alternative" "method"      "data.name"   "conf.int"
```

```
results$p.value
```

```
[1] 0.3135682
```

Over-Representation (ORA) and Enrichment Analyses

- Enrichment tests are widely used in biology to determine if the genes contain a trait more frequently than a random sampling of genes
  - Gene Ontology (GO) term (e.g. biological process, molecular function, or cellular component) and pathways are the most common comparisons made
- Several tools exist for doing enrichment analyses
  - The tests are either Fisher's exact test or hypergeometric test; these tests produce the same results
  - These calculations can be done in R using `fisher.test` and `phyper`

Calculating an ORA (Enrichment) P-Value

The significance of an over-representation (enrichment analysis) is calculated using a hypergeometric test:

$$P(k_i) = 1 - \sum \frac{\binom{M}{n}\binom{N-M}{n-m}}{\binom{N}{n}}$$

where

- N: number of studied genes,
- n: total number of genes identified by a previous analysis
- M: total number of genes ina pathway k_i
- m: number genes previously identified genes in pathway k_i

Enrichment Analysis Example

```
totalEntriesNum <- 19551
clusterEntryNum <- 181
gmtEntryNum <- 449
entriesInGmtNum <- 17

mat <- matrix(c(entriesInGmtNum, gmtEntryNum,
clusterEntryNum, totalEntriesNum), 2, 2)
results <- fisher.test(mat, alternative='greater')
results$p.value
```

```
[1] 3.693347e-06
```

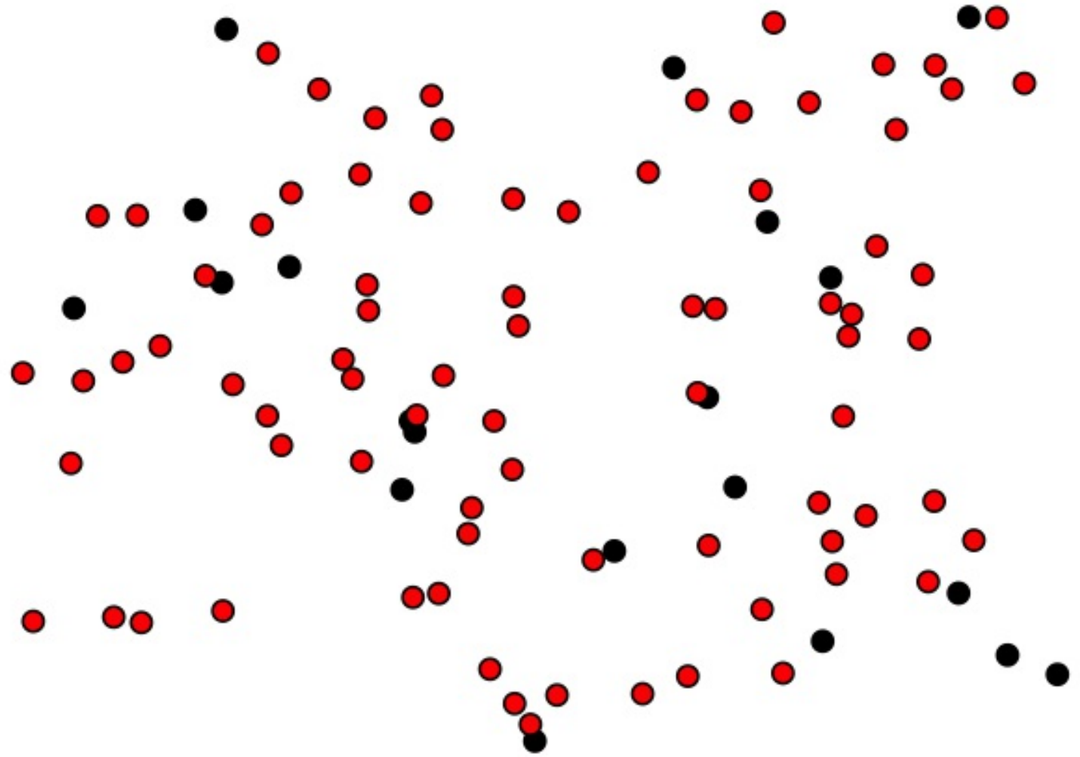
```
pval <- phyper(entriesInGmtNum-1,
               gmtEntryNum,
               sum(mat)-gmtEntryNum,
               clusterEntryNum,
               lower.tail=FALSE)
pval
```

