SISG 2014

SISG Module 18: High-Dimensional Omics Data

19th Summer Institute in Statistical Genetics

UNIVERSITY of WASHINGTON

(This page left intentionally blank.)

HIGH-DIMENSIONAL OMICS DATA: Introduction

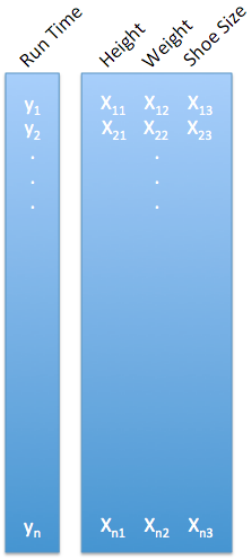
Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

A Simple Example

- Suppose we have $n = 500$ kids for whom we have $p = 3$ measurements: height, weight, and shoe size.
- We wish to predict these kids' 1600-meter run times using these measurements.

A Simple Example



Notation:

- n is the number of observations.
- p the number of variables/features/predictors.
- y is a n-vector containing response/outcome for each of n observations.
- X is a $n \times p$ data matrix.

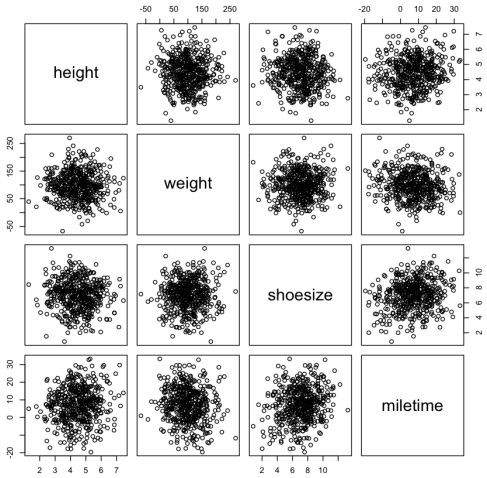
Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict run time using height, weight, and shoe size:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

  where y is run time, X_1, X_2, X_3 are height, weight, and shoe size, and ϵ is a noise term.
- You can look at the coefficients, p-values, and t-statistics for your linear regression model in order to interpret your results.
- You learned everything (or most of what) you need to analyze this data set in AP Statistics!

A Relationship Between the Variables?



Linear Model Output

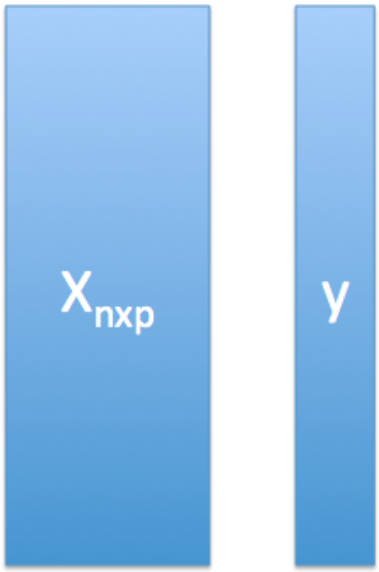
	Estimate	Std. Error	T-Stat	P-Value
Intercept	-2.265831	2.644654	-0.857	0.39199
height	1.074814	0.414789	2.591	0.00985 **
weight	-0.021155	0.008482	-2.494	0.01295 *
shoesize	0.955222	0.214449	4.454	1.04e-05 ***

$$\text{RunTime} \approx -2.27 + 1.07 \times \text{Height} - 0.021 \times \text{Weight} + 0.96 \times \text{ShoeSize}.$$

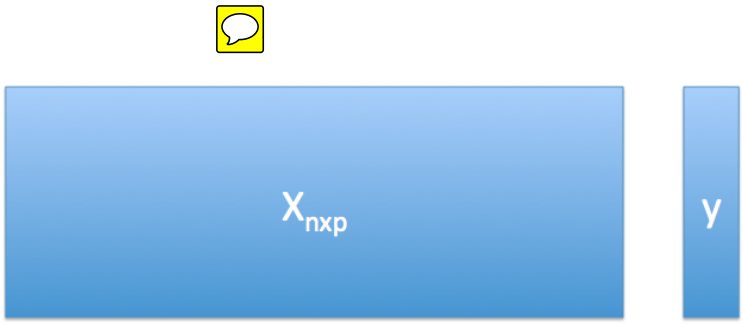
Low-Dimensional Versus High-Dimensional

- The data set that we just saw is low-dimensional: $n \gg p$.
- Lots of the data sets coming out of modern biological techniques are high-dimensional: $n \approx p$ or $n \ll p$.
- This poses statistical challenges! AP Statistics no longer applies.

Low Dimensional



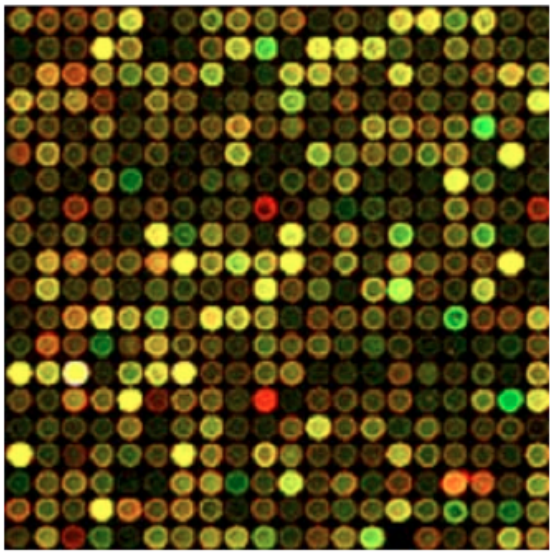
High Dimensional



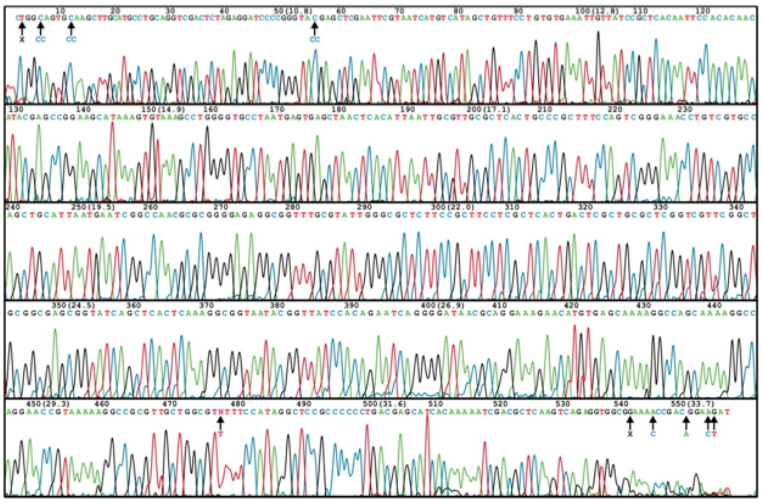
What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - 50-yard dash time
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.
- If we include too many predictors, we will overfit the data.
- Overfitting: Model looks great on the data used to develop it, but will perform very poorly on future observations.
- When $p \approx n$ or $p > n$, overfitting is guaranteed unless we are very careful.

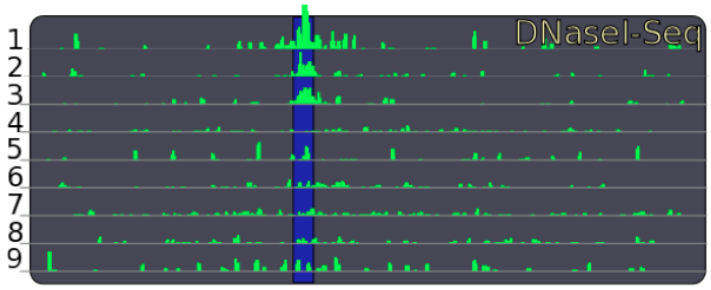
Gene Expression Data



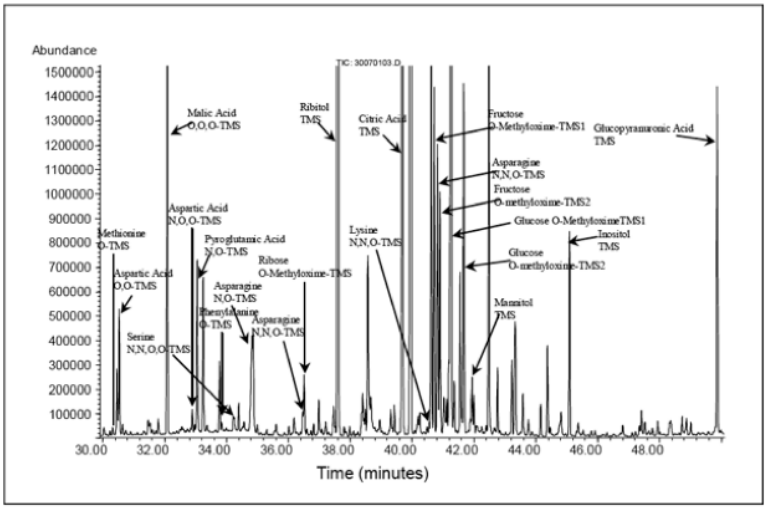
DNA Sequence Data



DNAse Hypersensitivity Data



Metabolomic Data



High-Dimensional Omics Analyses

For most omics analyses, we have many more variables than observations.... i.e. $p \gg n$.

- Predict risk of diabetes on the basis of DNA sequence data.... using $n = 1000$ patients and $p = 3000000$ variables.
- Cluster tissue samples on the basis of DNase hypersensitivity... using $n = 200$ cell types and $p = 1000000000$ variables.
- Identify genes whose expression is associated with survival time... using $n = 250$ cancer patients and $p = 20000$ variables.

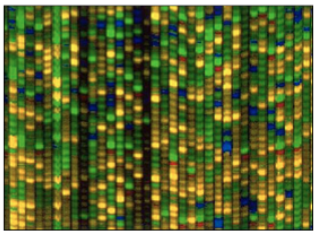
Why Does Dimensionality Matter?

- Classical statistical techniques, such as linear regression, *cannot* be applied.
- Even very simple tasks, like identifying variables that are associated with a response, must be done with care.
- High risks of overfitting, false positives, and more.

This course: Statistical machine learning tools for modeling high-dimensional omics data.

Statistical Machine Learning

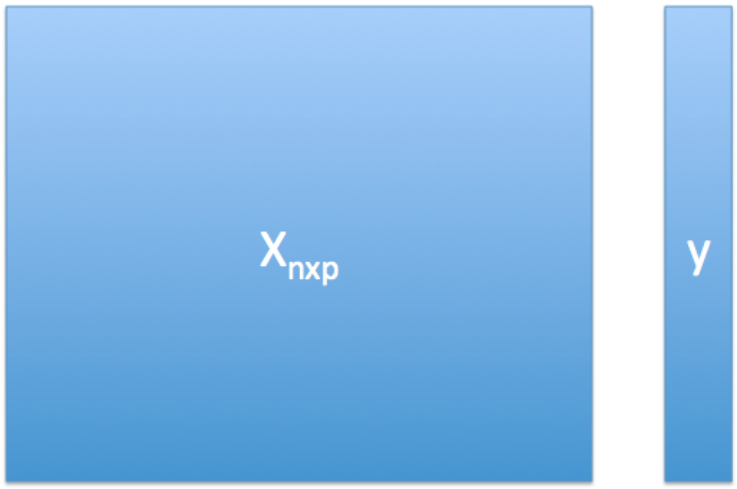
Google™



Supervised and Unsupervised Learning

- Statistical machine learning can be divided into two main areas: supervised and unsupervised.
- Supervised Learning: Use a data set X to predict or detect association with a response y.
  - Regression
  - Classification
  - Hypothesis Testing
- Unsupervised Learning: Discover the signal in X, or detect associations within X.
  - Dimension Reduction
  - Clustering

Supervised Learning



Unsupervised Learning

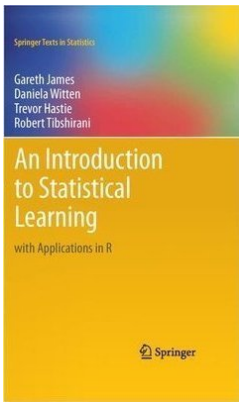
This Course

- Daniela will cover supervised learning in Lectures #1-4.
- Ali will cover unsupervised learning in Lectures #5-8.
- Ali will cover hypothesis testing in Lecture #9 – this is somewhere between supervised and unsupervised.

This Course

- We will cover the big ideas in statistical machine learning for high-dimensional omics data.
- The best way to use these methods: learn R.

"Course Textbook" . . . with applications in R



- Just published with Springer.
- Available for (free!) download from www.statlearning.com.
- An accessible introduction to statistical machine learning, with an R lab at the end of each chapter!!

Software Tutorial, TBD

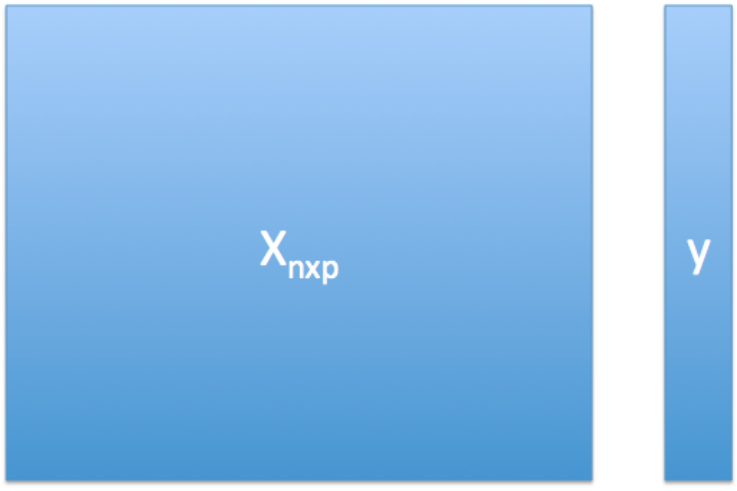
- Bring your laptop, with R installed.
- We will try out one of the labs in Introduction to Statistical Learning.
- To learn more... go through the labs on your own!

HIGH-DIMENSIONAL OMICS DATA: High-Dimensional Regression, Part I

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Supervised Learning



Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size
- Classification: Predict a categorical response, such as
  - tumor versus normal tissue
  - heart disease versus no heart disease
  - subtype of glioblastoma
- This lecture: Regression.

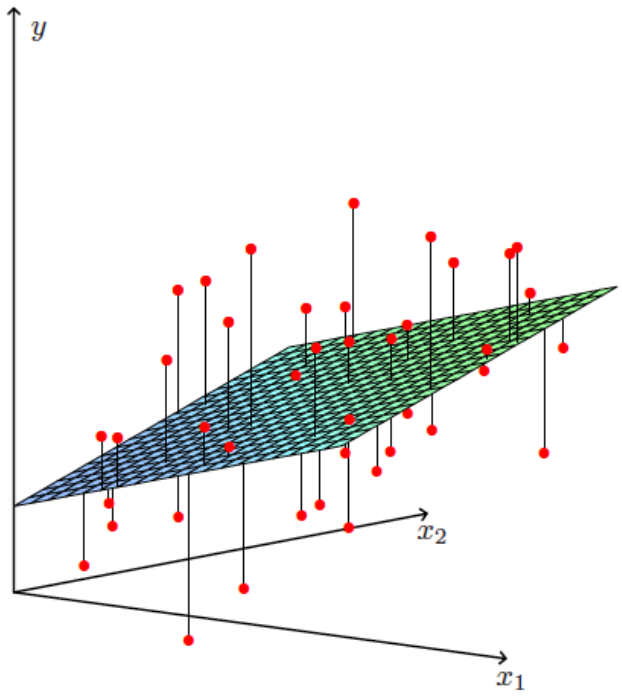
Linear Models

- We have n observations, for each of which we have p predictor measurements and a response measurement.
- Want to develop a model of the form

$$y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \epsilon_i.$$

- Here ϵ_i is a noise term associated with the ith observation.
- Must estimate $\beta_0, \beta_1, \ldots, \beta_p$ – i.e. we must fit the model.

Linear Model With $p = 2$ Predictors



What Makes a Model Linear?

- A linear model is linear in the regression coefficients!
- This is a linear model:

$$y_i = \beta_1 \sin(X_{i1}) + \beta_2 X_{i2} X_{i3} + \epsilon_i.$$

- This is not a linear model:

$$y_i = \beta_1^{X_{i1}} + \sin(\beta_2 X_{i2}) + \epsilon_i.$$

Linear Models in Matrix Form

- For simplicity, ignore the intercept β_0.
  - Assume $\sum_{i=1}^n y_i = \sum_{i=1}^n X_{ij} = 0$; in this case, $\beta_0 = 0$
  - Alternatively, let the first column of $\mathbf{X}$ be a column of 1's.
- In matrix form, we can write the linear model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

i.e.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}.$$

Least Squares Regression

- There are a lot of ways we could fit the model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

- Most common approach in classical statistics is least squares:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \right\}.$$

Here $||\mathbf{a}||^2 \equiv \sum_{i=1}^n a_i^2$.

- We are looking for $\beta_1, \dots, \beta_p$ such that

$$\sum_{i=1}^n (y_i - (\beta_1 X_{i1} + \dots + \beta_p X_{ip}))^2$$

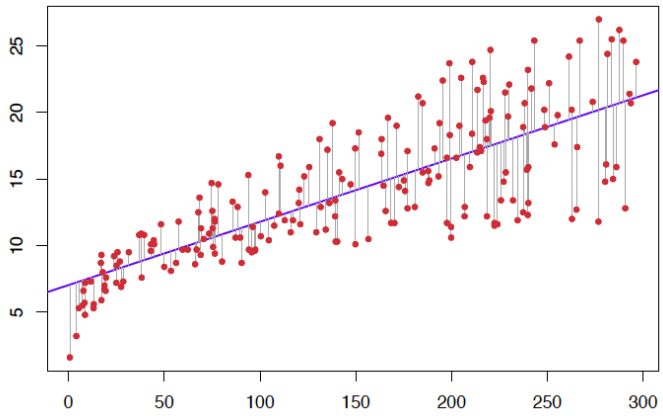
is as small as possible.

- Equivalently, we're looking for coefficient estimates such that

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

is as small as possible, where $\hat{y}_i$ is the ith predicted value.

Least Squares



- Horizontal axis: predictor
- Vertical axis: response
- Red dots: observations
- Blue line: least squares line

Blue line is positioned to minimize the sum of squared lengths of the gray lines.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \ldots + \hat{\beta}_p X_{ip}))^2.$$

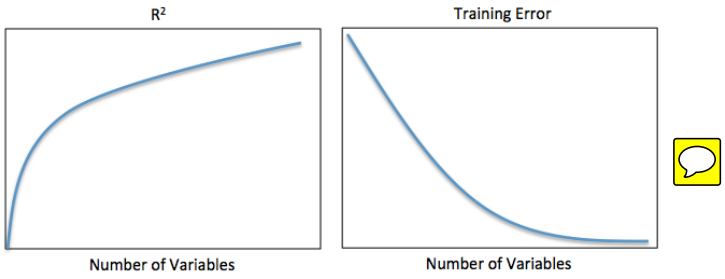
- The training error is closely related to the R^2 for a linear model – that is, the proportion of variance explained.
- Big $R^2 \Leftrightarrow$ Small Training Error.

Least Squares as More Variables are Included in the Model

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to use in developing a model, then I could surely come up with a regression model that fits the training data perfectly! Unfortunately, this model wouldn't capture the true signal in the data.
- We really care about the model's performance on test observations – observations not used to fit the model.

The Problem

As we add more variables into the model...



... the training error decreases and the R^2 increases!

Why is this a Problem?

- We really care about the model's performance on observations not used to fit the model!
  - We want a model that will predict the survival time of a new patient who walks into the clinic!
  - We want a model that can be used to diagnose cancer for a patient not used in model training!
  - We want to predict risk of diabetes for a patient who wasn't used to fit the model!
- What we really care about:

$$(y_{test} - \hat{y}_{test})^2,$$

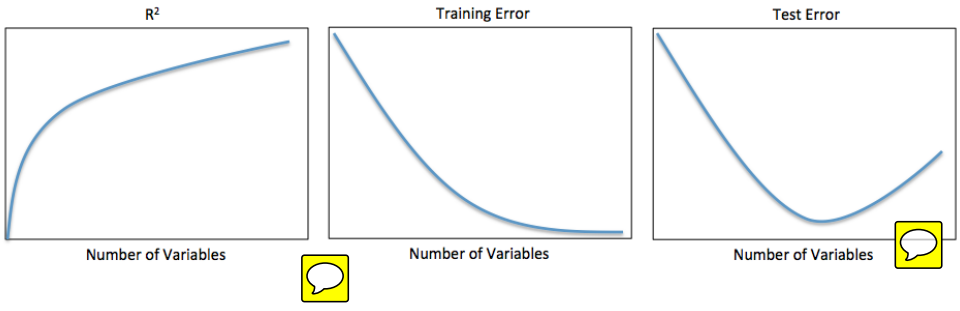
  where

$$\hat{y}_{test} = \hat{\beta}_1 X_{test,1} + \ldots + \hat{\beta}_p X_{test,p},$$

  and (X_{test}, y_{test}) was not used to train the model.
- The test error is the average of $(y_{test} - \hat{y}_{test})^2$ over a bunch of test observations.

Training Error versus Test Error

As we add more variables into the model...

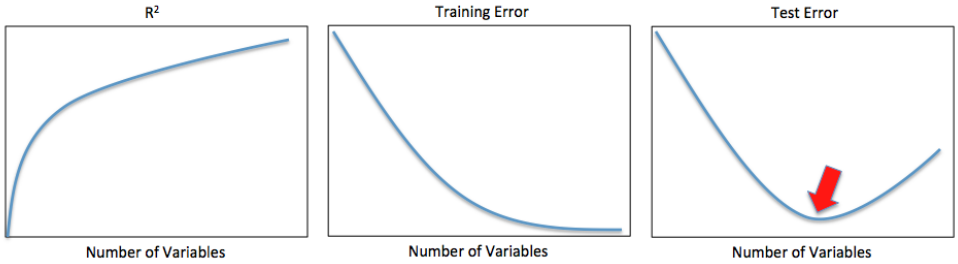


... the training error decreases and the R^2 increases!

But the test error might not!

Training Error versus Test Error

As we add more variables into the model...

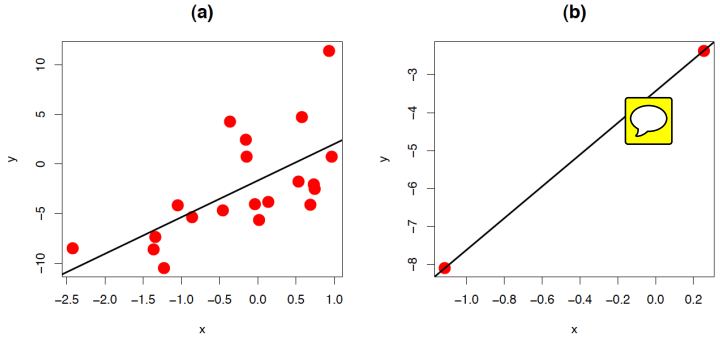


... the training error decreases and the R^2 increases!

But the test error might not!

Why the Number of Variables Matters

- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



- When $n \leq p$, you can always get a perfect model fit to the training data!
- But the test error will be awful.

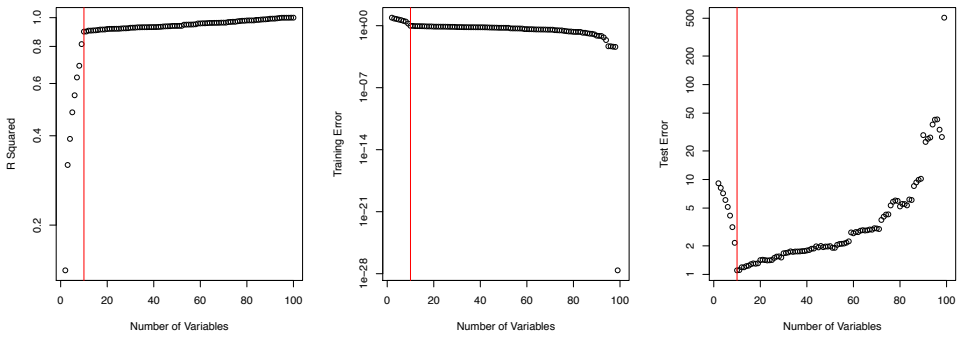
Model Complexity, Training Error, and Test Error

- In this course, we will consider various types of models.
- We will be very concerned with model complexity: e.g. the number of variables used to fit a model.
- As we fit more complex models – e.g. models with more variables – the training error will always decrease.
- But the test error might not.
- As we will see, the number of variables in the model is not the only – or even the best – way to quantify model complexity.

An Example In R

```
xtr <- matrix(rnorm(100*100),ncol=100)
xte <- matrix(rnorm(100000*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
yte <- xte%*%beta + rnorm(100000)
rsq <- trainerr <- testerr <-  NULL
for(i in 2:100){
mod <- lm(ytr~xtr[,1:i])
rsq <- c(rsq,summary(mod)$r.squared)
beta <- mod$coef[-1]
intercept <- mod$coef[1]
trainerr <- c(trainerr, mean((xtr[,1:i]%*%beta+intercept - ytr)^2))
testerr <- c(testerr, mean((xte[,1:i]%*%beta+intercept - yte)^2))
}
par(mfrow=c(1,3))
plot(2:100,rsq, xlab='Number of Variables', ylab="R Squared", log="y")
abline(v=10,col="red")
plot(2:100,trainerr, xlab='Number of Variables', ylab="Training Error",log="y")
abline(v=10,col="red")
plot(2:100,testerr, xlab='Number of Variables', ylab="Test Error",log="y")
abline(v=10,col="red")
```

Output of R Code



- 1st 10 variables are related to response; remaining 90 are not.
- R^2 increases and training error decreases as more variables are added to the model.
- Test error is lowest when only signal variables in model.

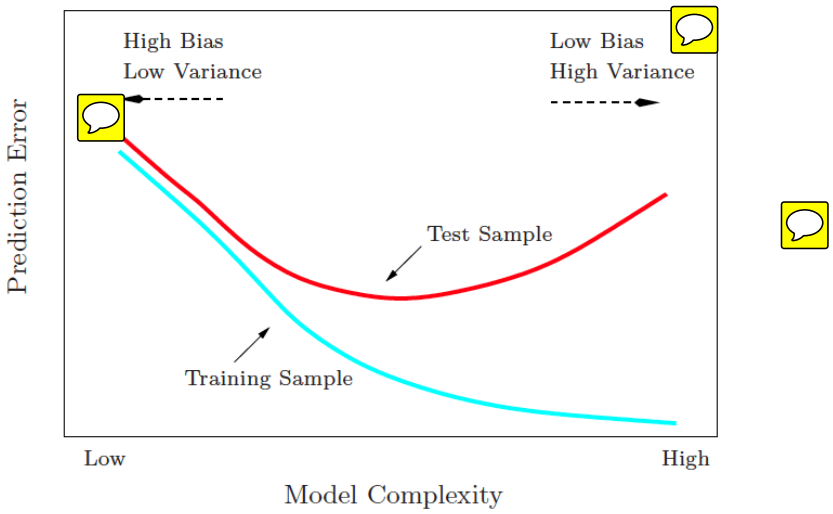
Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ – the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times – will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ – the amount by which the $\hat{\beta}$'s will differ across experiments – will increase.
- The test error depends on both the bias and variance:

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}.$$

- There is a bias-variance trade-off. We want a model that is sufficiently complex as to have not too much bias, but not so complex that it has too much variance.