```
[1] 6.43827e-07
```

```
# pval <- phyper(entriesInGmtNum-1,
```

Enrichment Analysis

- What is the probability of randomly drawing at least 4 black points in a random sample of 5 points?
  - The concept of “black” could be replaced by “genes from a given pathway” or “genes with a common function”



Multiple Testing Correction for Enrichment Analyses

- Enrichment analyses often do analyses over a large number of molecular functions or pathways
- If we conduct many such tests, we are likely to see false positives
  - A p-value significance cutoff of 0.05 means that we expect 1 test out of 20 to appear significant by random chance (i.e. a false positive)

Multiple Test Corrections Types

- Family-Wise Error Rate (FWER): Controls the probability that any test is a false positive
  - Bonferroni Correction: Very stringent correction
  - Adjusted p-value = original p-value * number of tests
- False Discovery Rate (FDR): Controls the proportion of tests that are false positives
  - Widely used alternative to FWER (e.g. Bonferroni correction)
  - Example (next slide)

Benjamini-Hochberg (BH) FDR Procedure

- α denotes the desired false positive rate, calculate the new p-value cutoff
  - Assume $n = 100, \alpha = 0.05$

1. Rank n p-values
2. Calculate q-values

$$q = \alpha \times \frac{n - rank + 1}{n}$$

3. Select the lowest ranked p-value that is lower than α

Benjamini-Hochberg Correction Example

p-value	Rank	q-value	p < q
0.9	1	0.05*1.00=0.05	FALSE
0.7	2	0.05*0.99=0.0495	FALSE
0.5	3	0.05*0.98=0.049	FALSE
0.04	4	0.05*0.97=0.0485	TRUE
…	…	…	…
0.005	n	0.05*0.01=5E-4	FALSE

Bonferroni and Benjamini-Hochberg Corrections in R

- Simulate the p-values with random numbers

```
pVals <- runif(10000)

pValsAdjusted <- p.adjust(pVals,
method="bonferroni")
head(pValsAdjusted)
```

```
[1] 1 1 1 1 1 1
```

```
# 'fdr' or 'BH' can be used for the
Benjamini-Hochberg method
pValsAdjusted <- p.adjust(pVals,
method="fdr")
head(pValsAdjusted)
```

Additional Enrichment Analyses

- Gene Set Enrichment Analysis (GSEA): GSEA is one of the best known enrichment analyses
  - This method additionally takes into account numeric values associated with the genes (e.g. gene expression levels)
  - They provide many collections of “gene sets” that can be used with GSEA or related methods
  - http://software.broadinstitute.org/gsea/msigdb

Regression

Example: Find the relationship between the expression of a set of genes and drug response

Given n observations each with a response variable y and p predictors (or features)

$$Y = (y_1, \dots, y_n)^T, \quad n \times 1$$
$$X = (X_1, \dots, X_p), \quad n \times p$$

Goal: We want to find a set of regression coefficients β for $x = (x_1, \dots, x_p)$ to describe the relationship between y and $x_1, \dots, x_p$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots$$

- $\hat{y}$ is the predicted value
- $\hat{\beta}$ are the predicted regression coefficients (as opposed to the true coefficients)

Example Regression

```
fit <- lm(Petal.Width ~ Petal.Length, data=iris)
summary(fit)
```

```

Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.56515 -0.12358 -0.01898  0.13288  0.64272 

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 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9266 
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

Interpreting lm() Results Summary

- Residuals: The difference between the actual and predicted values
- Estimate: Regression coefficient estimates
- Std. Error: Measurement of the variability of the coefficient estimate. Lower is better.
- t value: Coefficient score to describe the importance of predictor; used to calculate the p-value
- Pr(>|t|): Coefficient p-value. Probability the predictor is **NOT** relevant
- R-square: Score for evaluating how well the model fits the data. Higher is better. This can be adjusted for the number of predictors used in the model.
  - Values ~0.7 are of more interest, but there is no standard rule
- More information: http://blog.yhat.com/posts/r-lm-summary.html

Some Issues with Regression

- Missing Data: What if your data has missing values?
  - Imputation can be used to fill in missing values using other data points
- Too many predictors versus the number of samples
  - The “Curse of Dimensionality” (later slide)
  - Regularized regression methods can be used to select features to be included in the model
- Overfitting: Will your model work on other datasets?
  - Excessively complex models (e.g. having too many predictors) can have poor predictive performance when tested on new data
  - Regularized regression methods have properties to avoid overfitting

Missing Data

- By default, `lm()` removes rows that contain missing values
- An alternative is imputation to fill in missing values
  - `impute` imputes using K-Nearest Neighbors (KNN)
  - Step 1: Identify K number of neighbors based on Euclidean distance
  - Step 2: Average the values of the neighbors and replace the missing value

Basic Rules for Imputation

- Should be done when the number of missing values is small
  - A safe maximum threshold is 5% of the total for large datasets
- Should be done when the imputed values are plausible for the missing values
- Should be done when it is assumed that the missing values occur at random
  - If missing values do not occur at random, the data collection should be investigated and/or the values should be dropped

The "Curse of Dimensionality"

- If the number of predictors is greater than the number of samples, it will not be possible to estimate relevant regression parameters in the full model.
  - This is due to the degrees of freedom in the system.
  - There are n observations and $p + 1$ parameters (one regression coefficient for each predictor plus the intercept) leaving $n - p - 1$ degrees of freedom.
  - Increasing the sample size provides more information about the population test.
  - Increasing the number of predictors in the resulting model lowers the degrees of freedom available to estimate the variability of the predictors.

LASSO, Ridge, and Elastic Net Regularized Regression

- Least Absolute Shrinkage and Selection Operator (LASSO): Tends to produce sparse (i.e. few predictors) whereby the algorithm selects an arbritary predictor among a set of correlated ones
- Ridge: Tends to select all correlated predictors with their coefficient values equal to each other
- Elastic Net: Blends the concepts of LASSO and Ridge regression to attempt to create a model that is both sparse, but also includes correlated predictors
- LASSO, Ridge, and Elastic Net regression are available from the `glmnet` R package
  - Sousa FG et al. DNA damage response alterations and its relation with drug activity across the NCI-60. DNA Repair (2015) for example usage of `glmnet` and Elastic Net

Clustering

- Goal: Divide data into groups (clusters), so that group members are more “similar” to each other than to members outside the group
- Clustering differs from classification in that in classification we have known groups

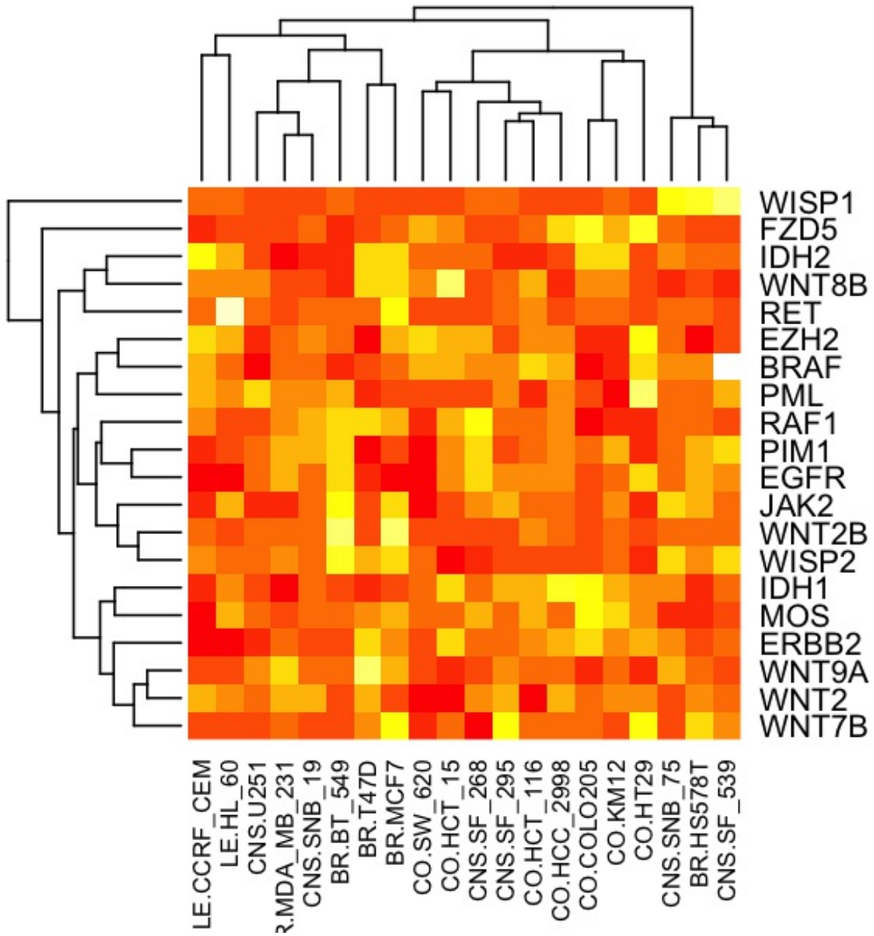
Hierarchical Clustering

- R uses an agglomerative (bottom-up) clustering approach
  - Alternative: Divise (top-down) is similiar to agglomerative, but in reverse
- Algorithm
  1. All points start in their own clusters
  2. At each iteration merge the 2 most similar structures
  3. Stop if there is a single cluster containing all points, else go to Step 2