A Really Fundamental Picture

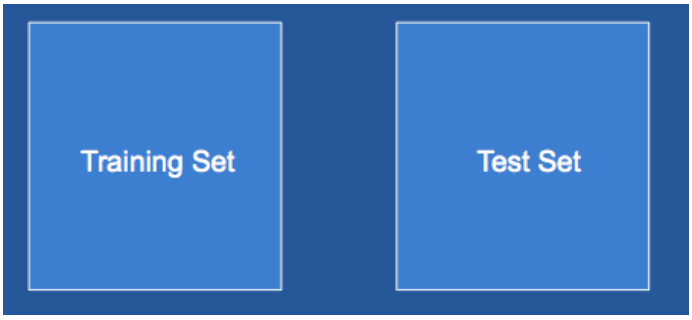


Overfitting

- Fitting an overly complex model – a model that has too much variance – is known as overfitting.
- In the omics setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.
- How can we estimate the test error?

Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.



Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?
A: No no no no no no.

Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.

You can't peek at the test set until you are completely done all aspects of model-fitting on the training set!

Training Set And Test Set

To get an estimate of the test error of a particular model on a future observation:

1. Split the samples into a training set and a test set.
2. Fit the model on the training set.
3. Evaluate its performance on the test set.
4. The test set error rate is an estimate of the model's performance on a future observation.

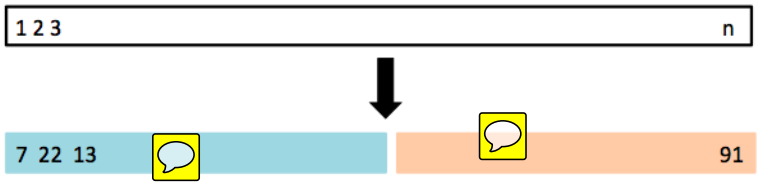
But remember: no peeking!

Choosing Between Several Models

- In general, we will consider a lot of possible models – e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.
- In what follows, assume that we have split the data into a training set and a test set, and the training set contains n observations.

Validation Set Approach

Split the n observations into two sets of approximately equal size. Train on one set, and evaluate performance on the other.



Validation Set Approach

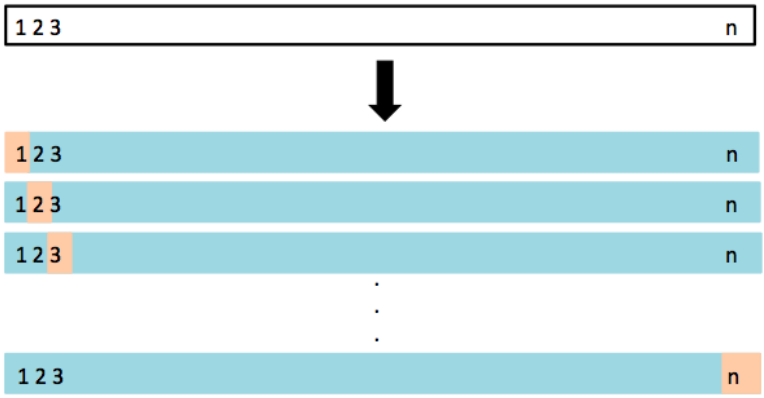
For a given model, we perform the following procedure:

1. Split the observations into two sets of approximately equal size, a training set and a validation set.
   a. Fit the model using the training observations. Let $\hat{\boldsymbol{\beta}}_{(train)}$ denote the regression coefficient estimates.
   b. For each observation in the validation set, compute the test error, $e_i = (y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_{(train)})^2$.
2. Calculate the total validation set error by summing the e_i's over all of the validation set observations.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

Leave-One-Out Cross-Validation

Fit n models, each on $n-1$ of the observations. Evaluate each model on the left-out observation.



Leave-One-Out Cross-Validation

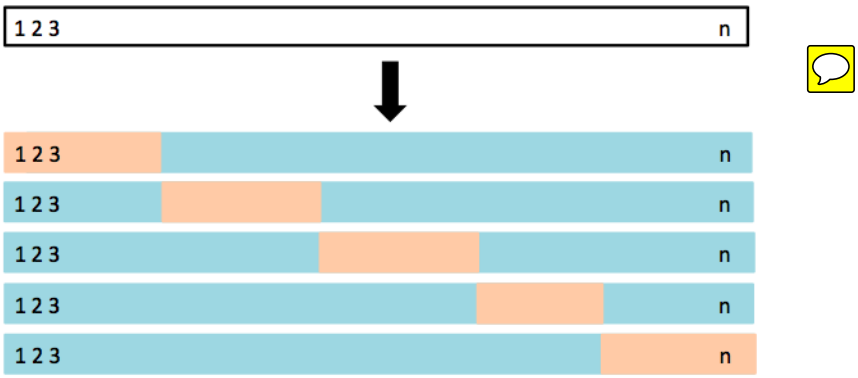
For a given model, we perform the following procedure:

1. For $i = 1, \ldots, n$:
   a. Fit the model using observations $1, \ldots, i-1, i+1, \ldots, n$. Let $\hat{\beta}_{(i)}$ denote the regression coefficient estimates.
   b. Compute the test error, $e_i = (y_i - \mathbf{x}_i^T \hat{\beta}_{(i)})^2$.
2. Calculate $\sum_{i=1}^{n} e_i$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

5-Fold Cross-Validation

Split the observations into 5 sets. Repeatedly train the model on 4 sets and evaluate its performance on the 5th.



K-fold cross-validation

A generalization of leave-one-out cross-validation. For a given model, we perform the following procedure:

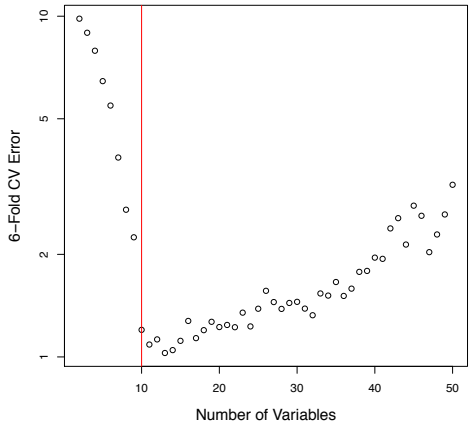
1. Split the n observations into K equally-sized folds.
2. For $k = 1, \ldots, K$:
   a. Fit the model using the observations not in the kth fold.
   b. Let e_k denote the test error for the observations in the kth fold.
3. Calculate $\sum_{k=1}^{K} e_k$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

An Example In R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
cv.err <- NULL
for(i in 2:50){
dat <- data.frame(x=xtr[,1:i],y=ytr)
mod <- glm(y~.,data=dat)
cv.err <- c(cv.err, cv.glm(dat,mod,K=6)$delta[1]
}
plot(2:50, cv.err, xlab="Number of Variables",
ylab="6-Fold CV Error", log="y")
abline(v=10, col="red")
```


Output of R Code



- Six-fold CV identifies the model with just over ten predictors.
- First ten predictors contain signal, and the rest are noise.

After Estimating the Test Error on the Training Set...

After we estimate the test error using the training set, we refit the "best" model on all of the available training observations. We then evaluate this model on the test set.

Big Picture

Big Picture

Big Picture

Big Picture

Big Picture

Summary: Four-Step Procedure

1. Split observations into training set and test set.
2. Fit a bunch of models on training set, and estimate the test error, using cross-validation or validation set approach.
3. Refit the best model on the full training set.
4. Evaluate the model's performance on the test set.

Why All the Bother?

Q: Why do I need to have a separate test set? Why can't I just estimate test error using cross-validation or a validation set approach using all the observations, and then be done with it?

A: In general, we are choosing between a whole bunch of models, and we will use the cross-validation or validation set approach to pick between these models. If we use the resulting estimate as a final estimate of test error, then this could be an extreme underestimate, because one model might give a lower estimated test error than others by chance. To avoid having an extreme underestimate of test error, we need to evaluate the "best" model obtained on an independent test set. *This is particularly important in high dimensions!!*

Regression in the Omics Setting

- We usually cannot perform least squares regression to fit a model in the omics setting, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.
- We will consider five ways to fit less complex models:
  1. Variable Pre-Selection
  2. Forward Stepwise Regression
  3. Ridge Regression
  4. Lasso Regression
  5. Principal Components Regression

- These are alternatives to fitting a linear model using least squares.
- Each of these approaches will allow us to choose the level of complexity – e.g. the number of variables in the model.

The Fundamental Truth About High-Dimensional Data

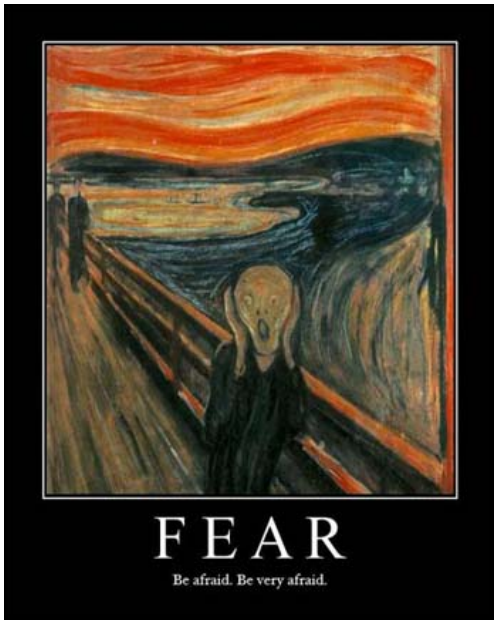
If you

- fit your model carelessly;
- do not properly estimate the test error;
- or select a model based on training set rather than test set performance;

then you will woefully overfit your training data, leading to a model that looks good on training data but will perform atrociously on future observations.

Our intuition breaks down in high dimensions, and so rigorous model-fitting is crucial.

The Curse of Dimensionality



The Curse of Dimensionality

Q: A data set with more variables is better than a data set with fewer variables, right?

A: Not necessarily!

Noise variables – such as genes whose expression levels are not truly associated with the response being studied – will simply increase the risk of overfitting, and the difficulty of developing an effective model that will perform well on future observations.

On the other hand, more signal variables – variables that are truly associated with the response being studied – are always useful!

Every Biostatisticians' Favorite Anecdote

A biostatistician walks into a collaborator's office with a list of genes found to be predictive of survival time in a condition of interest....

Wise Words

In high-dimensional data analysis, common mistakes are simple, and simple mistakes are common.

– Keith Baggerly

Before You're Done Your Analysis

- Estimate the test error.
- Do a "sanity check" whenever possible.
  - "Spot-check" the variables that have the largest coefficients in the model.
  - Rewrite your code from scratch. Do you get the same answer again?

Fitting models in high-dimensions: one mistake away from disaster!

HIGH-DIMENSIONAL OMICS DATA: High-Dimensional Regression, Part II

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Linear Models in High Dimensions

- When p is large, least squares regression will lead to very low training error but terrible test error.
- We will now see some approaches for fitting linear models in high dimensions.

Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes measured using a microarray.
  - protein levels.
  - mutations in genes potentially implicated in breast cancer.
- How can we develop a model with low test error in this setting?

Remember

- Before we begin any sort of model-fitting procedure, we split our observations into a training set and a test set.
- The test observations go in a locked box, not to be peeked at.
- We fit a bunch of models, and estimate the test error using e.g. cross-validation, using the training observations only.
- From here on out, we are describing what to do with the training set. Test observations are in the locked box, right where we left them!
- Assume we have n training observations.

Variable Pre-Selection

The simplest approach for fitting a model in high dimensions:

1. Choose a small set of variables, say the q variables that are most correlated with the response, where $q < n$ and $q < p$.
2. Use least squares to fit a model predicting y using only these q variables.

This approach is simple and straightforward.

Variable Pre-Selection in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
cors <- cor(xtr,ytr)
whichers <- which(abs(cors)>.2)
mod <- lm(ytr~xtr[,whichers])
print(summary(mod))
```

How Many Variable to Use?

- We need a way to choose q, the number of variables used in the regression model.
- We want q that minimizes the test error.
- For a range of values of q, we can perform the validation set approach, leave-one-out cross-validation, or K-fold cross-validation in order to estimate the test error.
- Then choose the value of q for which the estimated test error is smallest.

Estimating the Test Error For a Given q

This is the right way to estimate the test error using the validation set approach:

1. Split the observations into a training set and a validation set.
2. Using the training set only:
   a. Identify the q variables most associated with the response.
   b. Use least squares to fit a model predicting y using those q variables.
   c. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
3. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Estimating the Test Error For a Given q

This is the wrong way to estimate the test error using the validation set approach: 

1. Identify the q variables most associated with the response on the full data set.
2. Split the observations into a training set and a validation set.
3. Using the training set only:
   a. Use least squares to fit a model predicting y using those q variables.
   b. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
4. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Frequently Asked Questions

- **Q:** Does it really matter how you estimate the test error?
  A: Yes.
- **Q:** Would anyone make such a silly mistake?
  A: Yes.

A Better Approach

- The variable pre-selection approach is simple and easy to implement – all you need is a way to calculate correlations, and software to fit a linear model using least squares.
- But it might not work well: just because a bunch of variables are correlated with the response doesn't mean that when used together in a linear model, they will predict the response well.
- What we really want to do: pick the q variables that best predict the response.

Best Subset Selection

- We would like to consider all possible models using a subset of the p predictors.
- In other words, we'd like to consider all 2^p possible models.
- This is called best subset selection.
- Unfortunately, this is computationally intractable:
  - When $p = 3$, $2^p = 8$.
  - When $p = 6$, $2^p = 64$.
  - When $p = 250$, there are $2^{250} \approx 10^{80}$ possible models. According to www.universetoday.com, this is around the number of atoms in the known universe.
  - Not feasible to consider so many models!
- Need an efficient way to sift through all of these models: forward stepwise regression.

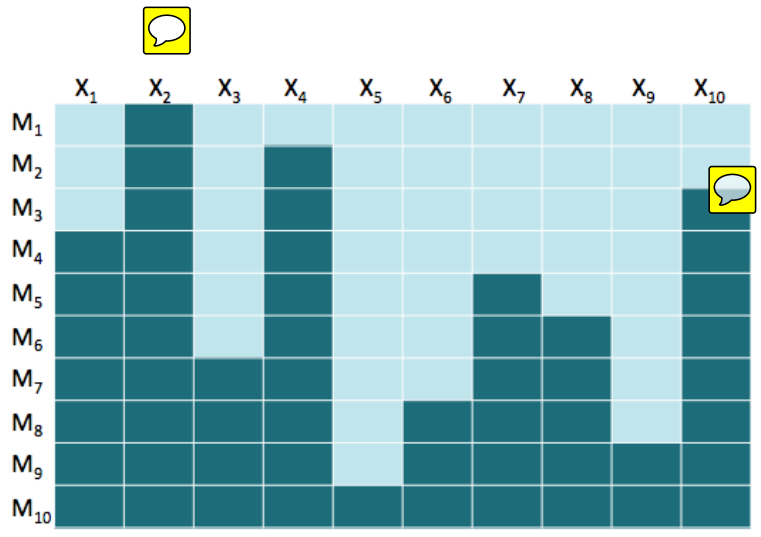
Forward Stepwise Regression

1. Use least squares to fit p univariate regression models, and select the predictor corresponding to the best model (according to e.g. training set MSE).
2. Use least squares to fit $p-1$ models containing that one predictor, and each of the $p-1$ other predictors. Select the predictors in the best two-variable model.
3. Now use least squares to fit $p-2$ models containing those two predictors, and each of the $p-2$ other predictors. Select the predictors in the best three-variable model.
4. And so on....

This gives us a nested set of models, containing the predictors

$$\mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \mathcal{M}_3 \subseteq \ldots.$$

Forward Stepwise Regression With $p = 10$



Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(leaps)
out <- regsubsets(xtr,ytr,nvmax=30,method="forward")
print(summary(out))
print(coef(out,1:10))
```


Which Value of q is Best?

- This procedure traces out a set of models, containing between 1 and p variables.
- The qth model contains q variables, given by the set $\mathcal{M}_q$.
- **Q:** Which value of q is best?
  A: *The one that minimizes the test error!*
- We can select the value of q using cross-validation or the validation set approach.

Drawback of Forward Stepwise Selection

- Forward stepwise selection isn't guaranteed to give you the best model containing q variables. 
- To get the best model with q variables, you'd need to consider every possible one; computationally intractable.
- For instance, suppose that the best model with one variable is

$$y = \beta_3 X_3 + \epsilon$$

and the best model with two variables is

$$y = \beta_4 X_4 + \beta_8 X_8 + \epsilon.$$

Then forward stepwise selection will not identify the best two-variable model.

- **Q:** Does this really happen in practice?
  A: Yes.

How To Do Forward Stepwise?

Wrong: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the model with best performance on the validation set. Then, refit the model (using those q variables) on the full data set.

Right: Split the data into a training set and a validation set. Perform forward stepwise on the training set, and identify the value of q corresponding to the best-performing model on the validation set. Then, perform forward stepwise selection in order to obtain a q-variable model on the full data set. 

Bottom Line: We estimate the test error in order to choose the correct level of **model complexity**. Then we refit the model on the full data set.

Ridge Regression and the Lasso

- Forward stepwise selection does a discrete search through model space, considering subsets of the predictors, and fitting each of the resulting models using least squares. Model complexity is controlled by using subsets of the predictors.
- Ridge regression and the lasso instead control model complexity by using an alternative to least squares, by shrinking the regression coefficients.
- This is known as regularization or penalization.
- Hot area in statistical machine learning today.

Crazy Coefficients

- When $p > n$, the least squares regression coefficients are highly variable because some of the variables are highly correlated with each other.
- Why does correlation matter?
  - Suppose that X_1 and X_2 are highly correlated with each other... assume $X_1 = X_2$ for the sake of argument.
  - And suppose that the least squares model is

$$\hat{y} = X_1 - 2X_2 + 3X_3.$$

  - Then this is also a least squares model:

$$\hat{y} = 100000001X_1 - 100000002X_2 + 3X_3.$$

- Bottom Line: When there are too many variables, the least squares coefficients can get crazy!
- This craziness is directly responsible for poor test error.
- It amounts to too much model complexity.

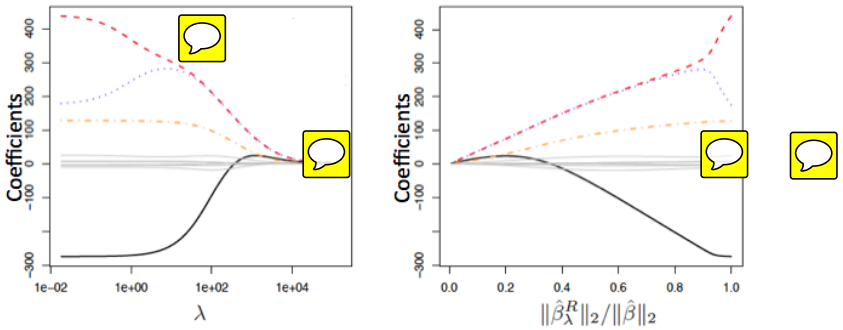
A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares involves finding $\boldsymbol{\beta}$ that minimizes
$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$
- Ridge regression involves finding $\boldsymbol{\beta}$ that minimizes
$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j \beta_j^2.$$
- Equivalently, find $\boldsymbol{\beta}$ that minimizes
$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$
subject to the constraint that
$$\sum_{j=1}^{p} \beta_j^2 \leq s.$$

Ridge Regression

- Ridge regression coefficient estimates minimize
$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j \beta_j^2.$$
- Here λ is a nonnegative tuning parameter that shrinks the coefficient estimates.
- When $\lambda = 0$, then ridge regression is just the same as least squares.
- As λ increases, then $\sum_{j=1}^{p} (\hat{\beta}_{\lambda,j}^R)^2$ decreases – i.e. coefficients become shrunken towards zero.
- When $\lambda = \infty$, $\hat{\boldsymbol{\beta}}_\lambda^R = 0$

Ridge Regression As λ Varies



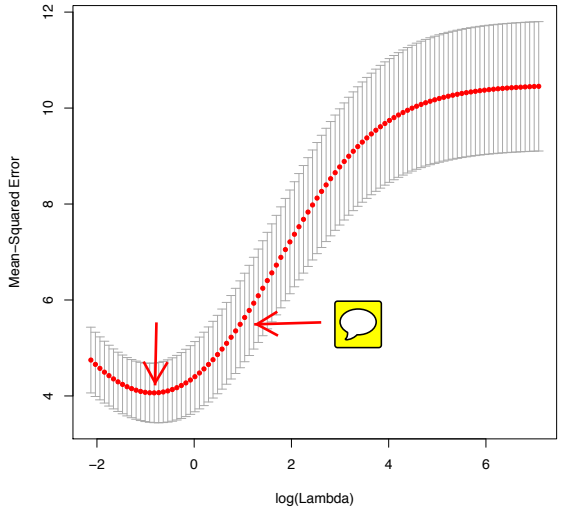
Ridge Regression In Practice

- Perform ridge regression for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform ridge on the full data set, using that value of λ.

Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=0,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",
sum(abs(coef(cv.out))<1e-8),fill=TRUE)
```


R Output



Drawbacks of Ridge

- Ridge regression is a simple idea and has a number of attractive properties: for instance, you can continuously control model complexity through the tuning parameter λ.
- But it suffers in terms of model interpretability, since the final model contains all p variables, no matter what.
- In the analysis of omics data, we often want a simpler model that is defined in terms of a subset of the features.
- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.
- In other words, the resulting model is sparse. We say that the lasso performs feature selection.
- The lasso is a very active area of research interest in the statistical community!

The Lasso

- The lasso involves finding $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_j |\beta_j|.$$

- Equivalently, find $\boldsymbol{\beta}$ that minimizes

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

subject to the constraint that

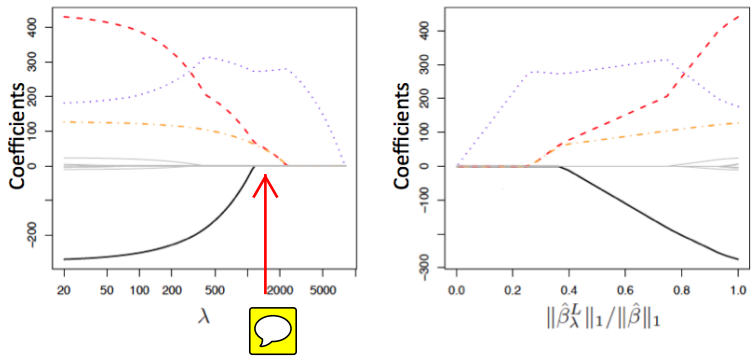
$$\sum_{j=1}^{p} |\beta_j| \leq s.$$

- So lasso is just like ridge, except that β_j^2 has been replaced with $|\beta_j|$.

The Lasso

- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.
  - When λ is very large, we get $\hat{\beta}^L_\lambda = 0$.
- But unlike ridge, lasso will give some coefficients exactly equal to zero for intermediate values of λ!

Lasso As λ Varies



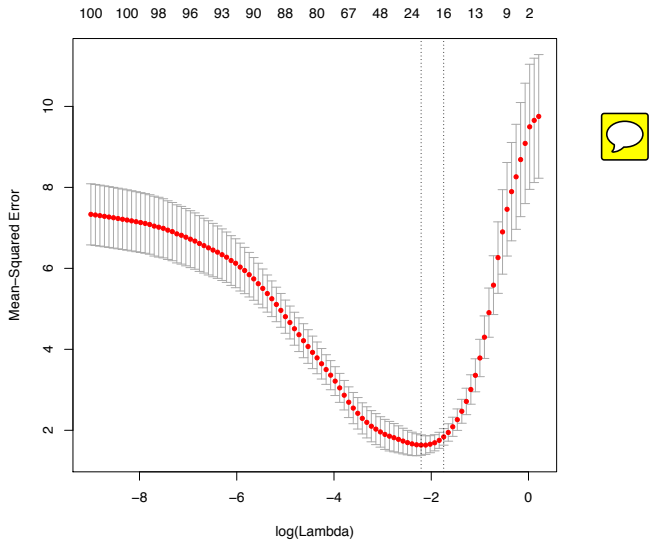
Lasso In Practice

- Perform lasso for a very fine grid of λ values.
- Use cross-validation or the validation set approach to select the optimal value of λ – that is, the best level of model complexity.
- Perform the lasso on the full data set, using that value of λ.