Hierarchical Clustering Example

- Heatmap shows the expression of 20 oncogenes from 20 NCI-60 celllines

```
dat <- read.table("files/heatmapExample.txt", sep="\t", header=TRUE)
mat <- as.matrix(dat); heatmap(mat, cexCol=0.75)
```



Cluster Similarity (Linkage)

- Distances between clusters are calculated to determine cluster similarity
- Options of Cluster Linkage Distances
  - Single: Distance between two clusters is defined by the distance between the two closest points.
  - Average: Average of all pairwise distances between the points in two clusters
  - Complete: Distance between two clusters is defined by the distance between the two farthest points.
- `hclust()` used by `heatmap()` in R uses the “complete” method by default

K-Means Clustering

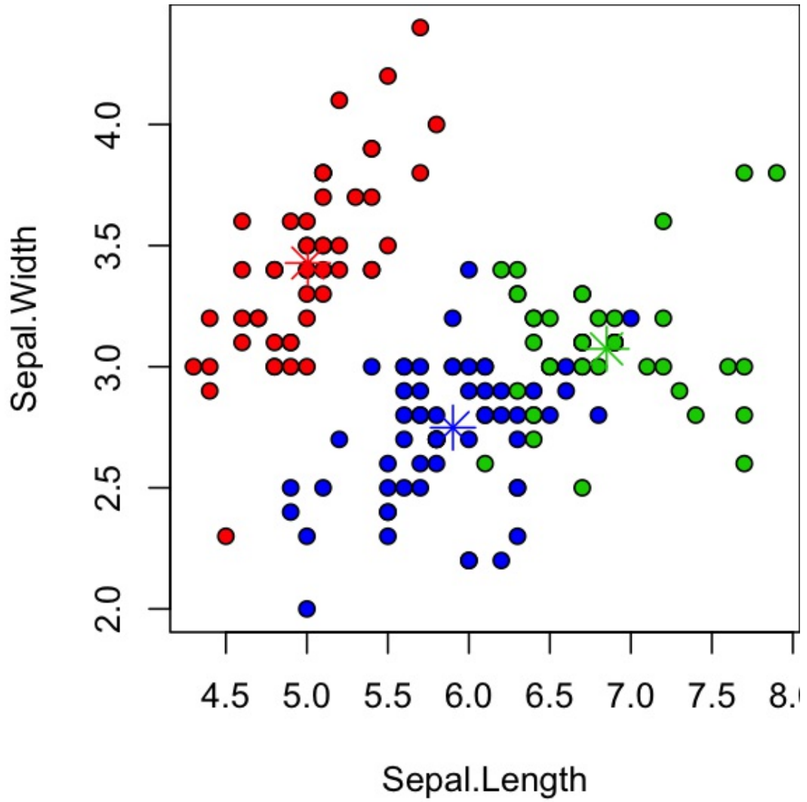
- Algorithm
  1. A user-selected (k) number of means are randomly generated from the data
  2. k clusters are created by grouping data points to the nearest mean.
  3. The centroid of the clusters becomes the new mean.
  4. Steps 2 and 3 are repeated until the clusters do not change anymore

K-Means Clustering Example

```r
# Retain only the numeric data in the iris dataset
iris_data <- iris[, 1:4]

kc <- kmeans(iris_data, 3)

par(mai=c(1,1,0,0))
plot(iris[c("Sepal.Length", "Sepal.Width")], bg=c("red","green3","blue")[kc$cluster],
pch=21)
points(kc$centers[,c("Sepal.Length", "Sepal.Width")], col=c("red","green3","blue"),
pch=8, cex=2)
```



Differences between Hierarchical and K-Means Clustering

- Partitions
  - K-means produces a single set of partitions
  - Hierarchical produces different partitions depending on where the tree is cut
- Cluster Number
  - K-means requires the number clusters to be set
  - Hierarchical clustering does not require the number of clusters to be set
- Speed
  - K-means is faster than hierarchical cluster

Dimensionality Reduction

- Goal: Seeks to reveal correlations that exist in data with many predictors
  - This reduces the dimensionality of the data by decreasing the redundancy of correlated variables
- Results:
  - Loadings: weight for the original variables to be multiplied by to get the component scores
  - Component Scores: Transformed variable values for a given point
- `prcomp` and `princomp` can do PCA in R; `prcomp` is the [[FIX]] advised function

What is Principal Component Analysis (PCA)