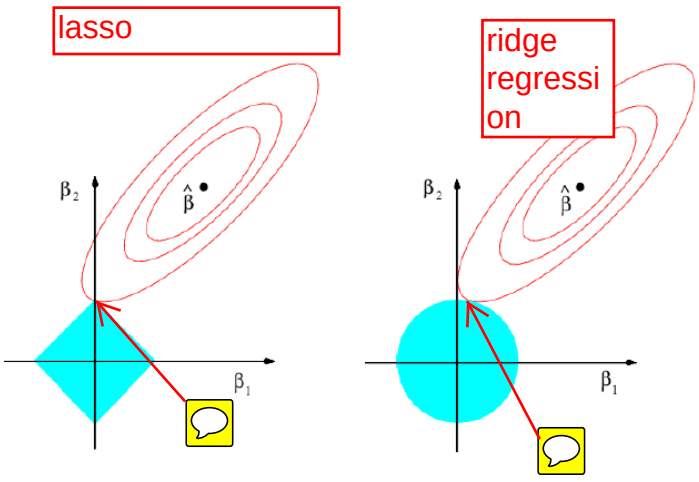
Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(glmnet)
cv.out <- cv.glmnet(xtr,ytr,alpha=1,nfolds=5)
print(cv.out$cvm)
plot(cv.out)
cat("CV Errors", cv.out$cvm,fill=TRUE)
cat("Lambda with smallest CV Error",
cv.out$lambda[which.min(cv.out$cvm)],fill=TRUE)
cat("Coefficients", as.numeric(coef(cv.out)),fill=TRUE)
cat("Number of Zero Coefficients",sum(abs(coef(cv.out))<1e-8),
fill=TRUE)
```

R Output



Ridge and Lasso: A Geometric Interpretation



Review

- So far we have seen two approaches that select subsets of the features and fit a least squares model:
  - Variable Pre-Selection
  - Forward Stepwise Selection
- And we have seen two approaches that fit a shrunken model instead of using least squares:
  - Ridge regression
  - Lasso
- Now we see one final approach, principal components regression, that first finds a low-dimensional subspace of the data and then fits a model on that low-dimensional subspace, using least squares.

Principal Components Regression

- Our data consist of n observations in a p-dimensional space.
- However, not all of those p dimensions are equally useful, especially when $p \gg n$.
- Many are either completely redundant (correlated features) or uninformative (noise features).
- Can we find a low-dimensional representation of the variables that captures most of the variability in the data?
- This is a dimension reduction approach.

PCR

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ linear combinations of the p predictors:

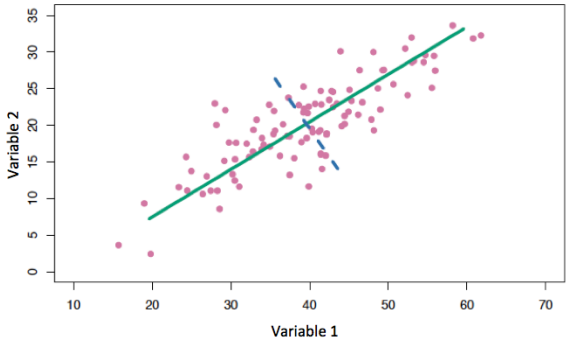
$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j.$$

- Use least squares to fit the model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, \ldots, n.$$

- In other words, we perform least squares using M new predictors, $Z_1, \ldots, Z_M$.
- $Z_1, \ldots, Z_M$ chosen to be the principal components of the data

Principal Components, Conceptually



- PCs are the linear combinations of the variables that contain as much as possible of the variability in the data.
- We will discuss this in more detail in a later lecture in the context of unsupervised learning.

PCR

Our final model is linear in the original predictors:

$$\begin{aligned} y_i &= \theta_0 + \sum_{m=1}^{M} \theta_m Z_{im} + \epsilon_i \\ &= \theta_0 + \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{mj} X_{ij} + \varepsilon_i \\ &= \theta_0 + \sum_{j=1}^{p} \left(\sum_{m=1}^{M} \theta_m \phi_{mj} \right) X_{ij} + \varepsilon_i \end{aligned}$$

More on PCR

- PCR doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCR can avoid overfitting and can give good results.
- Choose M by cross-validation or validation set approach.
- With $M = p$, we just get least squares regression: no dimension reduction occurs!
- Turns out that PCR is closely related to ridge regression.
- Shortcoming of PCR: the first M principal components are guaranteed to explain a lot of the variation in the data, but that doesn't mean that they are predictive of the response!
- Later in this course, we will see how principal components can be used for unsupervised learning.

Example in R

```
xtr <- matrix(rnorm(100*100),ncol=100)
beta <- c(rep(1,10),rep(0,90))
ytr <- xtr%*%beta + rnorm(100)
library(pls)
out <- pcr(ytr~xtr,scale=TRUE,validation="CV")
summary(out)
validationplot(out,val.type="MSEP")
```


Pros/Cons of Each Approach

Approach	Simplicity?*	Sparsity?**	Predictions?***
Pre-Selection	Good	Yes	So-So
Forward Stepwise	Good	Yes	So-So
Ridge	Medium	No	Great
Lasso	Bad	Yes	Great
PCR	Medium	No	Great

* How simple is this model-fitting procedure? If you were stranded on a desert island with pretty limited statistical software, could you fit this model?

** Does this approach perform feature selection, i.e. is the resulting model sparse?

*** How good are the predictions resulting from this model?

No "Best" Approach

- There is no "best" approach to regression in high dimensions.
- Some approaches will work better than others. For instance:
  - Lasso will work well if it's really true that just a few features are associated with the response.
  - Ridge will do better if all of the features are associated with the response.

- If somebody tells you that one approach is "best"... then they are mistaken. Politely contradict them.
- While no approach is "best", some approaches are wrong (e.g.: there is a wrong way to do cross-validation)!

Predicting Age Using DNA Methylation Data

- Comparison on 6 data sets
- SPC: Like principal components regression, but using a subset of features most associated with response. Between 1 and 3 principal components were used.
- Elastic Net: A hybrid between ridge and lasso.
- SVM: We'll see it next lecture in the classification context.
- Citation: Zhuang et al., BMC Bioinformatics, 2012

Didn't I Tell You? No Best Method!

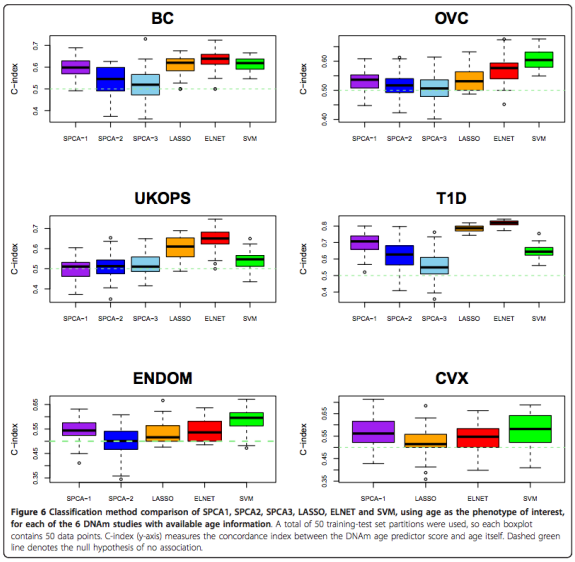


Figure 6 Classification method comparison of SPCA1, SPCA2, SPCA3, LASSO, ELNET and SVM, using age as the phenotype of interest, for each of the 6 DNAm studies with available age information. A total of 50 training-test set partitions were used, so each boxplot contains 50 data points. C-index (y-axis) measures the concordance index between the DNAm age predictor score and age itself. Dashed green line denotes the null hypothesis of no association.

High C-index indicates a low test error.

Bottom Line

Much more important than what model you fit is how you fit it.

- Was cross-validation performed properly?
- Did you keep your test observations in a locked box, or did you peek at them too early?

HIGH-DIMENSIONAL OMICS DATA: High-Dimensional Classification

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal 
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Classification problems tend to occur even more frequently than regression problems in the analysis of omics data.
- Just like regression,
  - Classification cannot be blindly performed in high-dimensions because you will get zero training error but awful test error;
  - Properly estimating the test error is crucial; and
  - There are a few tricks to extend classical classification approaches to high-dimensions, which we have already seen in the regression context!

Classification

- There are many approaches out there for performing classification.
- We will discuss two, logistic regression and support vector machines.

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{0, 1\}$: a two-class classification problem.
- The simple linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  doesn't make sense for classification.
- Instead, the logistic regression model is

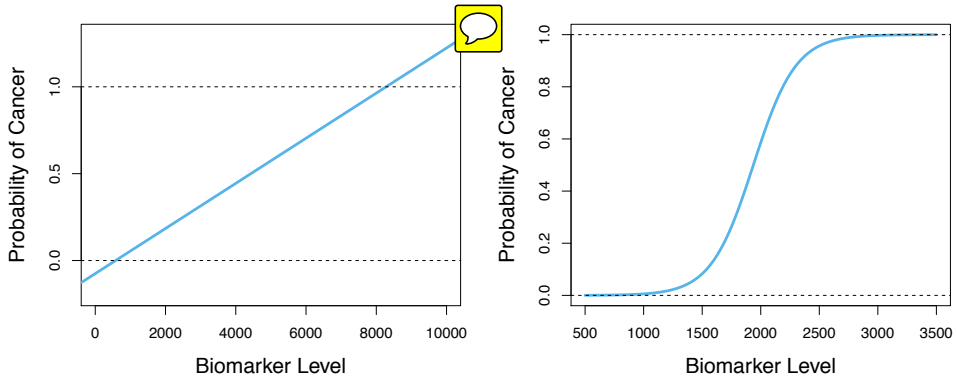
$$P(y = 1|X) = \frac{\exp(X^T\beta)}{1 + \exp(X^T\beta)}.$$

- We usually fit this model using maximum likelihood – like least squares, but for logistic regression.

Example in R

```
xtr <- matrix(rnorm(1000*20),ncol=20)
beta <- c(rep(1,10),rep(0,10))
ytr <- 1*((xtr%*%beta + .2*rnorm(1000)) >= 0)
mod <- glm(ytr~xtr,family="binomial")
print(summary(mod))
```


Why Not Linear Regression?



- Left: linear regression.
- Right: logistic regression.

Five Ways to Extend Logistic to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression
3. Ridge Logistic Regression
4. Lasso Logistic Regression
5. Principal Components Logistic Regression

How to decide which approach is best, and which tuning parameter value to use for each approach? Cross-validation or validation set approach.

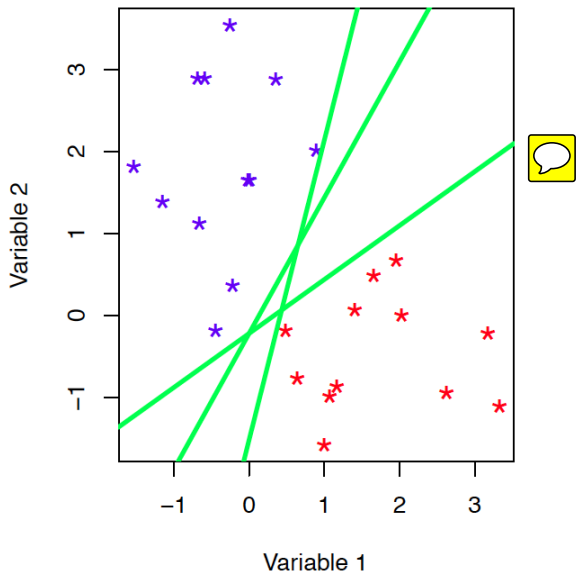
Example in R: Lasso Logistic Regression

```
xtr <- matrix(rnorm(1000*20),ncol=20)
beta <- c(rep(1,5),rep(0,15))
ytr <- 1*((xtr%*%beta + .5*rnorm(1000)) >= 0)
cv.out <- cv.glmnet(xtr, ytr, family="binomial", alpha=1)
plot(cv.out)
```

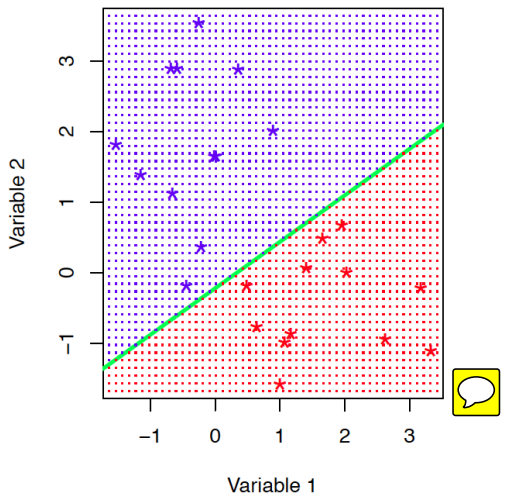
Support Vector Machines

- Developed in around 1995.
- Touted as "overcoming the curse of dimensionality."
- Does not overcome the curse of dimensionality!!!
- Fundamentally and numerically very similar to logistic regression.
- But, it is a nice idea.

Separating Hyperplane

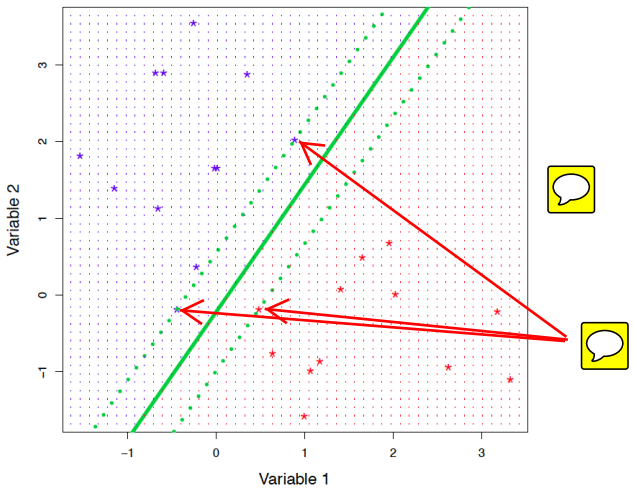


Classification Via a Separating Hyperplane



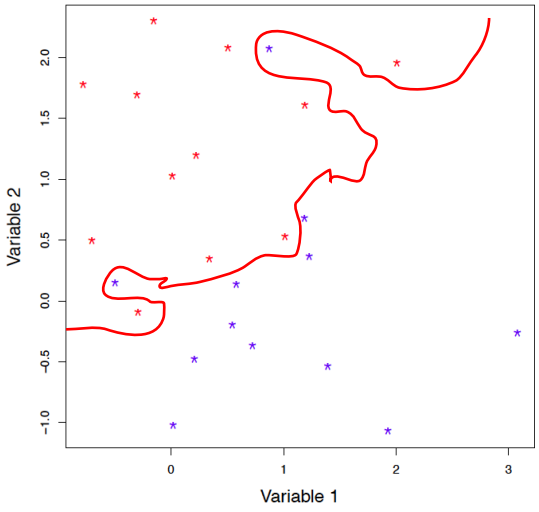
Blue class if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > c$; red class otherwise.

Maximal Separating Hyperplane

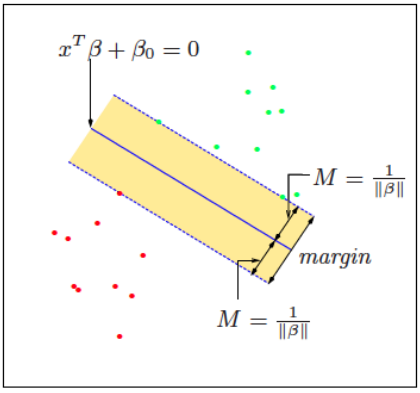


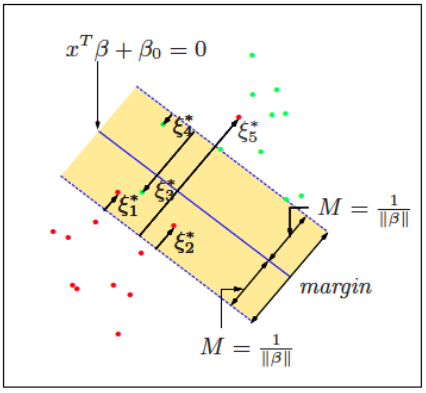
Note that only a few observations are on the margin: these are the support vectors.

What if There is No Separating Hyperplane?



Support Vector Classifier: Allow for Violations

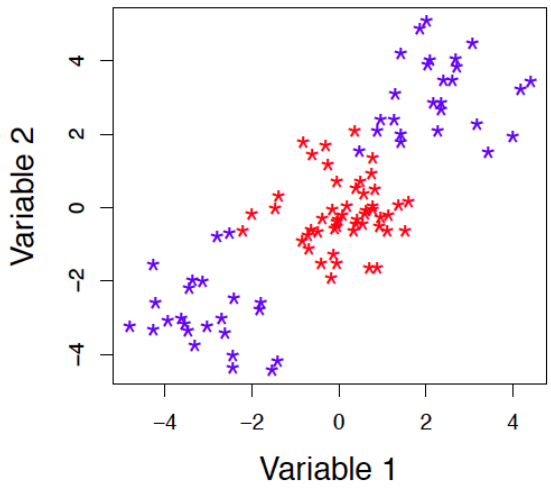




Support Vector Machine

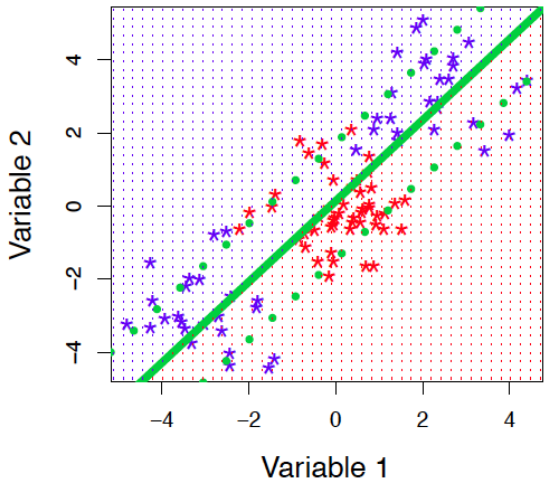
- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: "non-linear kernels".
- However, linear regression, logistic regression, and other classical statistical approaches can also be applied to non-linear functions of the variables.
- For historical reasons, SVMs are more frequently used with non-linear expansions as compared to other statistical approaches.

Non-Linear Class Structure



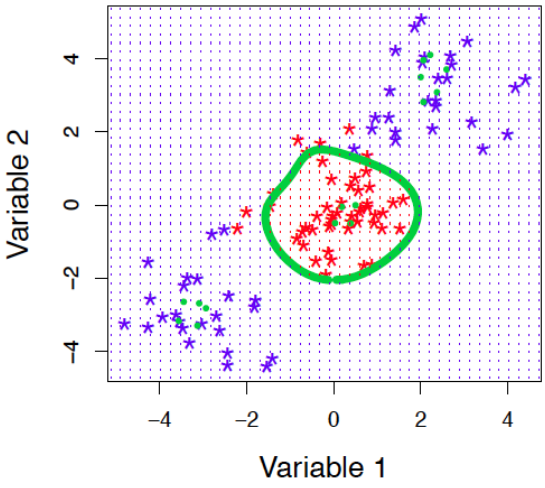
This will be hard for a linear classifier!

Try a Support Vector Classifier



Uh-oh!!

Support Vector Machine

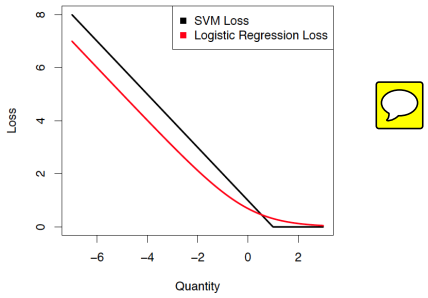


Much Better.

Is A Non-Linear Kernel Better?

- Yes, if the true decision boundary between the classes is non-linear, and you have enough observations (relative to the number of features) to accurately estimate the decision boundary.
- No, if you are in a very high-dimensional setting such that estimating a non-linear decision boundary is hopeless.

Support Vector Classifier Versus Logistic Regression



- Bottom Line: Support vector classifier and logistic regression aren't that different!
- Neither they nor any other approach can overcome the "curse of dimensionality".
- SVM uses a non-linear kernel... but could do that with logistic or linear regression too!

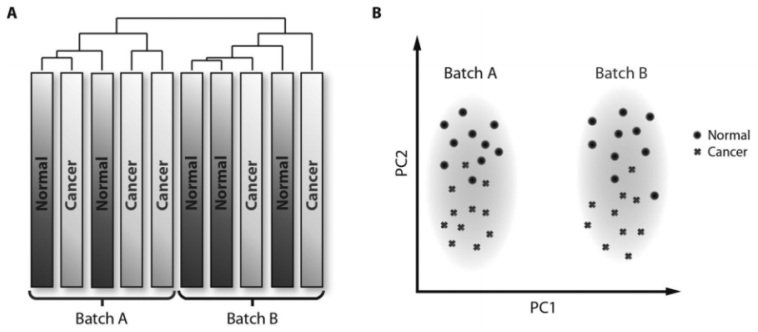
In High Dimensions...

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a ridge penalty, both mathematically and conceptually. The SVM decision rule involves all of the variables.
- Can get a sparse SVM using a lasso penalty; this yields a decision rule involving only a subset of the features.
- Logistic regression and other classical statistical approaches could be used with non-linear expansions of features. But this makes high-dimensionality issues worse.

Batch Effects

- In any sort of omics experiment, need to be very aware of batch effects, induced by non-biological factors such as inter-machine or inter-lab or inter-operator variability, time of day, day of week, position of ceiling fan, ...
- It has been shown many many times that batch effects can be much stronger than biological effects of interest!
- If you are not careful, batch effects can result in complete confounding, making your data worthless and your results nonsense.

Batch Effects



Steps to Reduce Batch Effects

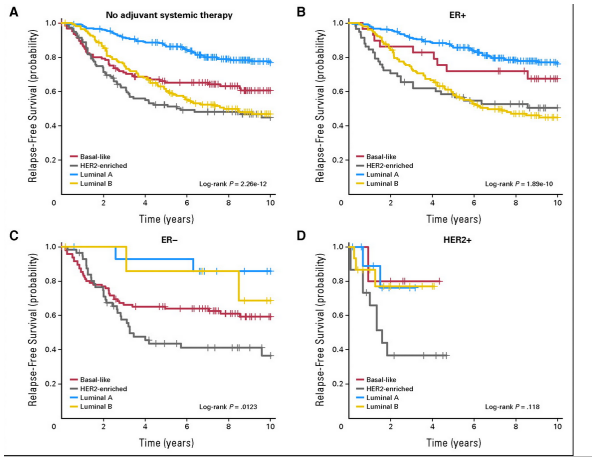
- Randomize sample run times: e.g. don't run cases first and controls second.
- Avoid any extraneous sources of variation, e.g. due to change in person running the experiment.
- It is often better to train a classification or regression method using multiple data sets collected at different institutions, rather than using a single data set.
- Need to validate any results obtained on independent data sets from a different institution.

Batch effects are almost inevitable. But you can do your best to design an experiment and analyze the data in such a way that batch effects do not compromise the results obtained.

Subtypes of Breast Cancer

- In the past 10 years, global gene expression analyses have identified at least 4 subtypes of breast cancer: Luminal A, Luminal B, Her2-enriched, and basal-like.
- Subgroups differ with respect to risk factors, incidence, baseline prognoses, responses to therapies.
- Want to be able to determine the subtype for a new patient with breast cancer.
- Controversy over the best classifier for this task:
  - PAM50 classifier involves 50 genes.
  - More recent proposal involving three genes.
- Moving target: nobody knows the "true" subtype!
- Prat et al., Breast Cancer Res Treat, 2012

Why Do We Care About Subtypes?



Citation: Parker et al, Journal of Clinical Oncology, 2009

Proteomics for Ovarian Cancer

- Ovarian cancer is the leading cause of gynecologic cancer deaths in the USA.
- Much interest in detecting the cancer at an earlier stage.
- In 2002, Petricoin and Liotta – investigators from FDA and NCI – reported in The Lancet that mass spectrometry analysis of circulating serum proteins can be used to discriminate between healthy patients and those with ovarian cancer.
- Great enthusiasm in the popular press and general public.
- Plans were made to begin marketing a test based on the reported diagnostic.

Not So Fast!!

- Independent researchers took a look at the data, which was publicly available, and discovered:
  - inadvertent changes in protocol mid-experiment: i.e. major batch effects.
  - problems with instrument calibration.
  - difference in processing between tumor and normal samples.
- In summary: the observed differences between cancer and normal proteomic patterns were attributable to "artifacts of sample processing, not the underlying biology of cancer."

Gene Expression Signatures for Cancer Treatment

- In the early 2000's, Joe Nevins, Anil Potti, and other researchers at Duke University began developing expression-based predictors of response to chemotherapy.
- Many (dozens of!) very promising and very high-profile papers were published in Nature Medicine, The Lancet, Journal of Clinical Oncology, and more.
- Several clinical trials were initiated, using these predictors to direct therapy for cancer patients.
- This research was hailed as a major breakthrough in cancer treatment, and researchers from all over the world tried to use these sorts of techniques in their own labs.

Upon Closer Inspection....

- Using the fact that some of the data were publicly available, independent researchers discovered the following errors (among many others):
  - Off-by-one errors in gene lists
  - The same heatmap displayed in multiple (unrelated) papers
  - Genes not measured on the array were reported as being part of the predictor obtained, and as providing evidence for biological plausibility
  - Reversal of sensitive/resistant labels
- A shocking paper published by Baggerly and Coombes in Annals of Applied Statistics, detailing all of the errors made: "One theme that emerges is that the most common errors are simple (e.g., row or column offsets); conversely, it is our experience that the most simple errors are common."

What Went Wrong?

A blasé approach to high-dimensional data analysis:

- Need to have a proper independent test set, that you simply cannot peek at under any circumstances!
- Need to have clearly documented code that contains all steps of the analysis, from start to finish. You must be able to share this code with independent researchers, and you must be confident that your code is correct. If not, then your work isn't ready for prime time.

The Stakes are High!

At Duke:

- Dozens of papers retracted;
- Careers and reputations ruined;
- Patients endangered through unethical clinical trials.

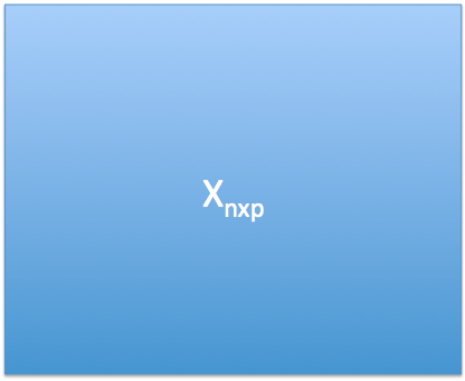
Plus, a 60 Minutes special feature and an Institute of Medicine Committee!!!

HIGH-DIMENSIONAL OMICS DATA: Dimension Reduction Methods - I

Ali Shojaie & Daniela Witten

July 21-23 2014
Summer Institute for Statistical Genetics
University of Washington

Unsupervised Learning: A Reminder



- n number of observations
- p number of variables/features
- no *response* variable
- In biological applications, often $p \gg n$

Unsupervised Learning Examples

- Do genes/samples in microarray expression data form interesting groups?
- Can individuals' SNP profiles be used to learn about their ethnic/racial backgrounds?
- Can we find cancer subtypes based on gene/metabolic expression patterns?
- What's the best way to visualize a high dimensional genomic data?
- How to find "interesting patterns" in the data?
- Which genes/protiens/metabolites are "associated" with the disease (this is really not an unsupervised learning question, but somewhat related...)?

Unsupervised Learning Methods

- Dimension Reduction: Find **low dimensional representation** of data; this is a very useful tool for **discovering patterns** in the data, and also improving performance of regression and prediction methods (recall PCR)
  - PCA: Principal Component Analysis
  - MDS: Multi-Dimensional Scaling
  - Sparse PCA, Kernel PCA, ICA, Manifold Learning, ...
- Cluster Analysis: Find **similar groups** of variables/samples; this can be the final goal of the analysis, or a preliminary step
  - Hierarchical Clustering: Agglomerative (bottom-up) clustering
  - Partition-based Methods: K-means, Model-based clustering, Spectral clustering
  - Self Organizing Maps, bi-clustering, ...
- Multiple Hypothesis Testing: Find genes/proteins/metabolites that are associated with the response
  - Family wise error rates (Bonferroni correction)
  - False discovery rate control (FDR)

Challenges in Unsupervised Learning

- Unlike supervised learning, there is no direct method for calculating p-values, or performing cross validation
- Comparison and selection of method becomes more difficult
- Difficult to validate the results of analysis 
- In high dimensional settings, choices for displaying results of analysis are limited
- Unsupervised methods are often based on notions of "similarity" or "distance". When $p \gg n$, these become less informative/accurate

The Plan

This is roughly our plan, although we may not follow it very closely...

- **Lecture 6**: PCA
- **Lecture 7**: PCA continued, MDS
- **Lecture 8**: Clustering (background and general issues, hierarchical clustering)
- **Lecture 9**: Clustering (K-means clustering, spectral clustering)
- **Lecture 10**: Multiple hypothesis testing

Why Dimension Reduction?

When dealing with high dimensional omics data ($p \gg n$)

- Data visualization becomes very difficult (cannot draw 2D scatterplots for large p).
- Prediction accuracy of traditional statistical models reduces.
- High dimensional data often have high degrees of redundancy (correlation among features).
- Many features may be uninformative for the particular problem under study (noise features).
- Dimension reduction ideally allows us retain information on most important features of the data, while reducing noise and simplifying visualization & analysis.

What is Dimension Reduction?

- Map the data into a new low-dimensional space, where important characteristics of the data are preserved.
- The new space often gives a (linear or non-linear) transformation of the original data.
- Visualization and analysis (clustering/prediction/...) is then performed in the new space.
- In some cases, (especially for non-linear transformations) interpretation becomes difficult.

Methods of Dimension Reduction

- Principal Component Analysis (PCA)
- Multi-Dimensional Scaling (MDS)
- Kernel PCA, Sparse PCA, Manifold Learning, ...

Principal Component Analysis (PCA)

Recall: PCR provides improvements for regression models in high dimensional settings.

- The idea in PCA is similar to PCR, but PCA is unsupervised!
- There is no response variable y, and the goal is data visualization or pattern discovery.
- In some cases, PCA is also used directly to find subclasses of observations with heterogenous properties (admixture populations, population stratification, ...).

Examples of PCA Applications

PCA is widely used in population genetics and genome-wide association studies: **to correct for stratification**.

nature genetics

Principal components analysis corrects for stratification in genome-wide association studies

Alkes L Price[1,2], Nick J Patterson[2], Robert M Plenge[2,3], Michael E Weinblatt[3], Nancy A Shadick[3] & David Reich[1,2]

Population stratification—allele frequency differences between cases and controls due to systematic ancestry differences—can cause spurious associations in disease studies. We describe a method that enables explicit detection and correction of population stratification on a genome-wide scale. Our method uses principal components analysis to explicitly model ancestry differences between cases and controls. The resulting correction is specific to a candidate marker's variation in frequency across ancestral populations, minimizing spurious associations while maximizing power to detect true associations. Our simple, efficient approach can easily be applied to disease studies with hundreds of thousands of markers.

Examples of PCA Applications

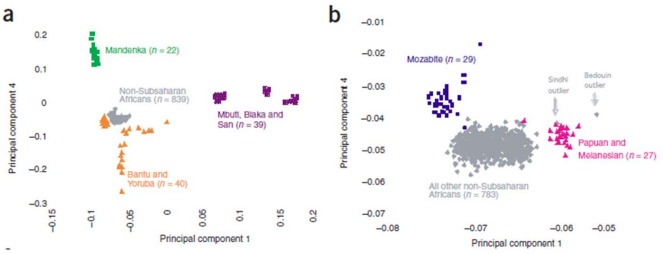
PCA is widely used in population genetics and genome-wide association studies: for the **study of human migration patterns**.

Principal component analysis of genetic data

David Reich, Alkes L Price & Nick Patterson

Principal component analysis (PCA) has been a useful tool for analysis of genetic data, particularly in studies of human migration. A new study finds evidence that the observed geographic gradients, traditionally thought to represent major historical migrations, may in fact have other interpretations.

Principal component analysis (PCA) has been used for several decades to study human population migrations, resulting in remarkable inferences about history. On page 646 of this issue, John Novembre and Matthew Stephens[1] show that the geographic gradients that emerge when PCA is applied to genetic data—and that are sometimes interpreted as highly suggestive of major historical migrations—can also have other explanations. We suggest guidelines for scientists interested in using PCA in genetic



Examples of PCA Applications

PCA is widely used in population genetics and genome-wide association studies: in the **study of admixture populations**.

Principal Component Analysis under Population Genetic Models of Range Expansion and Admixture

Olivier François,*[1] Mathias Currat,[2] Nicolas Ray,[3,4,5] Eunjung Han,[5] Laurent Excoffier,[3,4] and John Novembre[6,7]

[1]Laboratoire Techniques de l'Ingénierie Médicale et de la Complexité, Faculty of Medicine, University Joseph Fourier, Grenoble Institute of Technology, Centre National de la Recherche Scientifique UMR5525, La Tronche, France

[2]Laboratory of Anthropology, Genetics and Peopling history, Department of Anthropology and Ecology, University of Geneva, Geneva, Switzerland

[3]Computational and Molecular Population Genetics Lab, Institute of Ecology and Evolution, University of Berne, Berne, Switzerland

[4]Swiss Institute of Bioinformatics, Lausanne, Switzerland

[5]EnviroSPACE laboratory, Climate Change and Climate Impacts, Institute for Environmental Sciences, University of Geneva, Carouge, Switzerland

[6]Department of Ecology and Evolutionary Biology, University of California

[7]Interdepartmental Program in Bioinformatics, University of California-Los Angeles

Examples of PCA Applications

PCA is widely used in population genetics and genome-wide association studies: to **discover SNP sets** for pathway analysis

Genetic Epidemiology 26: 11–21 (2004)

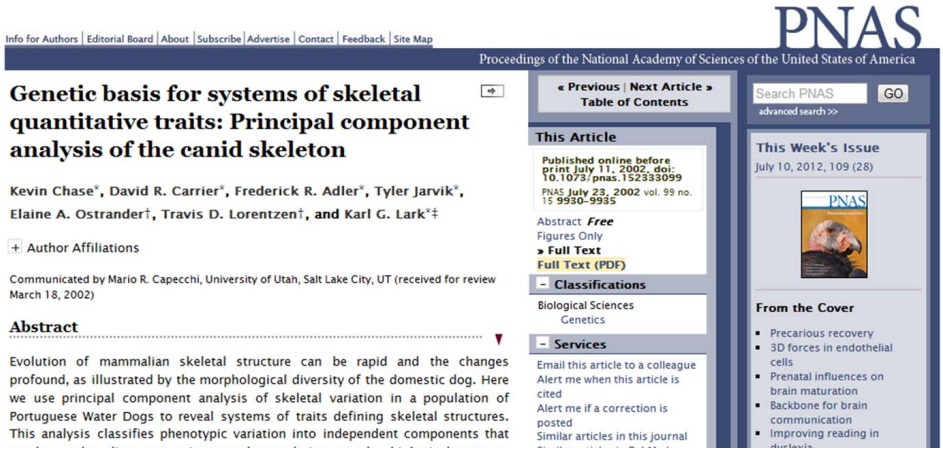
Principal Component Analysis for Selection of Optimal SNP-Sets That Capture Intragenic Genetic Variation

Benjamin D. Horne[1,2] and Nicola J. Camp[1]

[1]*Genetic Epidemiology Division, Department of Medical Informatics, University of Utah, Salt Lake City, Utah*
[2]*Cardiovascular Department, LDS Hospital, Salt Lake City, Utah*

Examples of PCA Applications

It is also used in a variety of other applications...

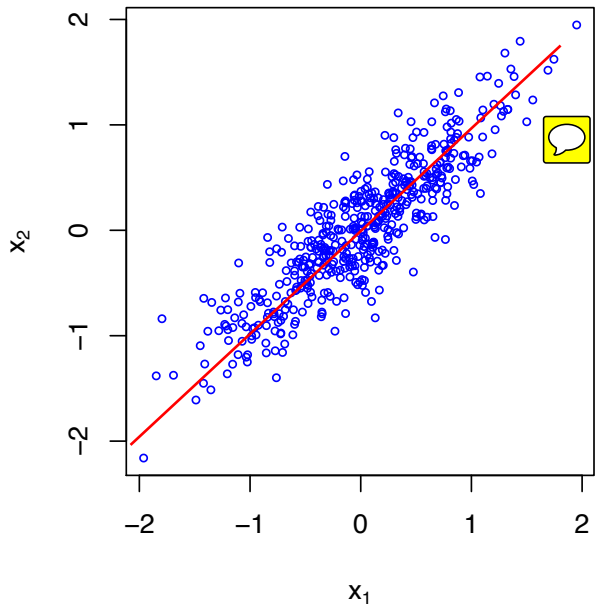


PCA: An Overview

- Data: n observations living in a p-dimensional space.
- Not all p dimensions are equally useful, especially when $p \gg n$.
- Many are either completely redundant (correlated features) or uninformative (noise features).
- Need low-dimensional representation of the variables that captures most of the "information" in the data.
- To maximize the information retained, we need to minimize the redundancy, and to do this, we look for low-dimensional representations that capture most of the variation in the data.

PCA: The Main Idea

Question: What is a good 1-dim representation of the data?



PCA: The Main Idea

Some Possibilities:

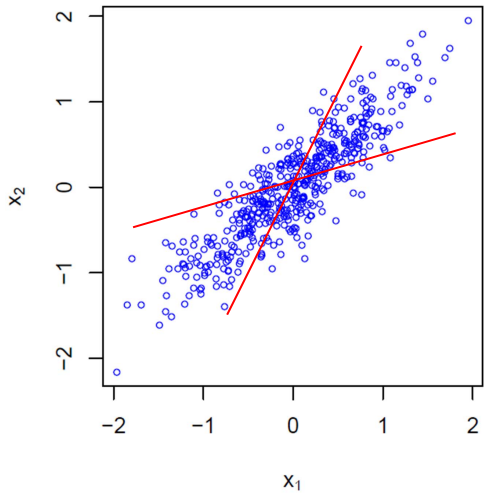
- Use one of the variables (e.g. X_1).
- Better idea: use a linear combination of the variables; i.e. a weighted average of the variables.

$$Z_1 = w_1 X_1 + w_2 X_2 = w^\top X$$

Question: what is a good choice for the weights w_i?

PCA: The Main Idea

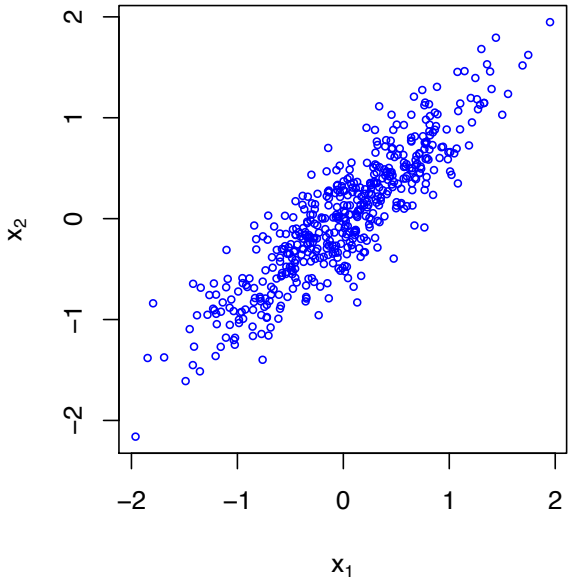
Many possibilities, but which one is a good choice?



Need some criterion for a principled choice of the weights.

The Criterion for Principal Components

- In PCA, we try to find the direction with maximum variance.



The Criterion for Principal Components

- In PCA, we try to find the direction with maximum variance.
- *Formally*, we find the vector of weights w using the following **criterion**:

$$\max_{w:\|w\|=1} \mathrm{Var}(w^\mathsf{T} X)$$

- But, we know that

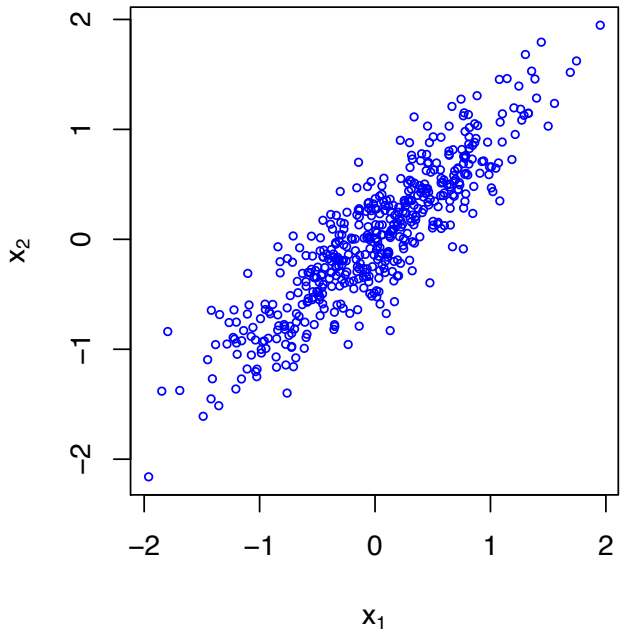
$$\mathrm{Var}(AX) = A^\mathsf{T} \mathrm{Var}(X) A$$

- So, we can write our criterion as:

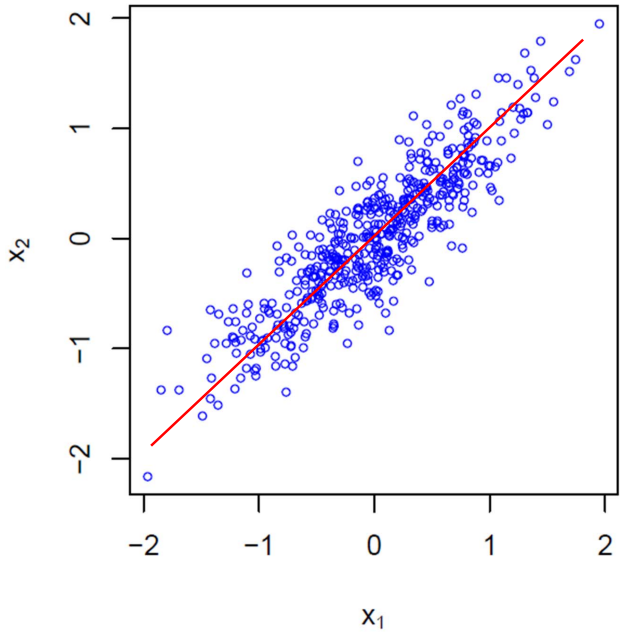
$$\max_{w:\|w\|=1} w^\mathsf{T} \mathrm{Var}(X) w = \max_{w:\|w\|=1} w^\mathsf{T} \Sigma w$$

  Where Σ is the covariance matrix of the X.
- I.e., the interesting direction according to the PCA criterion is the one that captures the majority of the variance in the data.

The 1-Dimensional PCA Solution



The 1-Dimensional PCA Solution

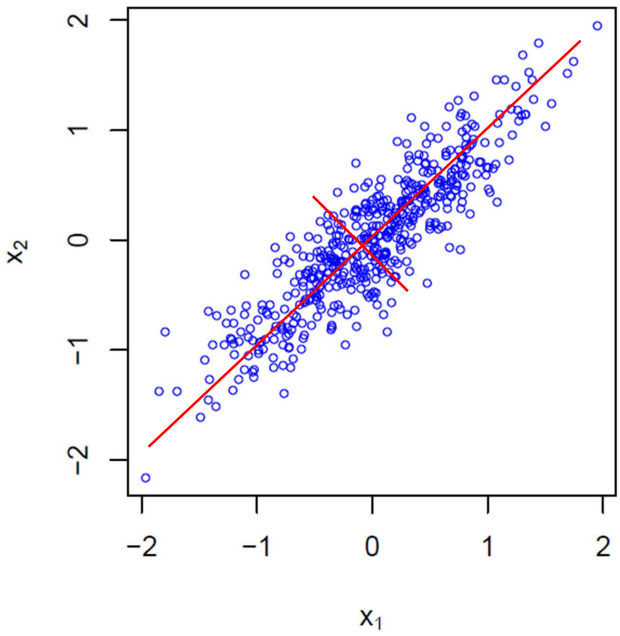


- But, what if we need another direction:

$$Z_2 = v_1 X_1 + v_2 X_2 = v^T X$$

- A systematic way to find additional principal components (PC's), is to choose subsequent linear combinations orthogonal/perpendicular to previous ones.
- This means that we want to choose v to be orthogonal to w, but to explain the majority of variability in the data.
- In the case of 2-dimensional data, there is only one choice!! This is always the case for the last PC.
- For $p > 2$, there are many orthogonal vectors to choose from, and we need to find the one that explains the maximum variation in the data, and is orthogonal to the first one

The Full PCA Solution for 2 Dimensions



PCA: The General Problem

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M \leq p$ linear combinations of the p predictors:

$$Z_m = \sum_{j=1}^{p} w_{mj} X_j.$$

- $Z_1, \ldots, Z_M$ are chosen to be the principal components of the data:

$$w_m = \max_{w:\|w\|=1} \text{Var}(w^T X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}$$

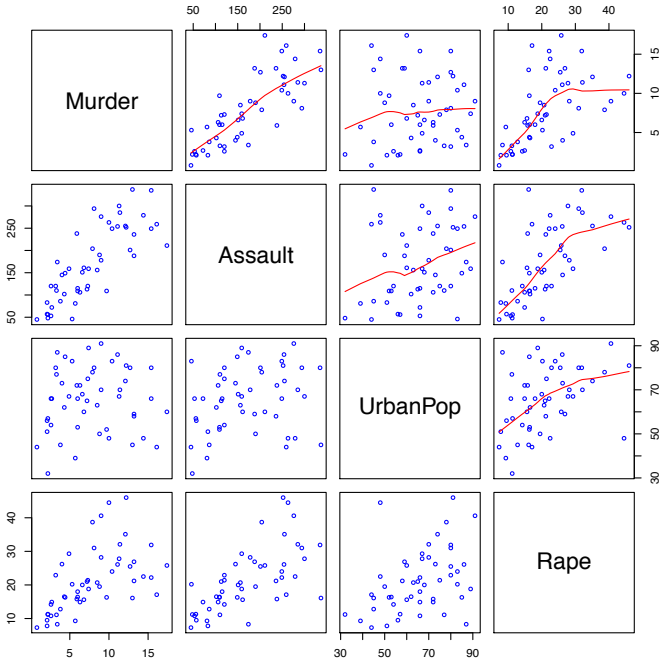
- w_m's are called factor loadings or PC loadings
- Z_m's are called principal components (PCs) or PC scores

Example: PC Analysis of `USArrests` data

Data Description:

```
This data set contains statistics, in arrests per 100,000 residents
for "assault", "murder", and "rape" in each of the 50 US states in
1973. Also given is the "percent of the population living in urban areas".

A data frame with 50 observations on 4 variables:
[,1]  Murder     numeric    Murder arrests (per 100,000)
[,2]  Assault    numeric    Assault arrests (per 100,000)
[,3]  UrbanPop   numeric    Percent urban population
[,4]  Rape       numeric    Rape arrests (per 100,000)
```


Take a Look At the Data



Example: PC Analysis of `USArrests` data

What are the means for each of the variables?

Variable	Murder	Assault	UrbanPop	Rape
Mean	7.788	170.760	65.540	21.232

What are the variances for each of the variables?

Variable	Murder	Assault	UrbanPop	Rape
Variances	18.97047	6945.16571	209.51878	87.72916

Vastly different means and variances, what happens if we fit PCA to this data?

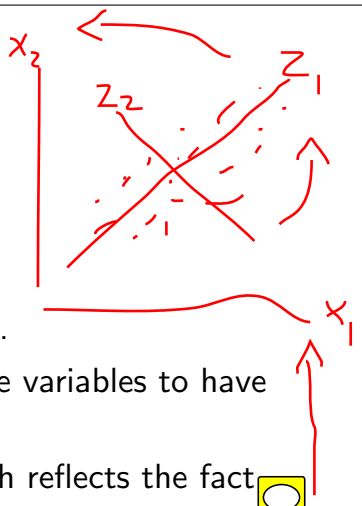
Some Remarks

- In PCA, it is assumed that the variables are centered. So remember to always center the variables, before performing PCA.
- PCA works with both standardized (scaled) or unscaled data; however, the solutions may be different!
- The PCA solutions are sensitive to scale of variables, and variables with larger variance will affect the results more.
- This may not necessarily desirable in many applications, therefore, it is better to also standardize the variables.

Example: PC Analysis of `USArrests` data

```
> states <- row.names(USArrests)
> states[1:5]
[1] "Alabama"    "Alaska"     "Arizona"    "Arkansas"   "California"

> apply(USArrests, 2, mean)
  Murder  Assault UrbanPop     Rape
   7.788  170.760   65.540   21.232

> apply(USArrests, 2, var)
    Murder    Assault   UrbanPop       Rape
  18.97047 6945.16571  209.51878   87.72916

> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```


PC Analysis in R



- By default, `prcomp` centers the variables.
- The option `scale=TRUE` standardizes the variables to have standard deviation 1.
- The `rot` is shorthand for **rotation**, which reflects the fact that PC's are linear combinations of the original variables.
- Each PC (column) in the above table gives the weights for the linear combination for calculating the mth PC. So the columns of the above table gives the *PC loadings*.

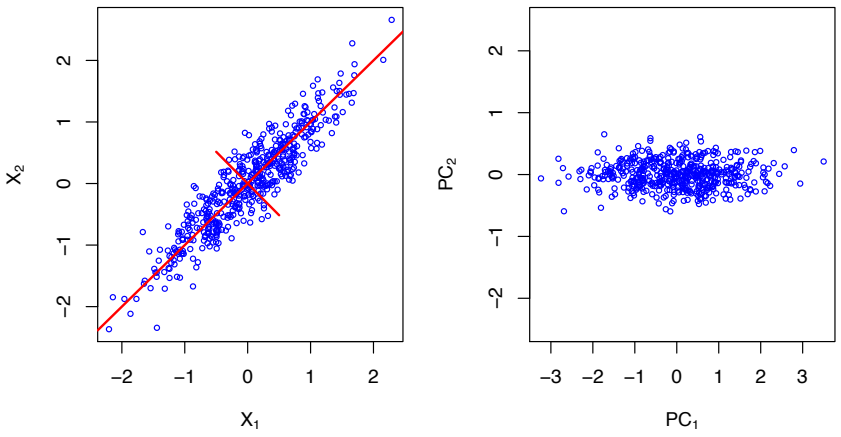
PC Loadings

```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

- There are 4 distinct PC loadings (in general $\min(n, p)$).
- Each PC loading is unique up to a sign flip (change of sign doesn't change the direction), so multiplying by -1 does not change the PCs.
- We can easily get the PC's from the PC loadings: the mth PC score vector is given by $Z_m = Xw_m$, where w_m is the mth PC loading (mth column of the above table).
- More generally, we can get the PC score matrix $Z = XW$, where W is the matrix of PC loadings in the above table.

PC Loadings

- Observations can be plotted ("projected") in the space of PC's.
- Roughly, this is equivalent to rotating the axes and plotting the original data points in the new axes Z_1 and Z_2.



PC Loadings

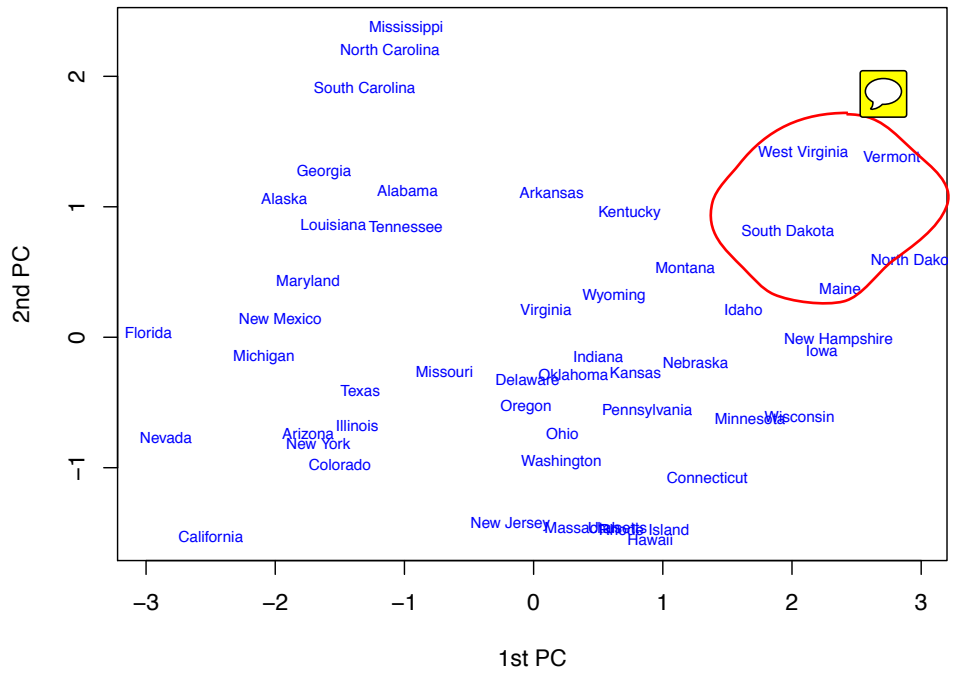
- Observations can be plotted ("projected") in the space of PC's.
- Roughly, this is equivalent to rotating the axes and plotting the original data points in the new axes Z_1 and Z_2.
- We can get these by setting `retx=TRUE` in the `prcomp` call:

```
pc.out <- prcomp(USArrests, scale=TRUE, retx=TRUE)
```

- `pc.out$x` is a matrix of dimension 50×4, which has as its columns the PC score vectors
- Plotting the observations in the space of PC scores can reveal interesting relationships between them.

```
plot(pc.out$x[,1], pc.out$x[,2], type="n", xlab="1st PC", ylab="2nd PC")
text(pc.out$x[,1], pc.out$x[,2], labels=states)
```

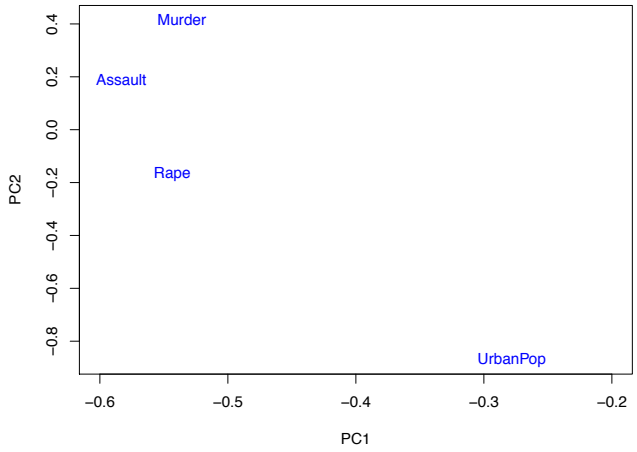

Plotting **Observations** in the Space of PC's



Plotting **Variables** in the Space of PC's

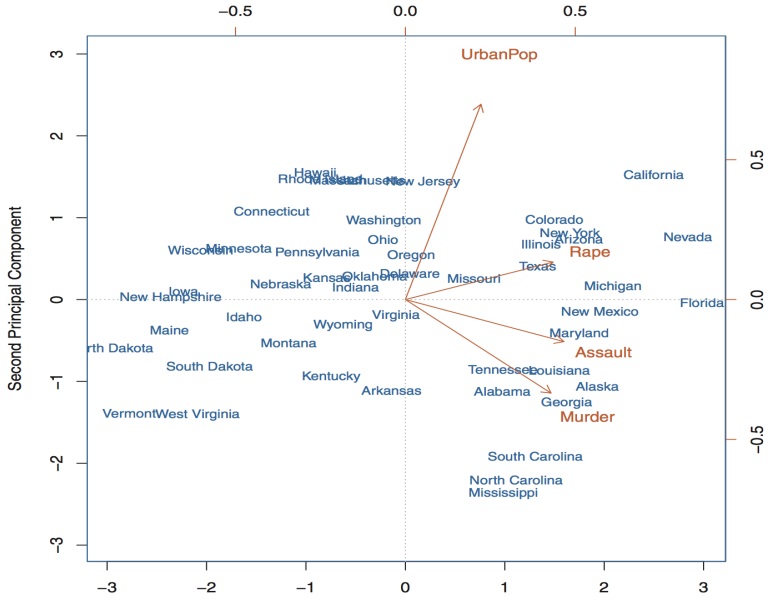
We can also plot the variables in the space of PC loadings:

```
plot(pc.out$rot, type="n")
text(pc.out$rot, names(USArrests), col=4)
```



Plotting **Both** Variables and Observations

We can do this using a `biplot()`



Recap

- PCA results in two main objects:
  1. PC loadings: these are the weights w_ms for the linear combinations of the original variables
  2. PC scores (or PCs): these are the projected observations: $Z_m = \sum_{j=1}^{p} w_{mj} X_j$
- can plot observations in space of PC scores; these are found from `pc.out$x`
- can plot variables in the space of PC loadings; these are found from `pc.out$rot`

Exercise

- Repeat the analysis without normalizing the data (i.e. `scale=FALSE`) and compare the results of the analyses.
- Repeat the analysis by (i) centering and (ii) scaling the data manually in R prior to use of `prcomp`.

HIGH-DIMENSIONAL OMICS DATA: Dimension Reduction Methods - II

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

PCA and Dimension Reduction

Recall the USArrests data:

```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

PCA and Dimension Reduction

Recall the USArrests data:

```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Q: With 4 PC's again have a 4 dimensional space, just as the original data, so why is PCA a dimension reduction technique?

PCA and Dimension Reduction

Recall the USArrests data:

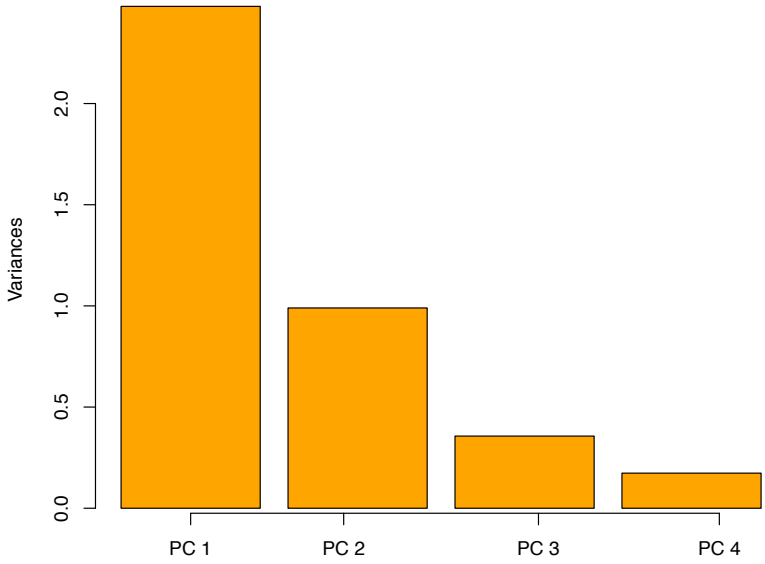
```
> pc.out <- prcomp(USArrests, scale=TRUE)
> print(pc.out$rot)
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Q: With 4 PC's again have a 4 dimensional space, just as the original data, so why is PCA a dimension reduction technique?