- PCA seeks to take a multi-dimensional (e.g. one with many predictors (features)) datasets
- Seeks to find a projection of the data that best separates data points
- Each “principal component” (PC) is an axis that captures the most variance
  - Variance is a measure of the spread of data points; standard deviation is the square root of variance
  - Each PC is a combination of the original variables scaled by a coefficient
  - Every PC explains some variance
  - Each additional PC explains less variance than the previous one

PCA Example

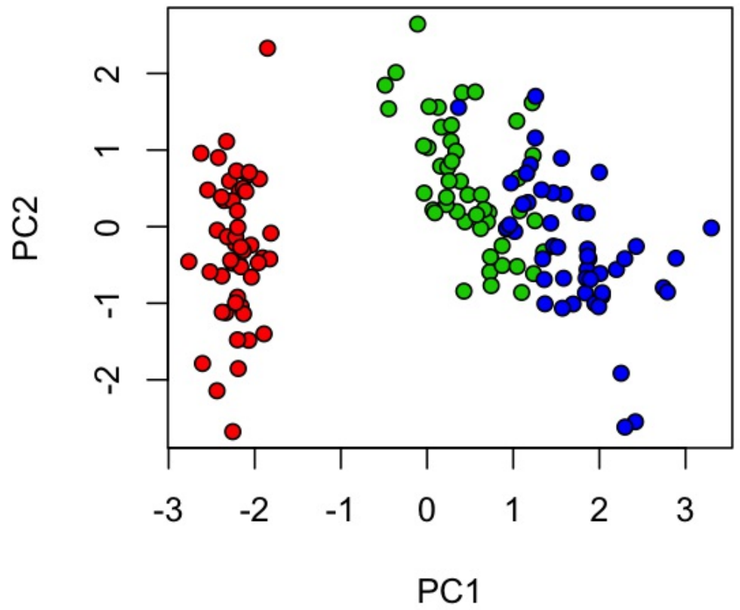
```r
iris_data <- iris[, 1:4]

# Variables are scaled to have a variance of 1; this is advisable
pcaResult <- prcomp(iris_data, scale=TRUE)
summary(pcaResult)
```

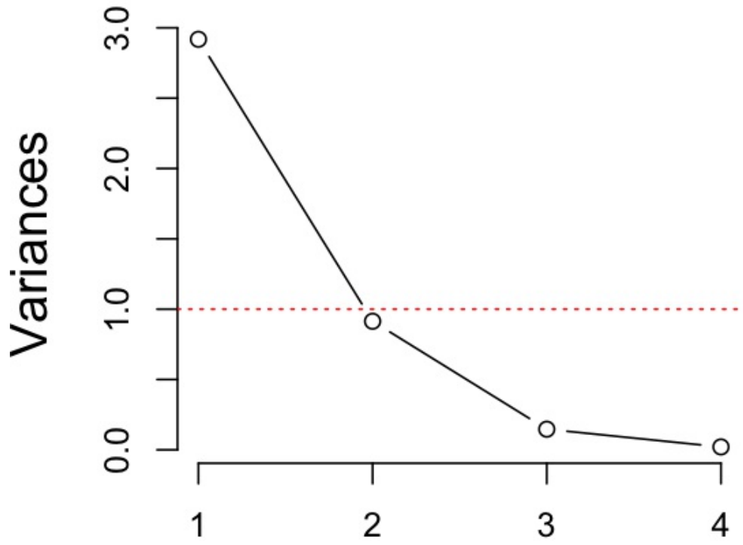
```
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

PCA Example Plots

```
# First 2 principal
components (PC)
plot(pcaResult$x, pch=21,
bg=c("red","green3","blue")
[unclass(iris$Species)])
```




```
# PC variances
plot(pcaResult,
type="line", cex.lab=1.5,
cex.main=1.5, main="")
abline(h=1, lty=3,
col="red")
```



Variable Combinations Yielding PCs

```
print(pcaResult)
```

```
Standard deviations:
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation:
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

How Many Principal Components Should be Kept?

- Kaiser criterion
  - Retain only principal components that with a variance greater than 1
  - Simple, but less advisable
- Scree Test
  - Find the place where the smooth decrease in the variances levels off
  - Multiple users may interpret the data different, unless trained the same

Getting Help

- Cross-Validated Stats Exchange
  - Part of Stack Overflow
  - http://stats.stackexchange.com/
- Biostars
  - https://www.biostars.org

Nature of PCs Using biplot

- PC1: Petal.Length and Petal.Width

```
par(mai=c(0.8,0.5,0.5,0))
biplot(pcaResult, scale=0, cex=.7)
```