A: Not if we use all 4 PC's! The **dimension reduction** in PCA comes from the fact that often few PC's explain most of the variation in the data.

Variances Explained by Each PC

```
pc.out <- prcomp(USArrests, scale=TRUE, retx=TRUE)
screeplot(pc.out)
```



The Proportion of Variance Explained

- In general, we would like to know, "how much of the information in the data is lost by projecting the observations onto the first M principal components".
- In other words, "how much of the variance in the data is not contained in the first M principal components"?
- To answer these question, we need to know the proportion of variance explained (PVE) by each principal component.

The Proportion of Variance Explained (ctd.)

- The total variance in the data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2$$

- The variance explained by a PC loading vector w is:

$$\sum_{i=1}^{n}(\sum_{j=1}^{p} X_{ij} w_j)^2$$

- So, the proportion of variance explained by each principal component loading vector is:

$$\frac{\sum_{i=1}^{n}(\sum_{j=1}^{p} X_{ij} w_j)^2}{\sum_{i=1}^{n}\sum_{j=1}^{p} X_{ij}^2}$$

Variances Explained by Each PC

- Let's look at the variances for each of the PC's:

Variance	PC1	PC2	PC3	PC4
	2.4802	0.9898	0.3566	0.1734

- These are the values plotted previously in the barplot
- If we take all of the PCs, we can explain all the variance in the data (if $p < n$)
- To compute the PVE for each principal component, we simply divide the variance explained by each principal component by the total variance explained by all four principal components:

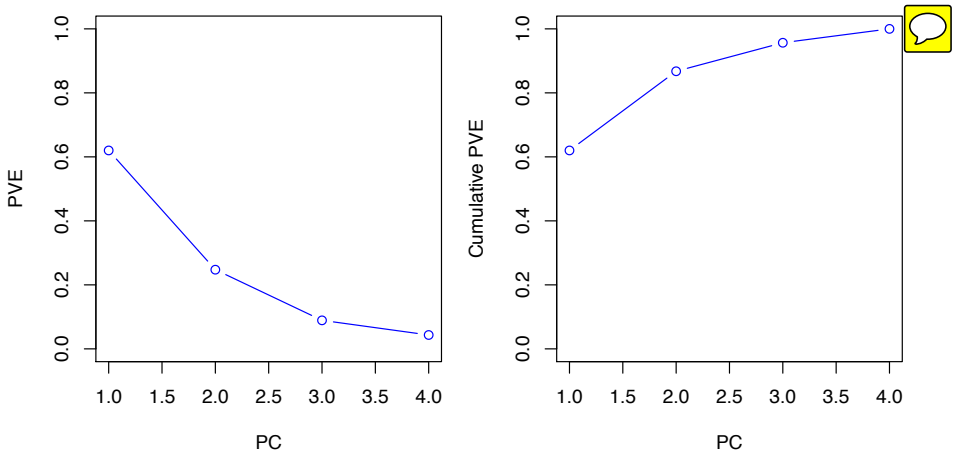
PVE	PC1	PC2	PC3	PC4
	0.620	0.247	0.089	0.043

- We can see that the first 2 PC's explain 87% of the variance in the data.

Scree Plots

- We can plot the PVE as well as the cumulative PVE:

```
plot((pc.out$sdev^2)/sum(pc.out$sdev^2), xlab="PC",
    ylab="PVE", ylim=c(0, 1), type='b')
plot(cumsum(pc.out$sdev^2)/sum(pc.out$sdev^2), xlab="PC",
   ylab="Cumulative PVE", ylim=c(0, 1), type='b')
```

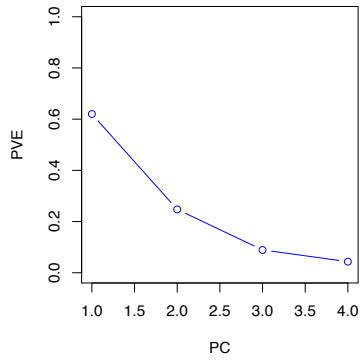


How Many PC's?

- A natural question that arises in applications of PCA in high dimensional settings, is how many PC's should we use?
- On the one hand, by increasing the number of PC's, we include more of the variation in the data.
- On the other hand, if we use too many PC's, we don't get the benefits of dimension reduction (e.g. in visualization and prediction).
- We want to use the smallest number of PC's that would give us a "good" understanding of the data.
- Unfortunately, like many other questions in unsupervised learning, there is no easy answer to this question.

How Many PC's?

- One idea is to use enough PC's that explain a significant proportion of variances in the data.
- To answer this question, we can use the plots in the previous slide to decide the number of PCs to include.
- This is done by eyeballing the scree plot, and looking for a point at which the PVE by each subsequent PC drops off. This is often referred to as an "elbow" in the scree plot.



PCA in High Dimensions: NCI60 Data

- The dataset includes gene expression data for 6830 genes from 64 cancer samples (from different cancer subtypes)
- Data can be downloaded form http://www-stat.stanford.edu/~tibs/ElemStatLearn/
- Missing values have been imputed
- For this analysis (and to simplify the plots), I have removed 5 subtypes with only 1 samples:

  `UNKNOWN, K562B-repro, K562A-repro, MCF7A-repro, MCF7D-repro`
- Let's perform PCA on this data!

PCA in High Dimensions: NCI60 Data

- First, which one of these datasets should we use?

```
> dat.1 <- read.table('nci-mod.data')
> dim(dat.1)
[1] 6830   59
> dat.2 <- t(dat.1)     #transposing the data
> dim(dat.2)
[1]   59 6830
```

- How many PC's will we get from applying PCA to `dat.1`? How about `dat.2`?
- What's the difference? Which one should we use?

PCA in High Dimensions: NCI60 Data (ctd.)

Let's try both!

```
> pc.1 <- prcomp(dat.1, scale=TRUE, retx=TRUE)
> pc.2 <- prcomp(dat.2, scale=TRUE, retx=TRUE)

> dim(pc.1$rot)
[1] 59 59
> dim(pc.1$x)
[1] 6830   59
> dim(pc.2$rot)
[1] 6830   59
> dim(pc.2$x)
[1] 59 59
```

PCA in High Dimensions: NCI60 Data (ctd.)

Let's try both!

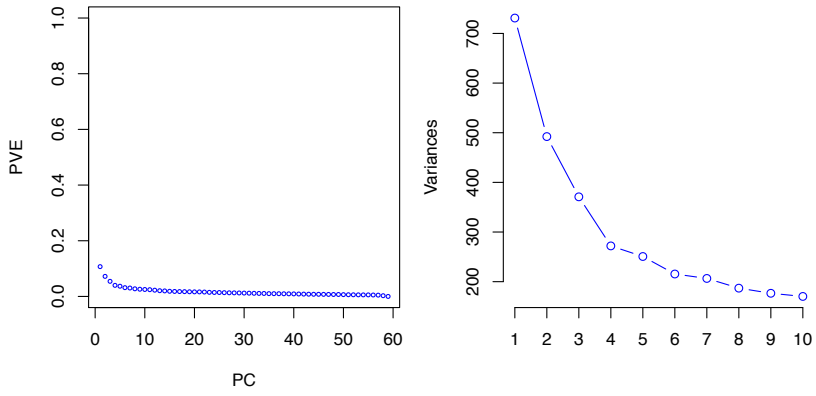
```
> pc.1 <- prcomp(dat.1, scale=TRUE, retx=TRUE)
> pc.2 <- prcomp(dat.2, scale=TRUE, retx=TRUE)

> dim(pc.1$rot)
[1] 59 59
> dim(pc.1$x)
[1] 6830   59
> dim(pc.2$rot)
[1] 6830   59
> dim(pc.2$x)
[1] 59 59
```

Remarks:

- Each PC loading should be of the same length as the number of variables! (each column of `$rot` is a PC loading)
- There are at most $\min(n, p)$ PC loadings; for $p \gg n$, we have at most n PC's.
- Variables should always be in columns of X!

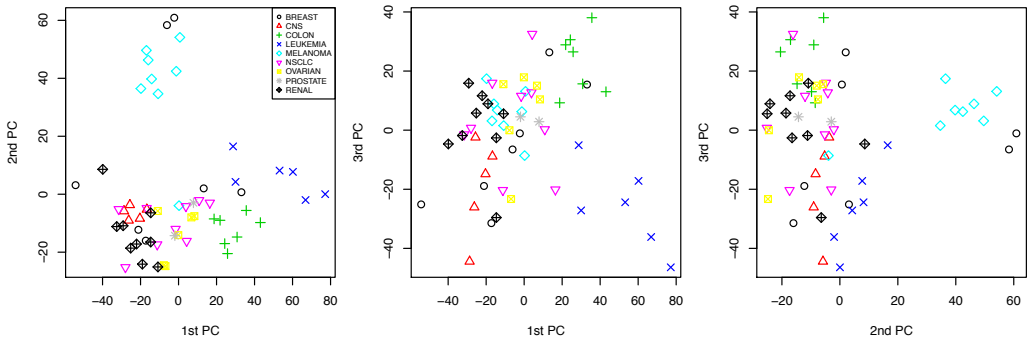
How Many PC's?

- In high dimensions, the proportion of variance explained for each PC is often small.
- However, the first few PC's usually provide a good projection, and we can usually find an "elbow" in the scree plot. Here, 4 PC's seems to be a good choice.



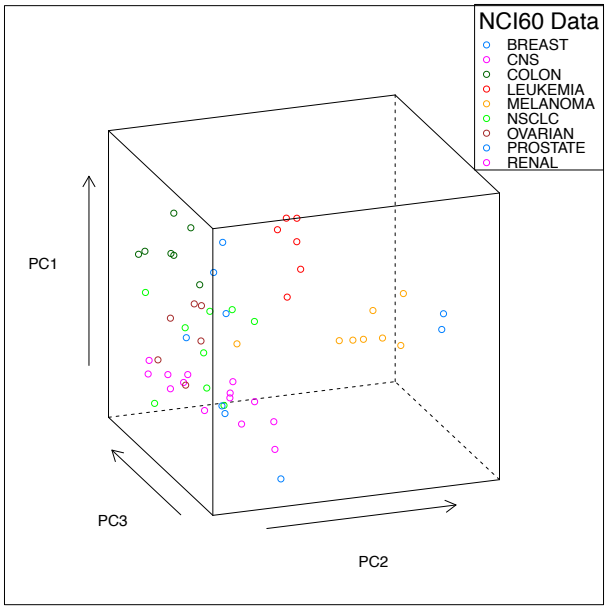
Projecting Samples into the PC Space

- Let's look at the space of the first 3 PC's



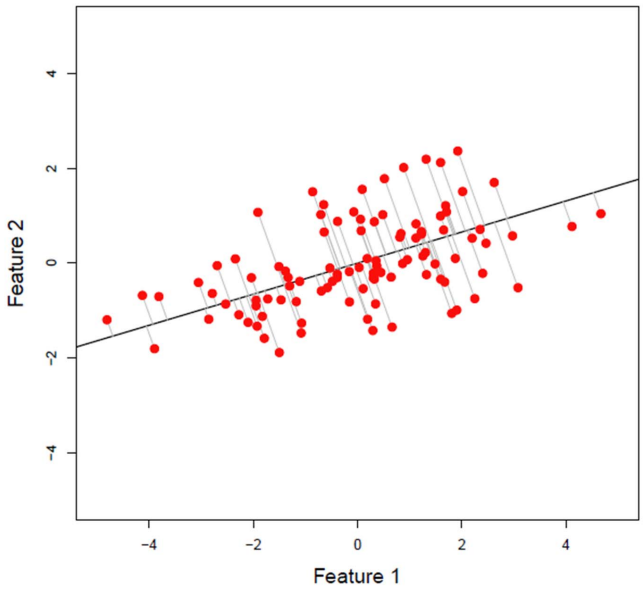
- After projecting the samples into the space of PC's, we can perform clustering or classification in the new space.

A Different View



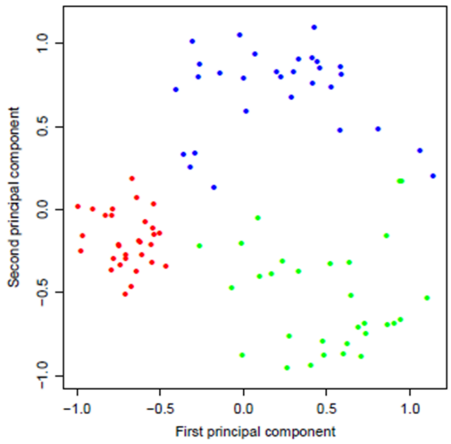
Another look at PCA

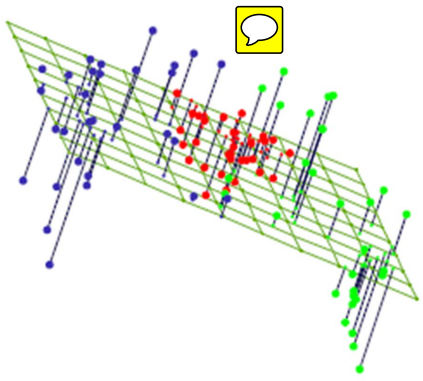
The first PC gives the direction with minimum orthogonal distance from the observations.



Another look at PCA

In general, the PCA Solution finds a *hyperplane* that is closest to the data in terms of squared orthogonal distance.





The PCA Solution

- Recall the PCA problem:

$$w_m = \max_{w:\|w\|=1} \text{Var}(w^T X) \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- Using the fact that, $\text{Var}(w^T X) = w^T \text{Var}(X) w = w^T \Sigma w$ (with Σ the covariance matrix of X), we can write this problem as:

$$w_m = \max_{w:\|w\|=1} w^T \Sigma w \text{ and } w_m \perp \{w_1, \ldots w_{m-1}\}.$$

- In compact matrix form, the problem can be written as:

$$\max_{W:W^T W=I} W^T \Sigma W$$

- This is a well-known problem in mathematics, and its solution is known to be given by the eigenvectors of Σ (i.e. PC loadings are eigenvectors of the covariance matrix).

The PCA Solution (ctd.)

- v is an eigenvector of Σ iff $\exists \lambda$ such that $\Sigma v = \lambda v$; λ is an eigenvalue of Σ
- This means that Σ only stretches the vector v, but does not change its direction
- We can also write the eigen decomposition of Σ as $V \Lambda V^T = \sum_1^p \lambda_j v_j v_j^T$.
  - Here, Λ is a diagonal matrix of eigenvalues.
  - The matrix of eigenvectors V gives the PC loading vectors.
  - The eigenvalues give the amount of variance explained by each eigenvector
  - For symmetric matrices, the eigenvectors are orthogonal: $v_j^T v_k = 0, j \neq k$

The PCA Solution (ctd.)

- In practice, the PCA problem is solved using singular value decomposition (SVD) on the data matrix X, which gives equivalent, but more robust results (especially for high dimensional problem).
- The `R` function `prcomp` solves the PCA problem using the SVD of X.
- However, there is another function in R for PCA: `princomp`, which solves the PCA problem by finding the eigen-decomposition of X.
- In most cases, these functions produce identical results. However, the method of estimation of PC's in the two functions is different.

More on PCA

- PCA doesn't yield feature selection – all of the original predictors are involved in the final model.
- But when M is small, then PCA offers efficient dimension reduction, which can result in better prediction and visualization.
- Often PCA results are used to gain insight into high dimensional data. Trying to guess what the components might mean is a good idea, but you should not over-interpret the results of PCA.
- The ideas and insight that we get from PCA is for exploration and pattern discovery, and we cannot say with certainty whether our interpretations of the PC's are accurate (e.g. the sign is arbitrary)
- And finally, remember that this is unsupervised learning!

Be Careful!

nature genetics

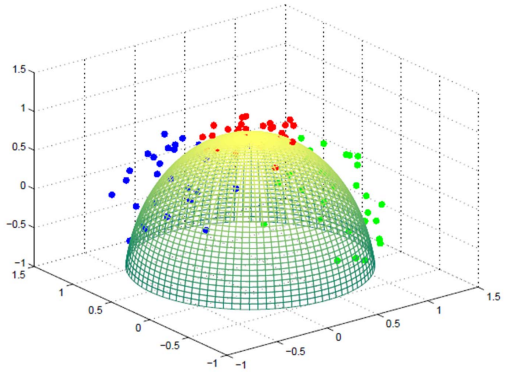
Interpreting principal component analyses of spatial population genetic variation

John Novembre[1,3] & Matthew Stephens[1,2]

Nearly 30 years ago, Cavalli-Sforza *et al.* pioneered the use of principal component analysis (PCA) in population genetics and used PCA to produce maps summarizing human genetic variation across continental regions[1]. They interpreted gradient and wave patterns in these maps as signatures of specific migration events[1–3]. These interpretations have been controversial[4–7], but influential[8], and the use of PCA has become widespread in analysis of population genetics data[9–13]. However, the behavior of PCA for genetic data showing continuous spatial variation, such as might exist within human continental groups, has been less well characterized. Here, we find that gradients and waves observed in Cavalli-Sforza *et al.*'s maps resemble sinusoidal mathematical artifacts that arise generally when PCA is applied to spatial data, implying that the patterns do not necessarily reflect specific migration events. Our findings aid interpretation of PCA results and suggest how PCA can help correct for continuous population structure in association studies.

Multi Dimensional Scaling (MDS)

- PCA is a linear dimension reduction method: it uses linear combinations of the original variables that explain the maximum variation in the data.
- However, in some cases, linear transformations of the variables may not work well.
- This is especially the case, if the data truly belongs on a nonlinear space (called a *manifold*).



Multi Dimensional Scaling (MDS)

- In PCA, similarities between variables are defined based on covariance.
- Covariances are really inner products, which correspond to the Euclidean distance.
- One way to make dimension reduction more flexible is to use a more general similarity and/or "distance".
- MDS directly works with a distance/dissimilariy matrix of observations.
- It then finds a lower dimensional representation (often 2-3D) to preserve the pairwise distances as well as possible. For example, MDS for 2 coordinates optimizes the object locations for a 2D scatterplot.

MDS: Some details

- Let D be a (arbitrary) dissimilarity matrix for objects $1, \ldots, n$. So, the dissimilarity between individuals i and i' is $d_{ii'}$.
- The goal of MDS is to find a representation of the data in low-dimensions (usually $\mathbb{R}^2$ or $\mathbb{R}^3$) so that the distances are preserved
- Formally, we want to find coordinates $z_i \in \mathbb{R}^m$ (i.e. vectors of length d) such that $\|z_i - z_{i'}\| \approx d_{ii'}$ for all i, i'
- More formally, MDS tries to find n vectors of length m, $z_i, i = 1, \ldots, n$, that minimize the following *stress function*:

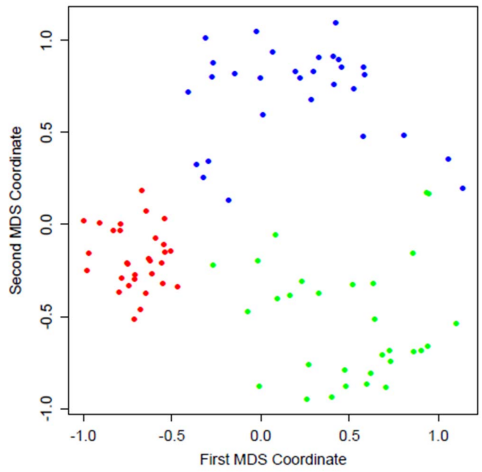
$$S_M(z_1, z_2, ..., z_n) = \sum_{i \neq i'} (d_{ii'} - \|z_i - z_{i'}\|)^2$$

MDS: Some comments

- Note that the only thing needed for the MDS is the dissimilarity matrix D. In general, different dissimilarity measures can give different MDS solutions
- The "classical" MDS uses the Euclidean distance, which gives results somewhat similar to PCA (though the objective is different). Classical MDS is also referred to as Principal Coordinate Analysis
  - PCA: finds a lower-dimensional representation that maximizes the variance
  - MDS: finds a lower-dimensional representation that best preserves the distances between the points
- We can use any other objective function or distance; this defines the general class of metric MDS
- We can also use rankings instead of distances which gives rise to *non-metric* MDS

MDS Applied to the Half-Sphere Data

In the new space, the points are well separated from each other:



MDS can be very useful if we believe nonlinear distances represent the data better, for instance in ecology, where we can incorporate the phylogenetic tree to define more informative distances

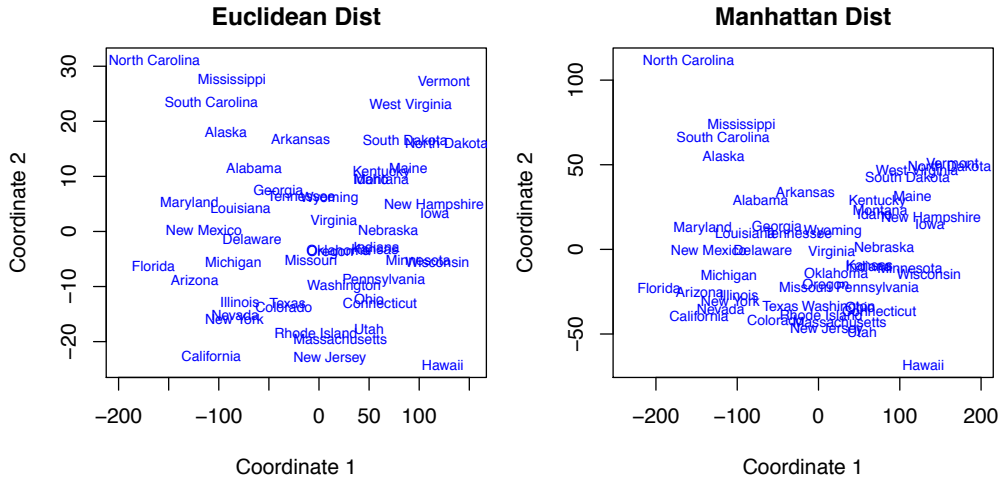
MDS for the `USArrests` Data

- Recall that MDS only uses a dissimilarity matrix, so given the data, we need to create one.
- The function `daisy()` in the `R`-package `cluster` has a couple of options to get a distance/dissimilarity matrix.
- The default is Euclidean (or ℓ_2) distance:
  $d(i,j) = \sqrt{\sum_{k=1}^{p} (X_{ik} - X_{jk})^2}$
- Can also use the "Manhattan" (or ℓ_1) distance:
  $d(i,j) = \sum_{k=1}^{p} |X_{ik} - X_{jk}|$.
- The function `cmdscale()` gives the "classic" MDS; `isoMDS()` gives a non-metric (rank-based) MDS

MDS for the `USArrests` Data

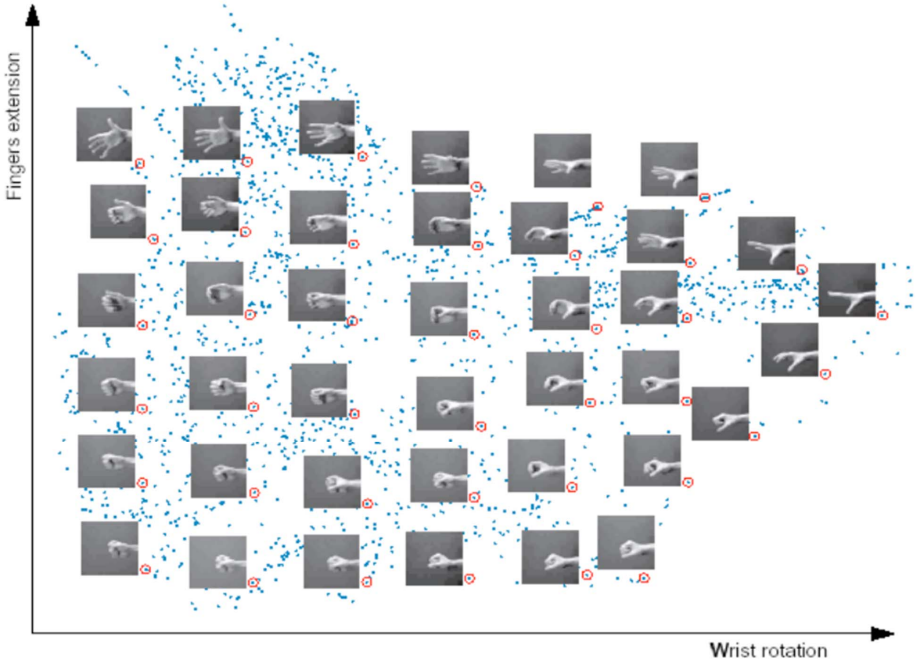
```
> dist.1=daisy(USArrests); mds.1=cmdscale(dist.1)
> dist.2=daisy(USArrests, metric="manhattan"); mds.2=cmdscale(dist.2)
>
> x=mds.1[,1]
> y=mds.1[,2]
>
> plot(x, y, xlab = "Coordinate 1", ylab = "Coordinate 2",
xlim = range(x)*1.2, type = "n", main='Euclidean Dist')
> text(x, y, labels = rownames(USArrests), col='blue', cex=0.7)
```

MDS for the `USArrests` Data



`isoMDS` gives slightly better results in this case!

An Interesting Application



From *Tenenbaum et. al.*

Remarks

- MDS belongs to the class of nonlinear dimension reduction techniques.
- A number of other methods for linear and nonlinear dimension reduction have been proposed.
- The main challenge in unsupervised learning is that we don't know whether a given methods has performed well in our dataset. This becomes even more challenging in high dimensional settings.
- More importantly, even if we can validate the performance of the method in our dataset, there is no guarantee that the method would work on similar datasets, or on subsequent data gathered for our study.
- Therefore, it is important to be cautious about interpreting the results of unsupervised learning; it is often easy to get excited with some aspects of the study!

HIGH-DIMENSIONAL OMICS DATA: Cluster Analysis - I

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Dimension Reduction vs Cluster Analsyis

- Dimension reduction methods find a low-dimensional representation of the observations that contain the good fraction of information in the original data
  - PCA tries to maximize the variance
  - MDS tries to preserve the distances among observations
- Clustering looks to find homogeneous subgroups among the observations

Cluster Analysis

Cluster analysis is one of the most-widely used techniques in analysis of omics data.

_computational BIOLOGY

PRIMER

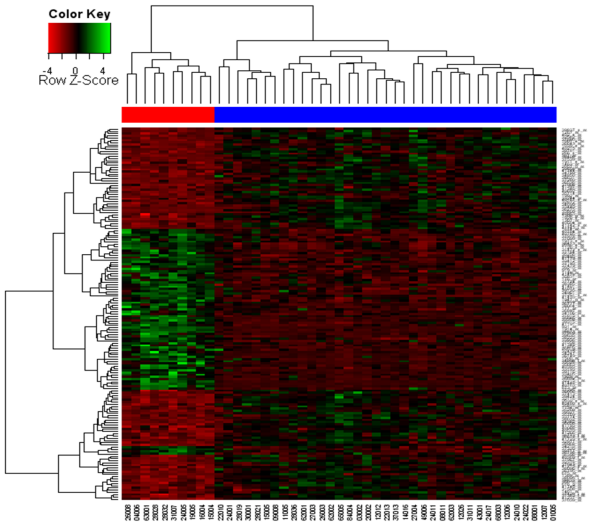
How does gene expression clustering work?

Patrik D'haeseleer

Clustering is often one of the first steps in gene expression analysis. How do clustering algorithms work, which ones should we use and what can we expect from them?

Cluster Analysis

Almost all papers on omics research, have a *heatmap*, which often includes a clustering of genes/samples.

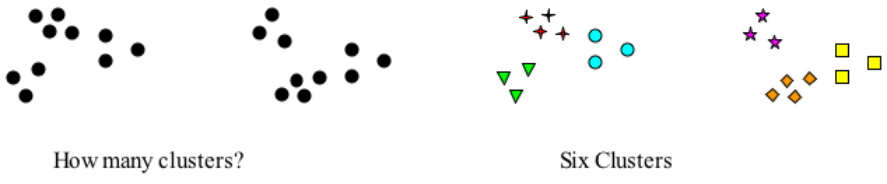


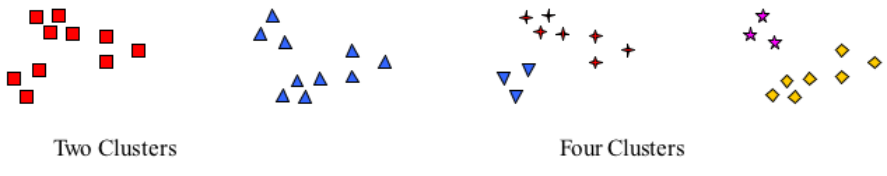
Objective

- Grouping objects into **meaningful** subsets or **clusters**, such that objects within each cluster are more **similar** to one another than objects in other clusters.
- Need to define what a "meaningful" cluster is
- Need to define what we mean by "similarity"
- Can cluster **observations** or **features**:
  - **observations**: clustering cancer samples to find cancer sub-types
  - **features**: clustering genes based on similar functions (pathways)
  - Clustering features is similar to clustering observations (work with the transpose of the data matrix)

Defining **meaningful** clusters

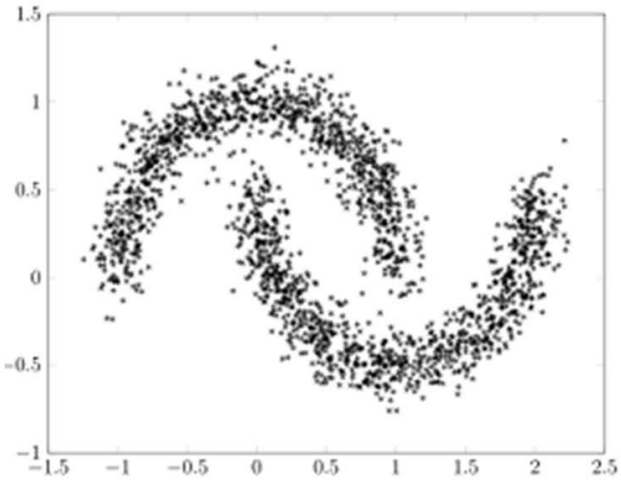
How many clusters?





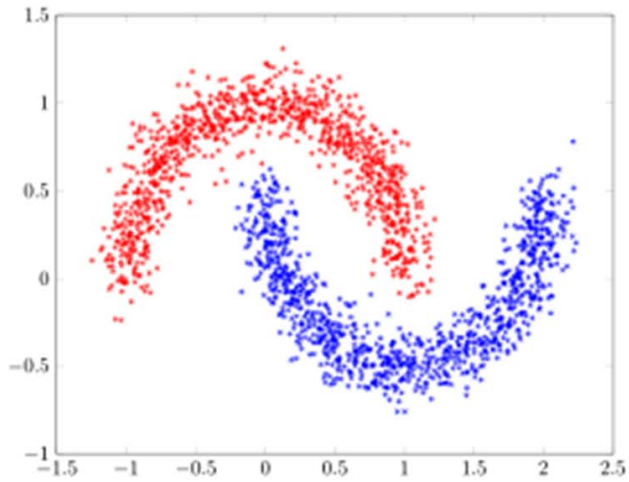
Defining **meaningful** clusters

Which points are more similar?



Defining **meaningful** clusters

Which points are more similar?

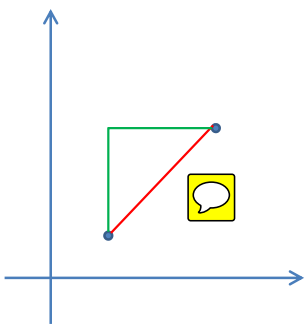


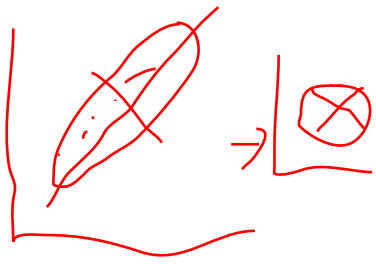
Similarity/Dissimilarity Measures

- An $n \times n$ matrix, calculated from the data matrix $\mathbf{X}$
- Similarity measure:
  - A numerical measure $s(i,j)$ that indicates how **similar** two objects are (high $s \equiv$ high similarity). Similarity is often normalized to be in the range $[0,1]$.
  - Properties: $s(i,j) \geq 0$ and $s(i,j) = s(j,i)$
- Dissimilarity measure:
  - A numerical measure $d(i,j)$ that indicates how **different** two objects are (lower $d \equiv$ high similarity). Unlike similarity, the upper bound may vary
  - Properties: $d(i,j) \geq 0$ and $d(i,j) = d(j,i)$
  - Any **distance** naturally defines a dissimilarity measure

Common Dissimilarity Measures

- **Euclidean** distance: $d(i,j) = \sqrt{\sum_{k=1}^{p} (X_{ik} - X_{jk})^2}$
- **Manhattan** distance: $d(i,j) = \sum_{k=1}^{p} |X_{ik} - X_{jk}|$
- **Mahalanobis** distance: For p-dimensional vectors X_i and X_j, $d(i,j) = (X_i - X_j)^{\mathsf{T}} \Sigma^{-1} (X_i - X_j)$, where Σ is the covariance matrix of X_k's





Common Similarity Measures

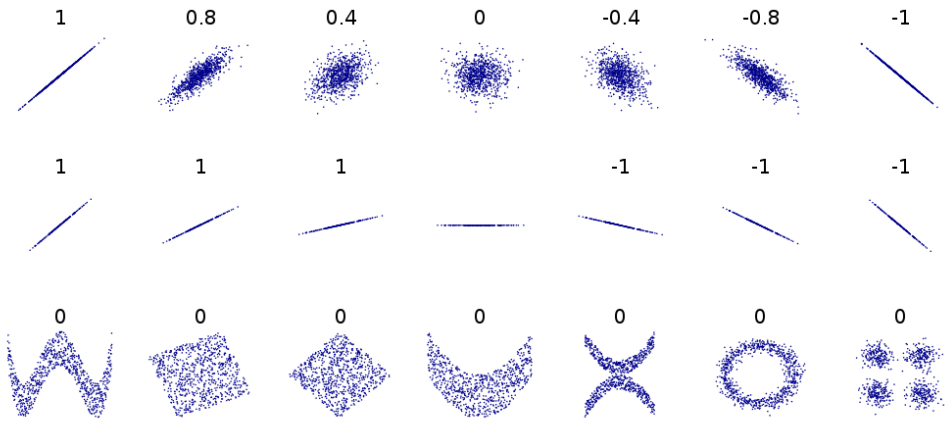
- **Correlation** coefficient
  - *Pearson* correlation (sample version):

$$r(X_i, X_j) = \frac{\sum_k (X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j)}{\sqrt{\sum_k (X_{ik} - \bar{X}_i)^2 \sum_k (X_{jk} - \bar{X}_j)^2}}$$

    This is the "usual" correlation coefficient, and measures the **linear** association between X_i and X_j
  - *Spearman* correlation: Same as Pearson correlation, but applied to **ranked** observations.
    Corresponds to an **increasing monotonic trend** between X_i and X_j (more appropriate for non-Gaussian observations)
  - *Kendall*'s τ: uses directly rankings among pairs of observations
- **Cosine** measure: $\cos(X_i, X_j) = \frac{X_i \cdot X_j}{\|X_i\| \|X_j\|} = \frac{\sum_k X_{ik} X_{jk}}{\sqrt{\sum_k X_{ik}^2 \sum_k X_{jk}^2}}$

  Measures the angle between two vectors, which determines how much they align.

Correlation Coefficient



Correlation Coefficient

Which data set has higher Pearson correlation?

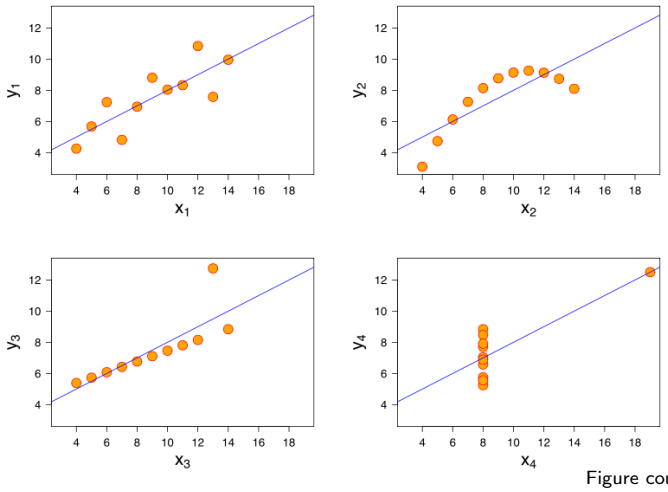


Figure courtesy of Wikipedia

In all of the above cases, $\mathbf{r = 0.816}$.

Methods of Hierarchical Clustering

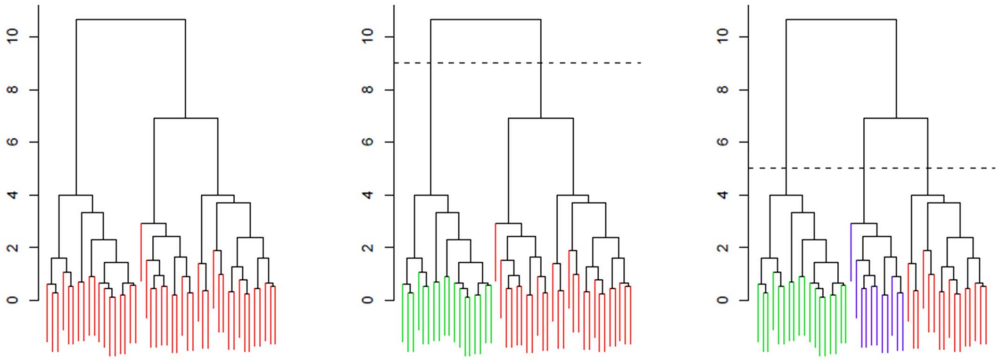
Hierarchical clustering results in a sequence of solutions (nested clusters), organized in a hierarchical tree structure, called the **dendrogram**

- **Bottom-Up** or Agglomerative: Start from n individual clusters, and group them together into using a measure of similarity.
  - Start from n individual clusters
  - At each step, merge the closest pair of clusters until all objects form a single cluster
- **Top-Down** or Divisive: Start from one cluster containing all objects, and break them down using a measure of distance
  - Start from 1 cluster
  - At each step, split the most heterogenous cluster until every cluster has only one member

We will focus on Bottom-Up methods.

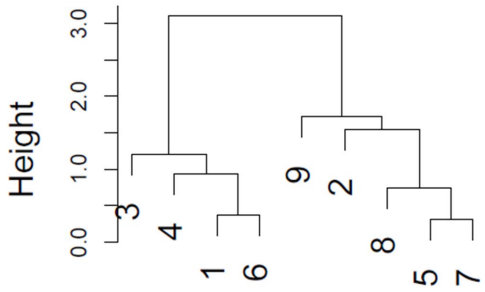
Interpreting the Dendrogram

How many clusters?



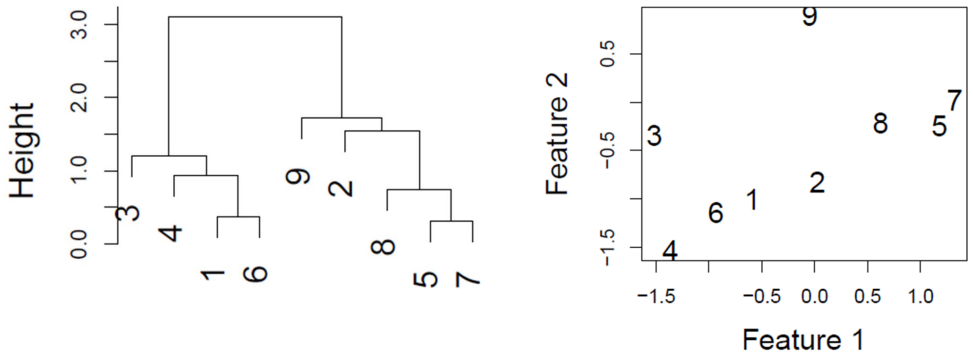
Interpreting the Dendrogram

Which points are clustered together?

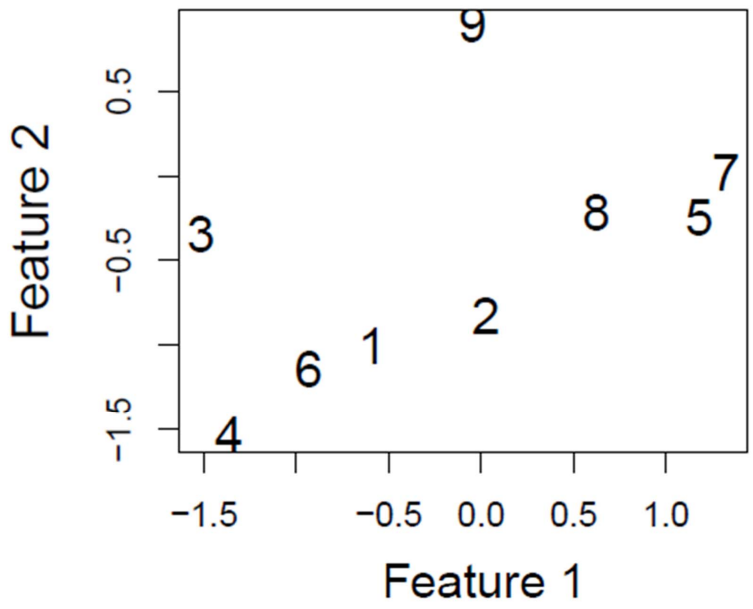


Interpreting the Dendrogram

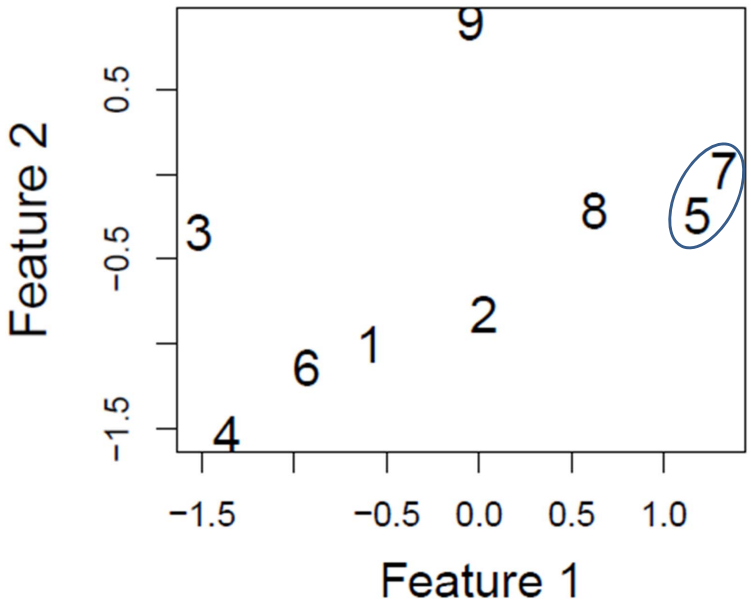
Which points are clustered together?



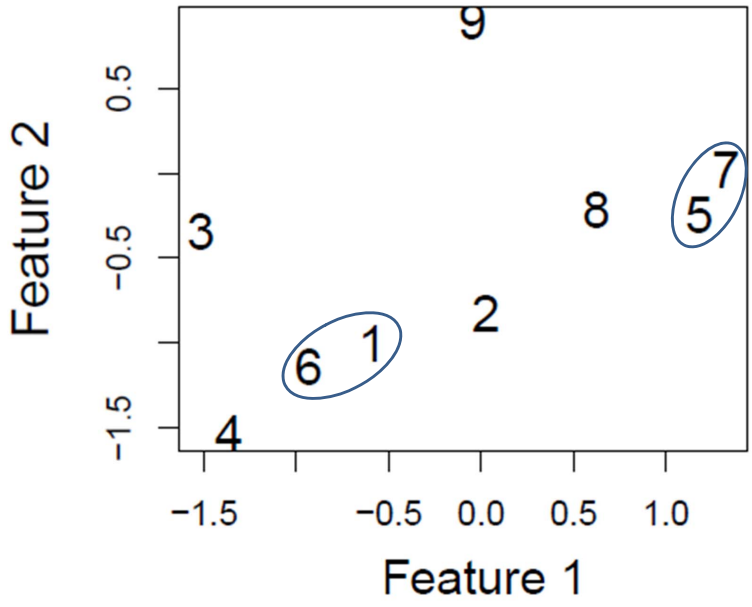
A closer look



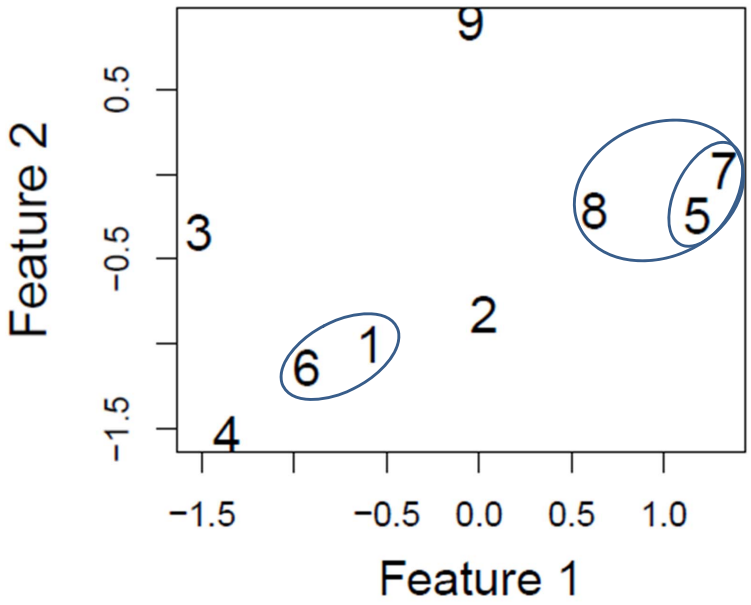
A closer look



A closer look



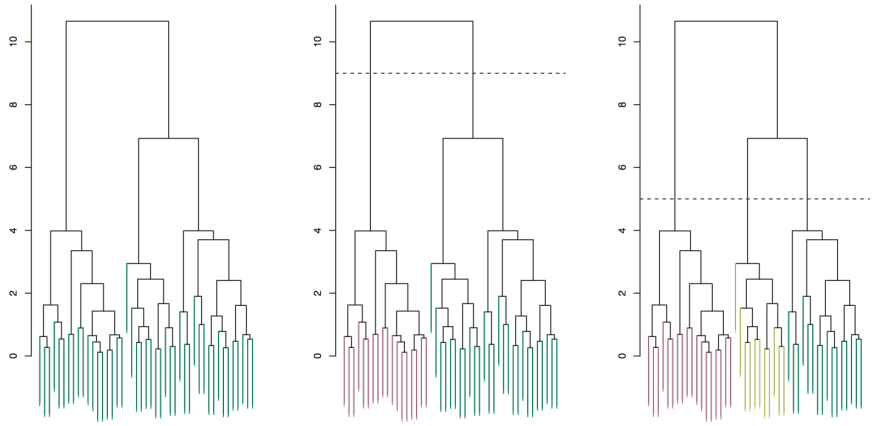
A closer look



Interpreting the Dendrogram

- At the bottom of the tree, there is a leaf for each observation
- As we move up the tree, some leaves begin to fuse into branches: these are observations that are similar to each other
- The earlier (lower in the tree) fusions occur, the more similar the groups of observations are to each other
- Observations that fuse later (near the top of the tree) can be quite different
- The height of the point in the tree where branches containing two observations are first fused, as measured on the *vertical axis*, indicates how different they are from each other
- However, we cannot draw conclusions about the similarity of two observations based on their proximity along the *horizontal axis*

Interpreting the Dendrogram



Measures of Inter-Cluster similarity

- Single linkage (min)
- Complete linkage (max)
- Average linkage
- Distance between centroids
- Ward's method

Single Linkage

- At each step, the minimum distance between points in two clusters is used to determine which two clusters should be merged
- Can handle **diverse shapes**
- **Very sensitive to outliers/noise**
- In practice, often results in **unbalance clusters**; may break down the clusters by "chaining" their members
- Can result in extended, trailing clusters in which observations are fused one-at-a-time

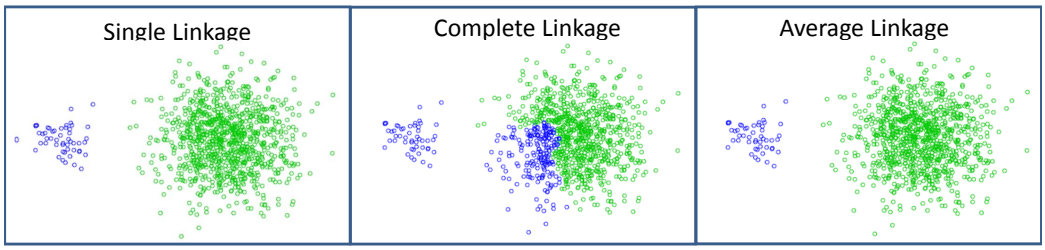
Complete Linkage

- At each step, the maximum distance between points in two clusters is used to determine which two clusters should be merged
- Often gives **comparable cluster sizes**
- Less sensitive to outliers
- Works better with **spherical distributions**

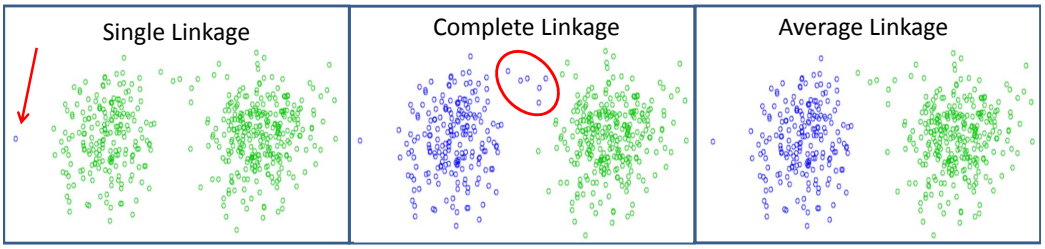
Average Linkage

- At each step, the average distance between points in two clusters is used to determine which two clusters should be merged
- A compromise between single and complete linkage
- Less sensitive to outliers
- Works better with **spherical distributions**

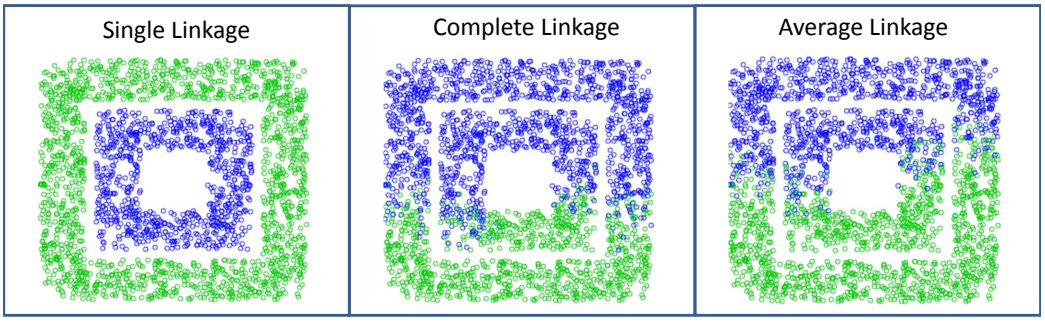
Unbalanced Clusters



Outliers



Non-Spherical Distributions



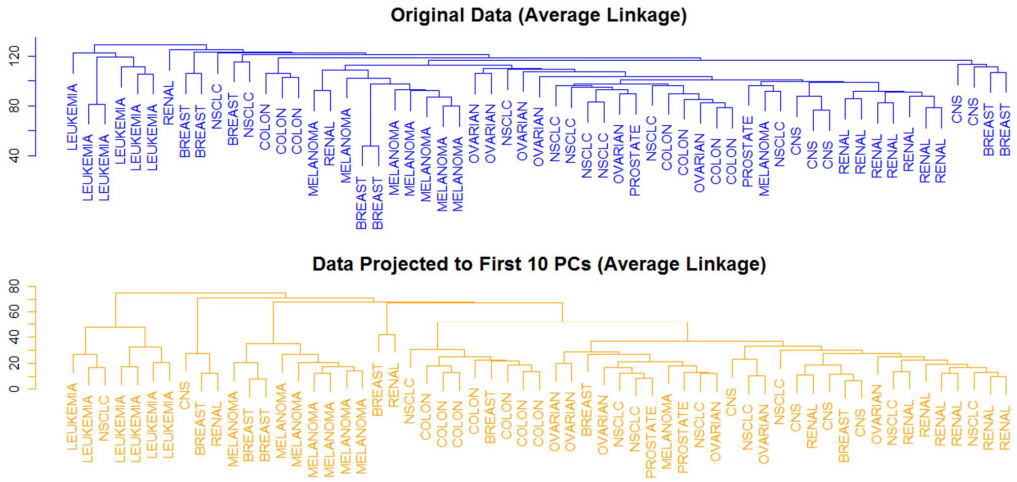
Example

- Clustering analysis on NCI60 data
- The dataset includes gene expression data for 6830 genes from 64 cancer samples
- Data can be downloaded form http://www-stat.stanford.edu/~tibs/ElemStatLearn/
- Missing values have been imputed
- Cluster the samples using different hierarchical clustering methods

Example, contd.

```
## Analysis NCI60 data
mydata <- read.table('nci.data')
dim(mydata)
mydata <- t(mydata)
mydata <- scale(mydata, center=F, scale=T)

labs <- read.table('nci.info', skip=14)
labs <- as.vector(as.matrix(labs))
table(labs)

## Define the distance
mydist <- dist(mydata)

## Plot the results
par(mfrow=c(1,3))
plot(hclust(mydist), labels=labs, col="green", main="Complete Linkage",
    hang=0.2, xlab="", sub="", ylab="", cex.main=1.5, cex.lab=0.6)
plot(hclust(mydist, method="average"), labels=labs, col="orange", hang=0.2,
main="Average Linkage", xlab="", sub="", ylab="", cex.main=1.5, cex.lab=0.8)
plot(hclust(mydist, method="single"), labels=labs, col="blue", hang=0.2,
main="Single Linkage", xlab="", sub="", ylab="", cex.main=1.5, cex.lab=0.8)
```

Hierarchical Clustering for NCI60 Data

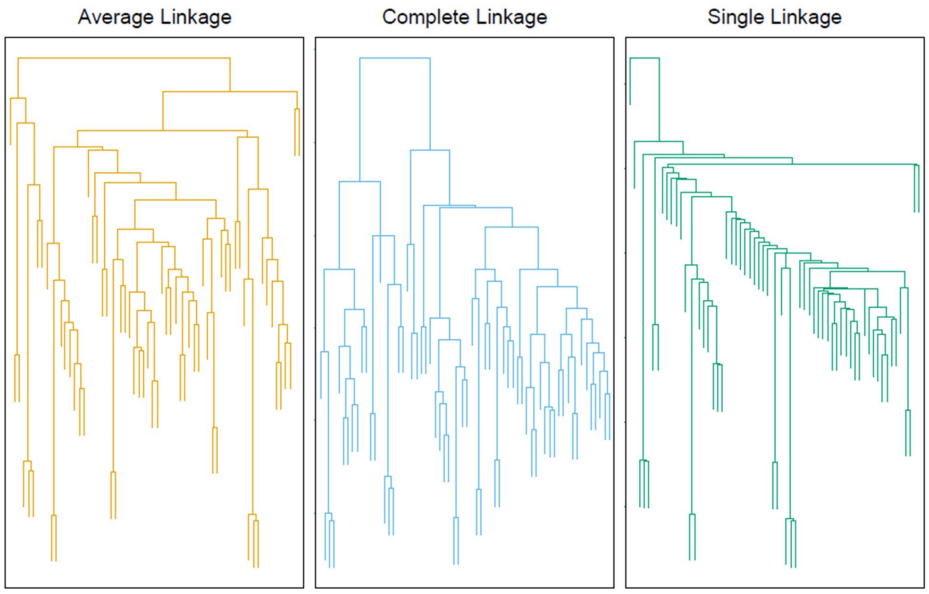


We will talk more about the second approach in the next lecture...

Exercise

- The analysis here is performed using **Euclidean distance**.
  Repeat the analysis using Spearman correlations, and compare the results.
  Hint: A good starting point is: `as.dist(1-cor(t(x)))`

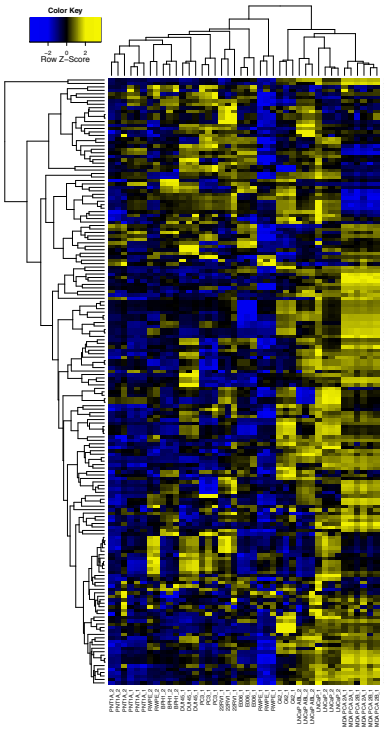
Which Linkage Function?



Properties of Hierarchical Clustering

- Advantages: Gives a family of possible solutions; computationally fast
- Disadvantages: No optimization criterion; final solution chosen by the data analyst; different merging (splitting) criteria give different solutions

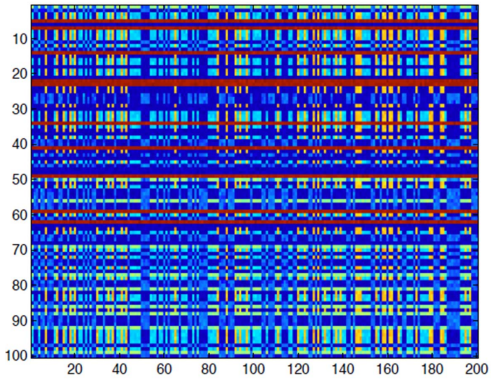
Bi-Clustering

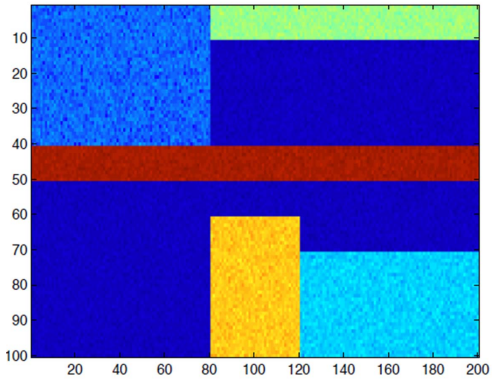


- Clustering variables and samples *simultaneously*
- Useful for e.g. finding subgroups of genes with similar activity in subclasses of cancer patients
- *Heatmap* of metabolite abundances in cancer cell lines

Bi-Clustering: Main Idea

Find a rearrangement of rows and columns that gives meaningful partitions





HIGH-DIMENSIONAL OMICS DATA: Cluster Analysis - II

Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Hierarchical Clustering vs Partition-Based methods

- In hierarchical clustering, a sequence of possible clusterings is generated
- The analyst then decides on the number of clusters (by deciding where to cut the dendrogram)
- In partition-based methods, data is partitioned into a number of *a priori* given clusters
- No need for intervention by analyst; however, number of clusters needs to be specified.

K-Means Clustering

- The most popular partition-based method is K-means clustering
- Motivated by a simple and intuitive mathematical problem
- Uses Euclidean distance between points
- Computationally efficient, and can be applied to datasets with a large number of samples (variables)
- No hierarchy among clusters, if K is changed, the cluster memberships will also change 

K-Means Clustering: Problem Formulation

- Let $C_1, \ldots, C_K$ be any partition of the samples
- We want to find the an optimal partition, such that the *within-cluster variation* is minimized:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} WCV(C_k)$$

- There are different ways to define $WCV(C_k)$ but squared Euclidean distance is often used in K-means :

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (X_{ij} - X_{i'j})^2$$

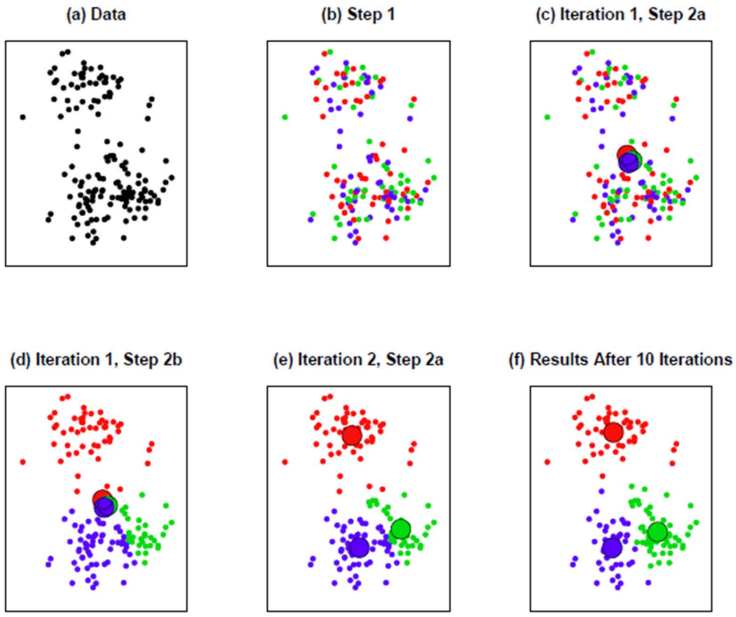
- So, we need to solve:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \left\{ \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (X_{ij} - X_{i'j})^2 \right\}$$

K-Means Clustering: Solution

- Although the optimization problem in *K*-means is simple and intuitive, finding exact solutions (global minimum) is not tractable
- However, we can efficiently find good approximate solutions for this problem (local minimum) using the following algorithm:
  1. Randomly assign each observation to one of K clusters.
  2. Iterate until the cluster assignments don't change:
     (a) For each of the K clusters, compute the cluster **centroid**, i.e. the mean of the observations assigned to the each cluster. This is a vector of length p (for p features).
     (b) Assign each observation to the cluster with closest centroid (based on Euclidean distance).

K-Means Clustering: An Example with Three Clusters

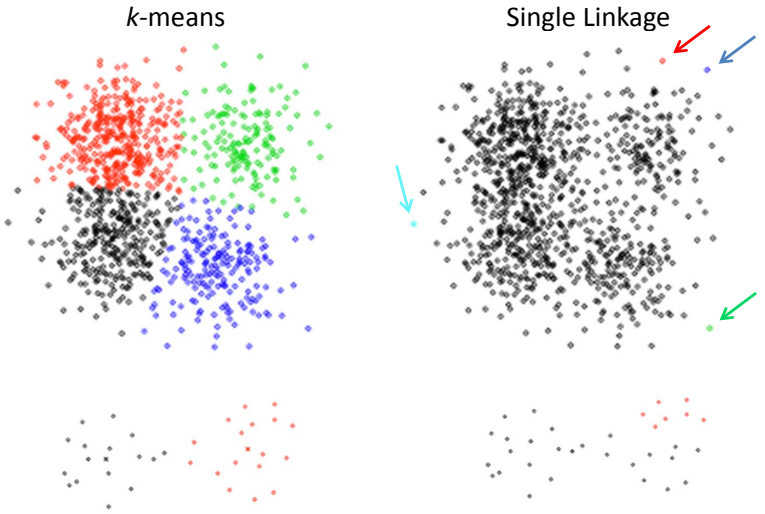


Properties of *K*-Means

- In *K*-means there is no hierarchy among clusters, if *K* is changed, the cluster memberships will also change
- *K*-means works best with compact spherical clusters with comparable number of members
- Does not work well when cluster sizes are different
- Results depend on the initial cluster assignment; may not get the best solution
- May result in empty clusters
- May result in artificially small clusters (a possible solution is to eliminate outliers)

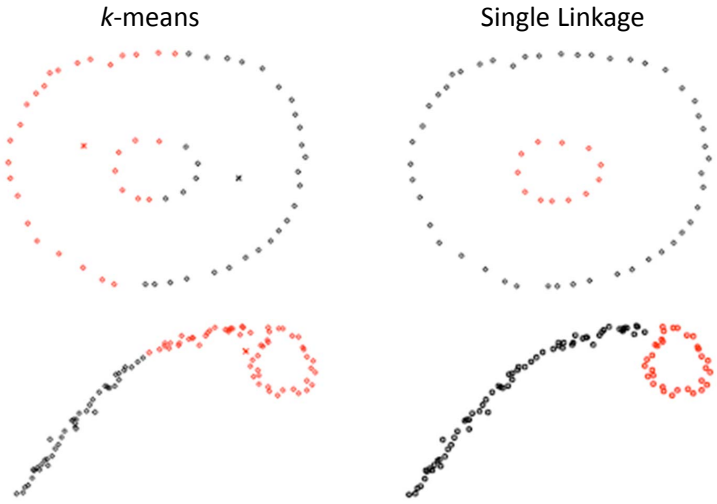
K-Means Performance

Examples, where *K*-means works well



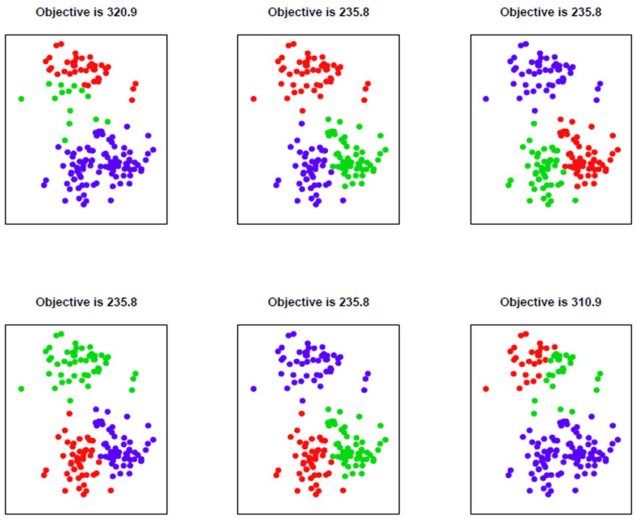
K-Means Performance

Examples, where *K*-means does not work well



Improving *K*-Means

A practical solution for obtaining better solutions for *K*-means is to run the algorithm with a number of random starting points, and then choose the solution with lowest objective function.



Evaluating the Quality of a Clustering

- **Cluster homogeneity**
  - Within sum of squares (WSS)
  - For each object the "error" is the distance to its cluster centroid:

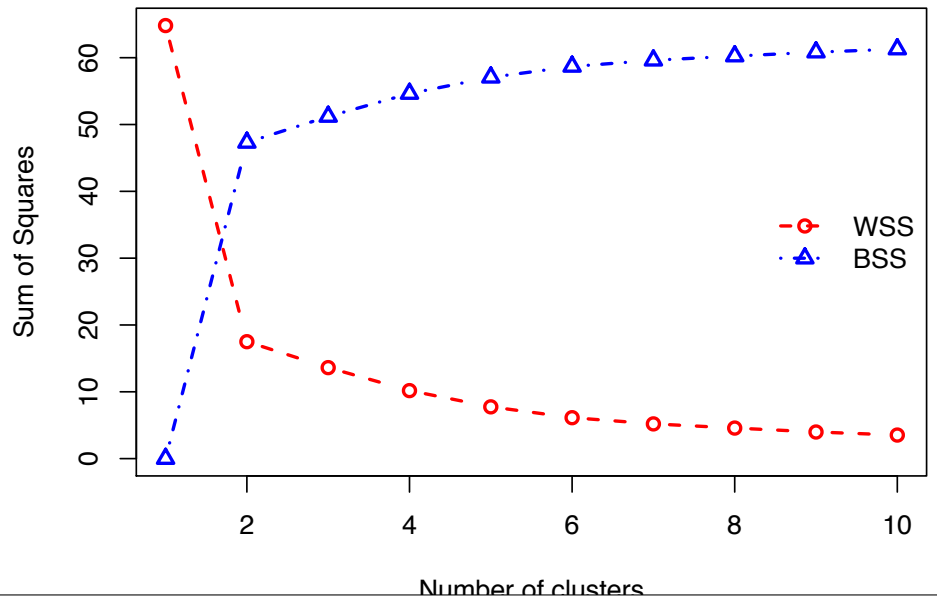
$$WSS = \sum_{k=1}^{K} \sum_{i \in C_k} d^2(m_k; X_i)$$

  - Given two clustering solutions, the one with smaller WSS is preferred
  - WSS usually decreases as K increases
- **Cluster separation**
  - Between sum of squares (BSS)
  - For each cluster the "error" is the distance between the cluster centroid and the *grand mean*:

$$BSS = \sum_{k=1}^{K} d^2(m_k; m)$$

BSS vs WSS

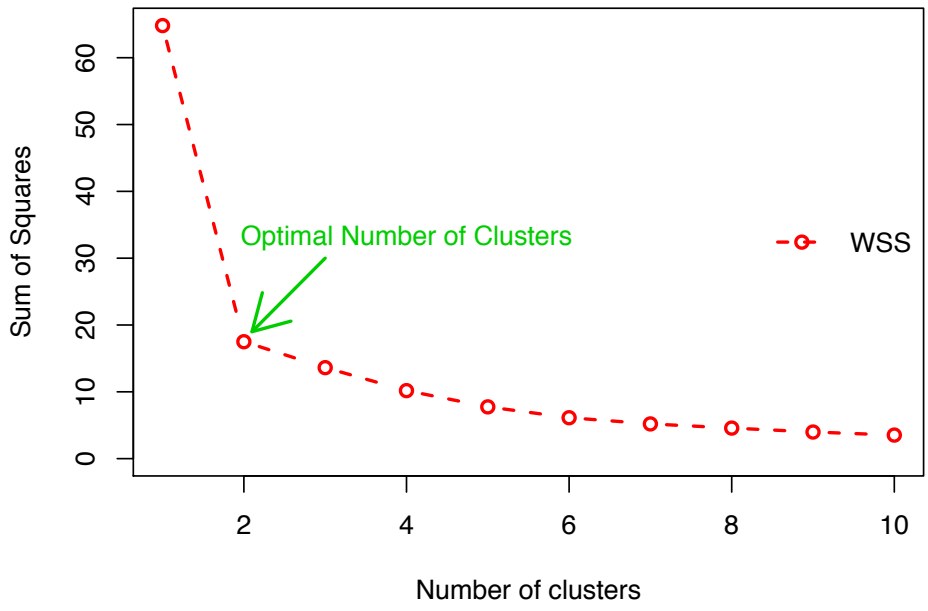
For K-means , $d^2(X_i; m_k) = \sum_{j=1}^{p} (X_{ij} - m_{kj})^2$ (Euclidean dist.)



Choosing K (Number of Clusters)

- Look at the values of within cluster dissimilarity (e.g. *WSS*) for different K.
- In general, *WSS* decreases as K increases.
- Suppose there is K^* true clusters
- If $K < K^*$, increasing K will improve *WSS* significantly
- If $K > K^*$, increasing K will improve *WSS* slightly
- Therefore, need to look for an **elbow** in the plot of *WSS* over K

WSS for K-Means



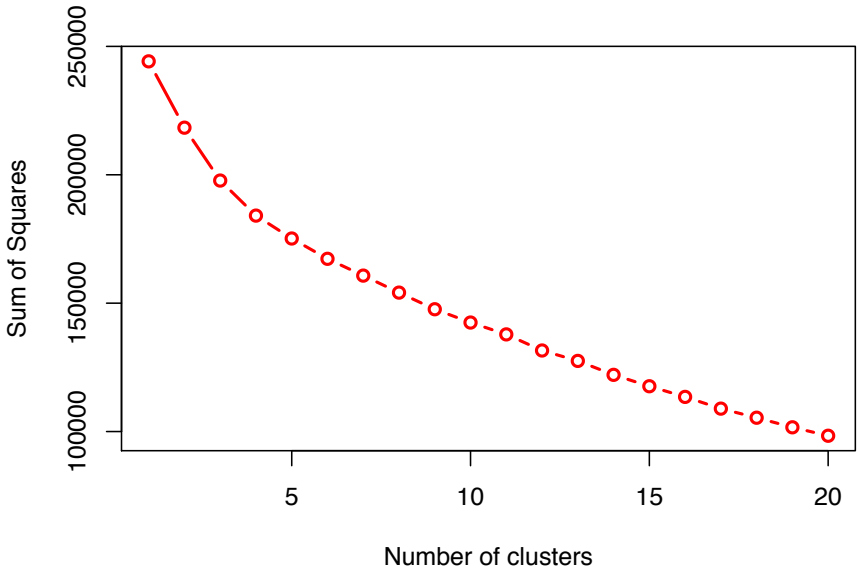
Other Methods for Choosing K (Number of Clusters)

- Gap Statistics (Tibshirani et al, 2001): Compares the curve of $\log WSS$ to the curve obtained from data uniformly distributed (i.e. no clusters), and estimates the optimal number of clusters to be the place where the gap between the two curves is largest.
- The Silhouette Coefficient (Rousseeuw, 1986): Combines homogeneity and separation, can find the best number of clusters by minimizing this coefficient over range of values of K.

Example: K-Means Clustering on NCI60 Data

```
## K-means clustering for the NCI60 data
mydata <- read.table('nci.data')
mydata <- t(mydata)
mydata <- scale(mydata, center=F, scale=T)

Kmax <- 20              #maximum number of clusters
Nstrt <- 25             #Number of starting partitions
wss <- numeric(K)
bss <- numeric(K)
tss <- numeric(K)
for(k in 1:Kmax){
  cl <- kmeans(dat, k, nstart=Nstrt)
  tss[k] <- cl$totss
  wss[k] <- cl$tot.withinss
  bss[k] <- cl$betweenss
}

## Plot the results
par(mar=c(4.5,4.5,1,1))
plot(1:Kmax, wss, type='b', lty=1, lwd=2, col=2,
    ylab='Sum of Squares', xlab='Number of clusters')
```

How Many Clusters?

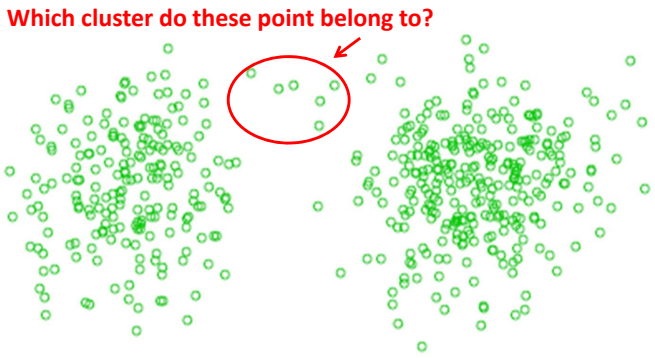


Exercise

- Repeat the analysis without normalizing the data. What do you see?
- Repeat the clustering with different values of `nstart` $\in \{1, 5, 10, 50\}$
- How do the results change?
- What is the optimal number of clusters?
- Compare the optimal number of clusters with what you observed using hierarchical clustering.

Model-Based Clustering

- Hierarchical or K-means clustering assign each data point to one of the K clusters
- This makes these methods sensitive to outliers and the compactness of the clusters
- In many applications, some points fall between 2 clusters
- Need to allow for soft cluster memberships



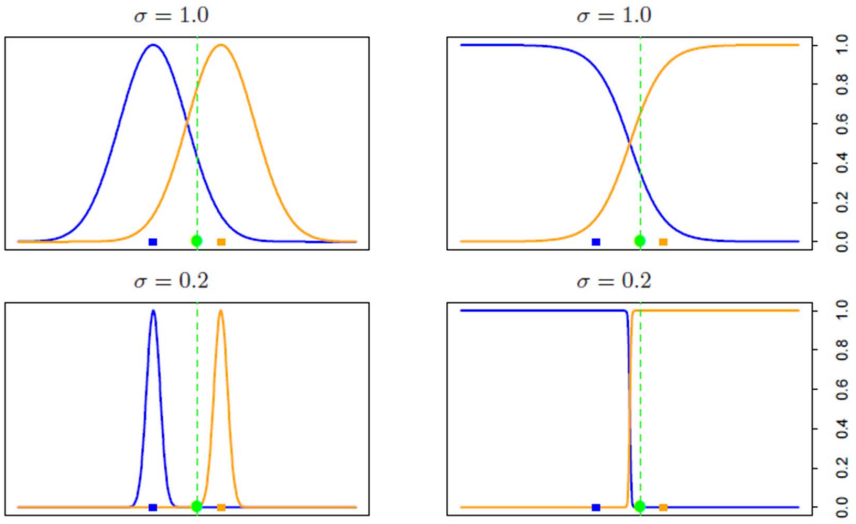
Model-Based Clustering

- A popular approach for model-based clustering is Gaussian mixture models
- It is assumed that data is generated from

$$\sum_{k=1}^{K} \pi_k N(\mu_k, \Sigma_k)$$

- Here π_k gives the probability that data is generated from kth distribution
- Often it is assumed that $\Sigma_k = \sigma_k^2 I$
- As σ_k^2 becomes smaller, this becomes more similar to K-means clustering
- Function `Mclust` in the R-package `mclust` gives an implementation of this method

Model-Based Clustering



Gaussian Mixtures | Relative Densities

Spectral Clustering

- Traditional clustering methods (K-means, model-based etc) use spherical/elliptical metrics, and may not work with non-convex clusters
- Spectral clustering uses the top M eigenvectors of the affinity matrix (or similarity matrix)
- Can detect non-spherical clusters, or clusters with different shapes and sizes
- Can work with any similarity matrix, and is hence more flexible
- Similar ideas are used in the PageRank algorithm used by Google

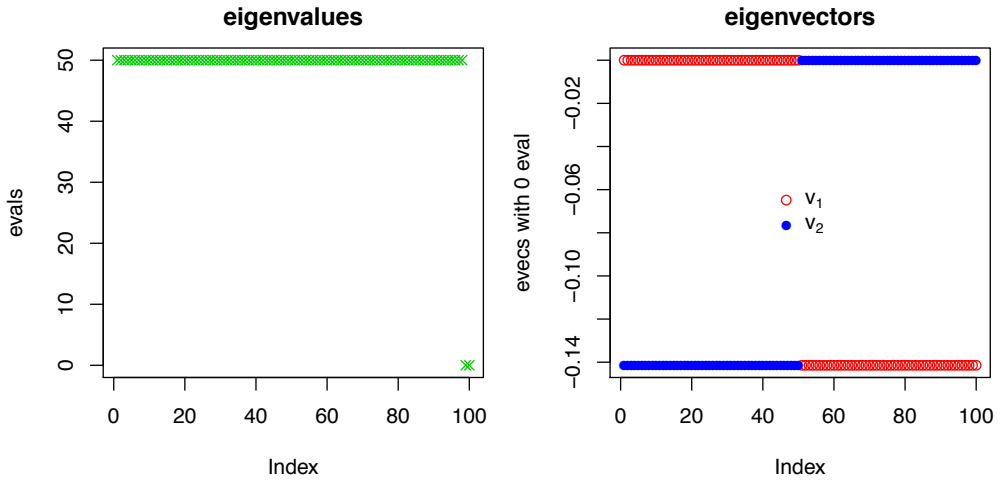
Spectral Clustering: Details

- Start with a $n \times n$ matrix of pairwise similarities $s_{ii'} \geq 0$
- Represent observations as an undirected similarity graph $G = (V, E)$:
  - verticies/nodes of G are observations;
  - edges of G show positive similarities;
  - in other words, the similarity matrix is the adjacency matrix of the graph;
  - ideally the similarity graphs should represent the local neighborhood relationships.

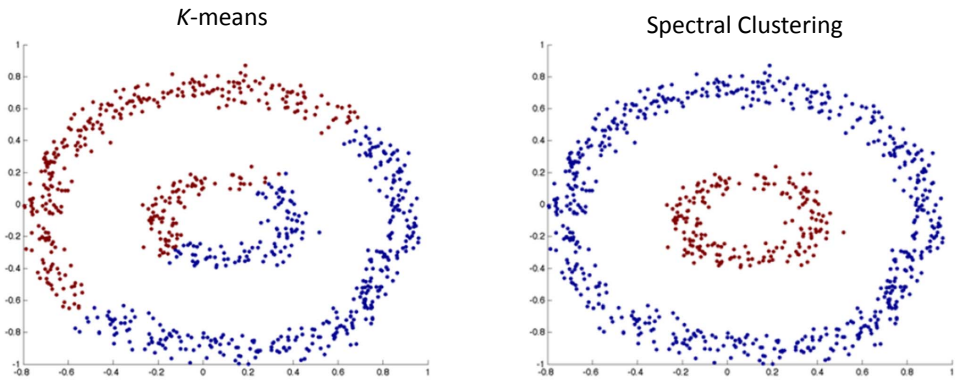
Spectral Clustering: Details

- In spectral clustering, the problem of finding clusters is translated to a graph partitioning problem:
  - want to find groups of nodes that are highly connected to each other, but less connected to other nodes;
  - let D be a diagonal matrix of degrees: $d_{ii} = \sum_{i' \neq i} s_{ii'}$
  - optimal graph partitions can be found using the eigenvectors of the graph Laplacian: $L = D - S$;
  - if the graph has e.g. K connected components, then L has (exactly) K eigenvectors $v_k, k = 1, \ldots, K$, with eigenvalue 0;
  - the non-zero values in each each v_k identify one of the connected components.
- In practice, we find *approximate* partitioning of the graph by selecting K eigenvectors, and dropping the nodes corresponding to "small" entries

Spectral Clustering



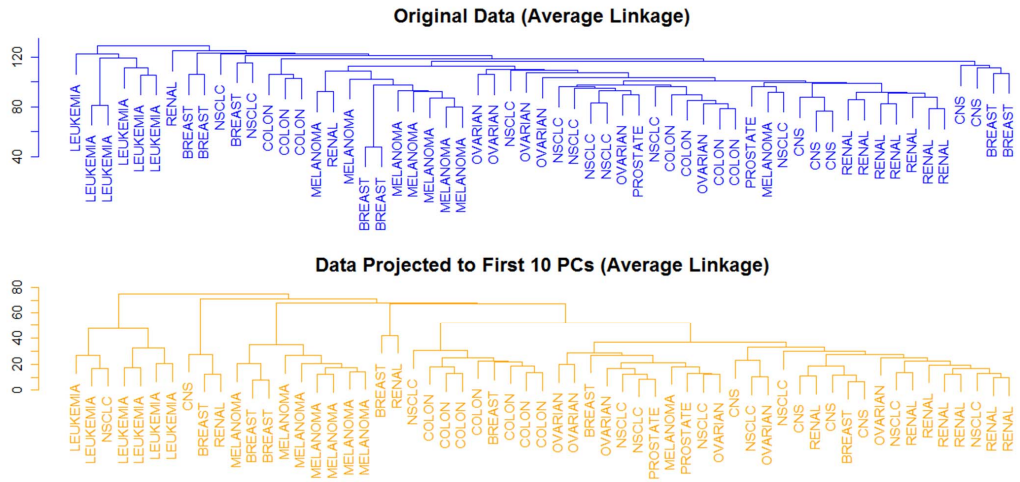
Spectral Clustering



Spectral clustering and Principal Components

- Consider a fully connected graph, with Euclidean distances between each pair of points
- Then, spectral clustering is equivalent to clustering the observations after projecting them into the PC-space
- Basically, in this case, the affinity matrix is the sample covariance matrix, and eigenvectors are PCs
- The type of similarity graph (connected or nearest neighbors) as well as number of eigenvectors are important choices in spectral clustering.

NCI60 Data, again



The second approach is basically a version of spectral clustering, using the correlation matrix

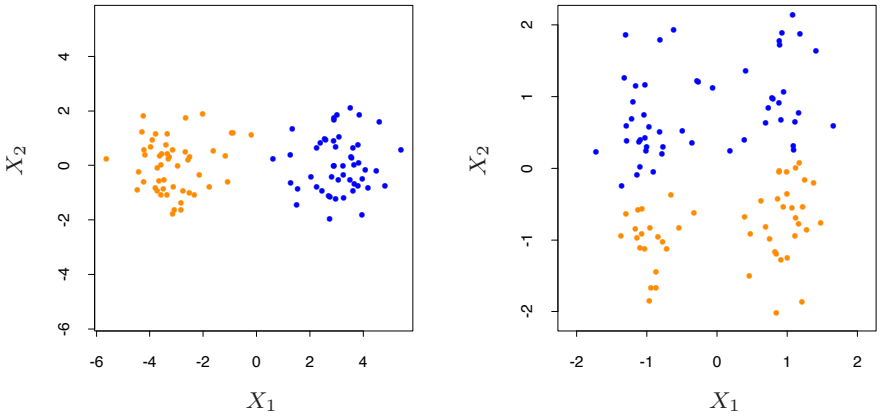
Caveats of Cluster Analysis (Dhaeseleer, 2005)

Although many clustering algorithms are available, this is still a challenging problem:

- There is no one-size-fits-all solution to clustering, or even a consensus of what a "good" clustering should look like.
- There is no single best criterion for obtaining a partition because no precise and workable definition of "cluster" exists.
- Clusters can be of any arbitrary shapes and sizes in a high dimensional data.
- Each clustering criterion imposes a certain structure on the data, and if the data happen to conform to the requirements of a particular criterion, the true clusters are recovered.
- It is very hard to evaluate how well a clustering algorithm performs on typical omics data sets.

Caveats of Cluster Analysis

- Also, note that centering and scaling will significantly affect the results of cluster analysis
  - Left: K means clustering on the original data;
  - Right: K means clustering on the scaled data.



Final Remarks on Clustering

- Clustering is a very useful for gaining insight in high dimensional omics data
- Different clustering results for the same data, using different methods/distance matrices
- No formal way to verify the results of clustering
- Determining number of clusters is a challenging problem
- Results of clustering may vary for normalized observations
- Best practice is to try different methods/distances and use the results that are more consistent across methods/distances
- Clustering is a data exploration step, and its results should be interpreted that way; cannot make strong claims...

HIGH-DIMENSIONAL OMICS DATA: High-Dimensional Hypothesis Testing

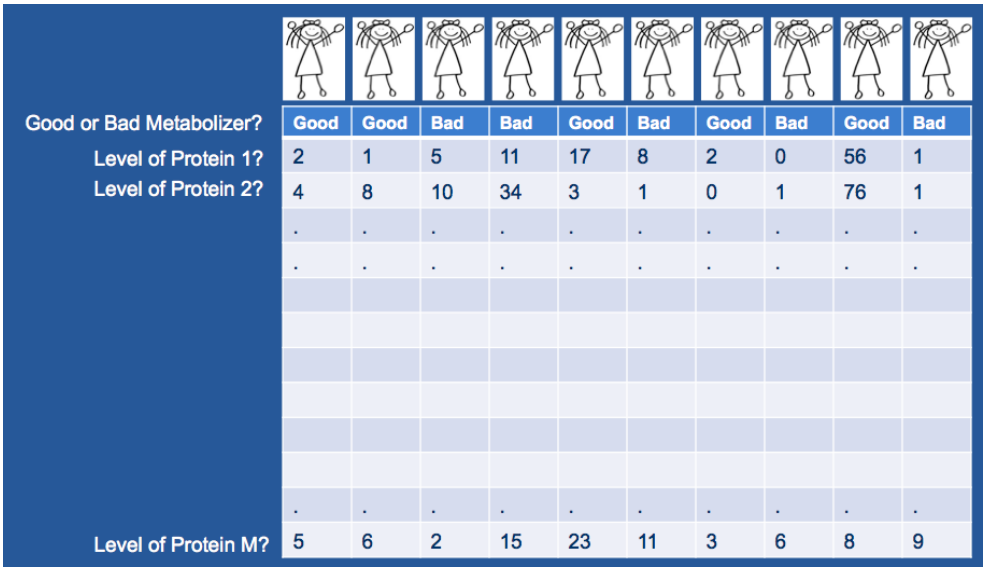
Ali Shojaie & Daniela Witten

July 21-23, 2014
Summer Institute for Statistical Genetics
University of Washington

Hypothesis Testing

- In this course so far, we have talked about methods for prediction in the high-dimensional setting.
- We now consider a seemingly much simpler topic: hypothesis testing.
- But hypothesis testing in high dimensions has its challenges!

Hypothesis Testing



Good or Bad Metabolizer?	Good	Good	Bad	Bad	Good	Bad	Good	Bad	Good	Bad
Level of Protein 1?	2	1	5	11	17	8	2	0	56	1
Level of Protein 2?	4	8	10	34	3	1	0	1	76	1
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
Level of Protein M?	5	6	2	15	23	11	3	6	8	9

Hypothesis Testing

- We have M features, each of which is measured in n observations.
- (In this lecture only, M is number of features, rather than p.)
- We also have a response vector of length n. Could be
  - blood pressure
  - tumor size
  - survival time
  - cancer subtype
- We wish to test the null hypothesis

$$H_{0j} : j\text{th feature is not associated with the response}$$

for $j = 1, \ldots, M$.

Testing One Hypothesis

Suppose we had only one hypothesis to test:

$$H_0 : \text{ feature is not associated with the response.}$$

The type of test that we use will depend on the type of response:

- binary response (e.g. cancer versus normal): two-sample t-test
- categorical response (e.g. cancer type 1 versus cancer type 2 versus cancer type 3): F-statistic for one-way ANOVA
- survival response (e.g. time to recurrence): score statistic for Cox proportional hazards model
- quantitative response (e.g. blood pressure): t-statistic for regression coefficient

Regardless of the response type, we get a test statistic and a p-value for the association between the feature and the response.

Quick Review: Test Statistics & P-Values

- Suppose we have measured the expression level of gene A in n patients, and want to test its association with the response, cancer versus normal.
- We compute a two-sample t-statistic quantifying its association with the response.
- Large (absolute) t-statistic: strong association with response.
- How big is big?
- The p-value is the probability of observing a t-statistic at least this large, under the null hypothesis of no association between the feature and the response.
- A p-value of 0.02 means that the probability of seeing such a strong association between the response and the feature by chance, under the null hypothesis, is 0.02.
- A small p-value indicates strong evidence for association: i.e. reject the null hypothesis.

Type I Error

- When we reject the null hypothesis if a p-value is below 0.05, we are controlling Type I Error at level $\alpha = 0.05$.
- Type I error occurs when we reject the null hypothesis when the null hypothesis actually holds.
- We are not required to control Type I error at level $\alpha = 0.05$. For instance, we could use $\alpha = 0.01$ instead. Then we'd reject the null hypothesis if the p-value is below 0.01.
- Physicists control Type I error at around $\alpha = 5 \times 10^{-7}$. (Then again, atoms are cheaper than patients and mice.)

Multiple Testing

- Now, instead of testing one feature's association with the response, suppose we test the association of M features with the response.
- We can compute a p-value for the association between each feature and the response.
- Probability that the jth p-value will be less than 0.05, assuming the null hypothesis holds for the jth feature, is 0.05.
- But the probability that at least one of the M p-values will be less than 0.05, assuming that all of the null hypotheses hold, is much greater than 0.05.
- In other words, we will erroneously reject the null hypothesis if we test a lot of hypotheses and reject the null hypothesis for any p-values less than the usual threshold, like 0.05.

Multiple Testing

- The probability of falsely rejecting the jth null hypothesis is α.
- The probability of falsely rejecting at least one of M null hypotheses is $1 - (1 - \alpha)^M$.
- Let $\alpha = 0.05$, and assume that the tests are all independent.
  - $M = 1$: probability of falsely rejecting at least one null hypothesis is 0.05.
  - $M = 2$: probability of falsely rejecting at least one null hypothesis is 0.0975.
  - $M = 10$: probability of falsely rejecting at least one null hypothesis is 0.40.
  - $M = 200$: probability of false rejecting at least one null hypothesis is 0.9999649.

Hypothesis Testing

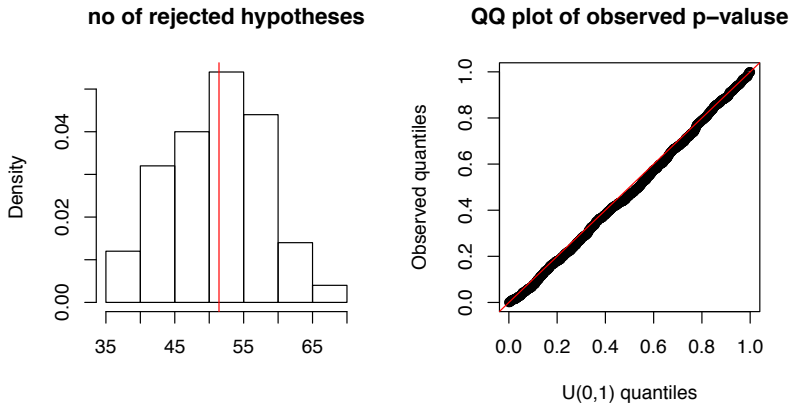
We will get a lot of false positives if we use 0.05 as a cut-off for rejecting the null hypothesis.

Protein	T-stat	P-Value
1	3.2	0.00137
2	-1.8	0.0718
3	5.8	6e-9
4	13.2	0
5	1.4	0.1615
6	-0.2	0.8414
.	.	.
.	.	.
.	.	.
M	4	6e-5

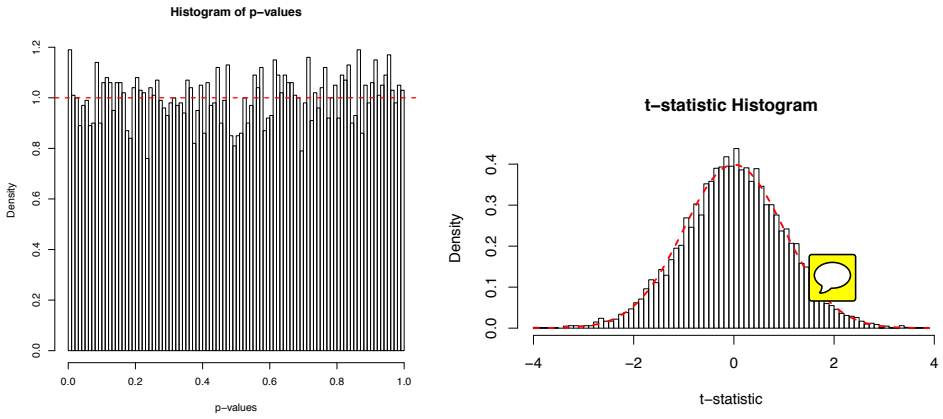
A Simple Illustration

- Consider 1000 hypotheses & 50 samples in two groups (25 treatment, 25 control) (e.g. mRNA expression for 1000 genes) from $N(0,1)$: nothing significant!
- 10 smallest p-values:

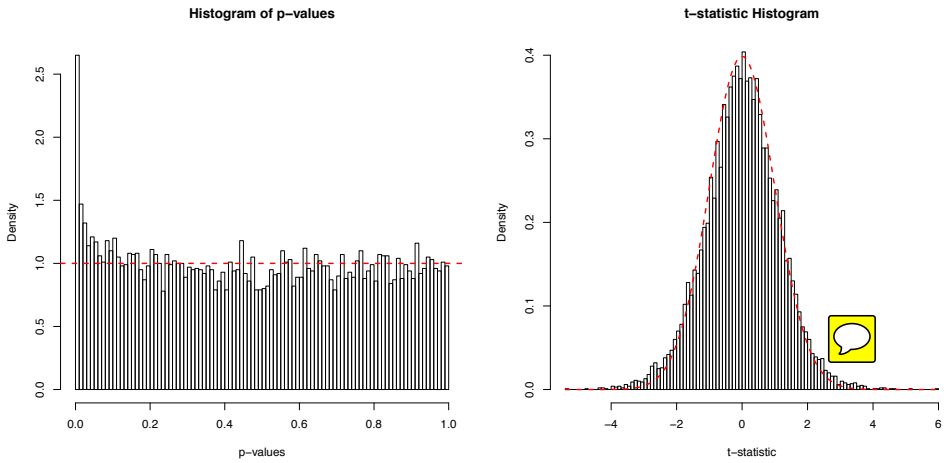
  0.0005, 0.0069, 0.0076, 0.0087, 0.0123, 0.0124, 0.0132, 0.0160, 0.0168, 0.0176
- All of these will be rejected by chance at the level of 5%!!



Example 1: All Null Hypotheses Hold



Example 2: Not All Null Hypotheses Hold



Bonferroni Correction

- Simple solution: A Bonferroni correction.
- Rather than rejecting the jth null hypothesis is less than 0.05, we reject the jth null hypothesis if the jth p-value is less than $0.05/M$.
- More generally, to control the overall Type I error – probability of falsely rejecting the null hypothesis – at level α, we must reject the jth null hypothesis if its p-value is below α/M.

Bonferroni Correction

Why does the Bonferroni correction control Type I Error?

- Let A denote the event that at least one null hypothesis is falsely rejected, and let A_j be the event that the jth null hypothesis is falsely rejected. So

$$P(A) = P(\bigcup_{j=1}^{M} A_j) \leq \sum_{j=1}^{M} P(A_j).$$

  This means that if we keep $P(A_j)$ below α/M, then $P(A)$ will be below α.
- And keeping $P(A_j)$ below α/M means that we reject the jth null hypothesis if the p-value is below α/M.

Bonferroni Bottom Line

- To summarize, to control the overall Type I error at level α, we reject the jth null hypothesis if its p-value is below α/M.
- So if we are testing each SNP for association with a phenotype in a GWAS with $M = 10^6$ SNPs, then we reject the jth null hypothesis if its p-value is below 0.05×10^{-6}.
- This is very conservative! We will hardly ever reject the null hypothesis.
- Slightly less conservative alternatives to Bonferroni, that also control overall Type I error, are also available (but not discussed here).

A Less Conservative Notion of Error Control

- The Bonferroni correction allows us to be confident that we will almost never falsely reject the null hypothesis.
- But in the analysis of omics data, this approach can be a little too conservative.
- If an investigator spends a lot of money measuring DNA methylation levels at millions of sites, he or she might be willing to sometimes falsely reject the null hypothesis, in exchange for correctly rejecting the null hypothesis more frequently.
- In other words: willing to have some false positives, in exchange for more true positives.
- A new notion of error control: false discovery rate.
- Proposed in 1995, but still a very active area of research!

Possible Outcomes from M Hypothesis Tests

	Called Not Significant	Called Significant	Total
H_0 True	U	V	M_0
H_0 False	T	S	M_1
Total	$M - R$	R	M

- V is number of false positives.
- T is number of false negatives.
- Type I error is $E(V)/M_0$.
- Type II error is $E(T)/M_1$.
- The power is $1 - E(T)/M_1$.
- The Bonferroni correction guarantees that $P(V \geq 1) \leq \alpha$.

False Discovery Rate

$$FDR = E\left(\frac{\text{number of null hypotheses falsely rejected}}{\text{number of null hypotheses rejected}}\right).$$

For instance, in a gene expression experiment,

$$FDR = E\left(\frac{\text{number of genes incorrectly declared significant}}{\text{number of genes declared significant}}\right).$$

In other words, this is the fraction of discoveries that are false positives.

False Discovery Rate

	Called Not Significant	Called Significant	Total
H_0 True	U	V	M_0
H_0 False	T	S	M_1
Total	$M - R$	R	M

- FDR: V/R, which is random... we often control $E(V/R)$.
- By controlling the FDR, we are guaranteeing that we don't have too many false discoveries.
- If we use an FDR threshold of 0.2, then no more than 20% of the null hypotheses that we reject were actually true. (Unfortunately, we don't know which ones!)
- For instance, in a GWAS, if we reject the null hypothesis of no association for SNPs with $FDR \leq 0.2$, then we expect no more than 20% of those SNPs to be false positives.

Benjamini-Hochberg Algorithm for FDR Control

1. Fix the false discovery rate, α.
2. Let $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(M)}$ denote the ordered p-values for the M hypothesis tests.
3. Define
$$L = \max \left\{ k : p_{(k)} < \alpha \frac{k}{M} \right\}.$$
4. Then, reject the jth null hypothesis if $p_j \leq p_{(L)}$, the Benjamini-Hochberg rejection threshold.
5. In other words, find the maximum order statistic (k) such that
$$\frac{M \times p_{(k)}}{k} \leq \alpha$$
Then reject all tests for which $p_j \leq p_{(k)}$

FDR vs Bonferroni Control

- Benjamini & Hochberg:
  - Find the maximum order statistic (k) such that
$$p(k) \leq \frac{\alpha k}{M}$$
  - Reject all tests j with $p_j < p(k)$.
- Bonferroni:
  - Reject test j if
$$p_j \leq \frac{\alpha}{M}$$

Example in R: All Null Hypotheses Hold

```
x <- matrix(rnorm(1000*50),ncol=50)
y <- sample(c(0,1),50,rep=TRUE)
ps <- NULL
for(i in 1:1000) ps <- c(ps,
 t.test(x[i,y==0],x[i,y==1])$p.value)
cat("Around 5% of p-values are below 0.05:",
mean(ps<.05),fill=TRUE)
fdrs.bh <- p.adjust(ps, method="BH")
plot(ps,fdrs.bh)
plot(fdrs.bh)
```

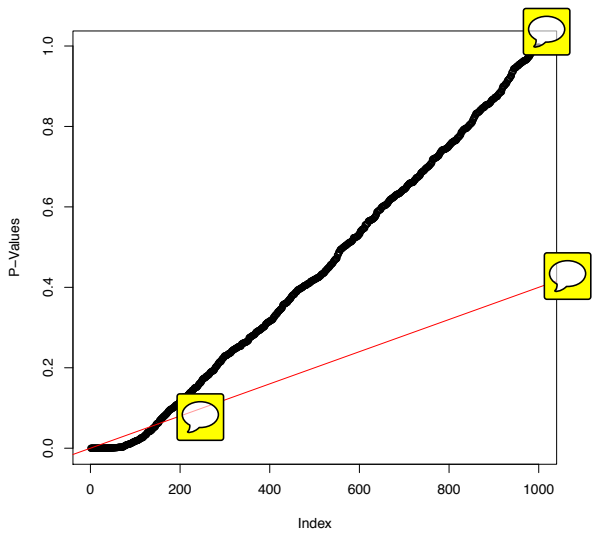

Example in R: Not All Null Hypotheses Hold

```
x <- matrix(rnorm(1000*50),ncol=50)
y <- sample(c(0,1),50,rep=TRUE)
x[1:100,y==0] <- x[1:100,y==0] + 1
ps <- NULL
for(i in 1:1000) ps <- c(ps,
t.test(x[i,y==0],x[i,y==1])$p.value)
cat("Way more than 5% of p-values are below 0.05:",
mean(ps<.05),fill=TRUE)
fdrs.bh <- p.adjust(ps, method="BH")
plot(ps,fdrs.bh)
plot(fdrs.bh)
```

Example, Continued

```
cat("Number of Tests with FDR below 0.4:",
sum(fdrs.bh<0.4), fill=TRUE)
cat("Compute the BH FDR Directly:",
max(which(sort(ps,decreasing=FALSE) < .4*(1:1000)/1000)), fill=TRUE)
plot(sort(ps,decreasing=FALSE),ylab="P-Values")
abline(a=0, b=0.4/1000,col="red")
```


Output From R



Correlated Hypotheses

- So far, we have not discussed the effect of correlation in the data
- In real data examples, genes, proteins etc work together, and are correlated
- How would methods of multiple testing adjustment work if tests are correlated?

Correlated Hypotheses

- Bonferroni correction works regardless of the correlation structure (however, we know that in general, Bonferroni is not very appealing)
- Benjamini & Hochberg can handle positive dependence: think of this as all genes being positively correlated with each other
- However, in practice genes may have both positive (inducers) and negative (inhibitors) correlations with each other, so this assumption may not hold
- This is still an active area of research

FDR Correction Under Dependence

- A control under dependence due to Benjamini & Yakutielli
  - Find the maximum order statistic (k) such that

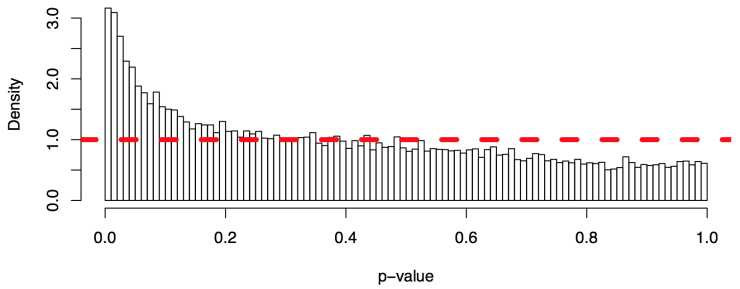
$$p(k) \leq \frac{\alpha k}{M \times \left(\sum_{j=1}^{M} 1/j\right)}$$

  - Reject all j with $p_j \leq p(k)$.
- $\sum_{j=1}^{M} 1/j \approx \log(M)$, so we pay an additional price of $\log(M)$, which makes this procedure more conservative than Benjamini & Hochberg (but then again, there is no free lunch!)
- In R, `p.adjust(ps, method=''BY'')`

A Real Data Example

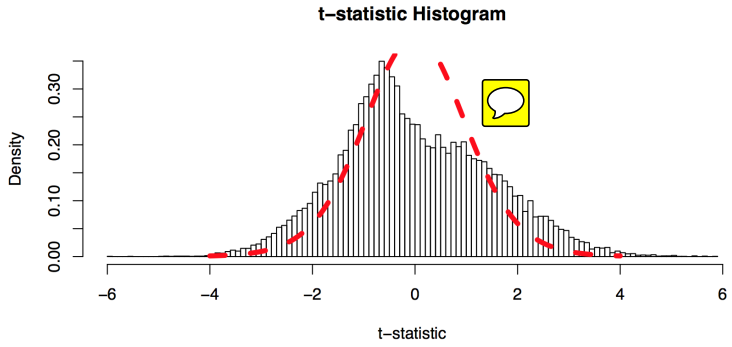
- Mechanisms for Cancer immortality! (TELO vs ALT)
- Paper by Lafferty-whyte and co claiming to build a gene signature to differentiate mechanisms

Let's look at the p-values



A Real Data Example

And, test statistics



What is going on??

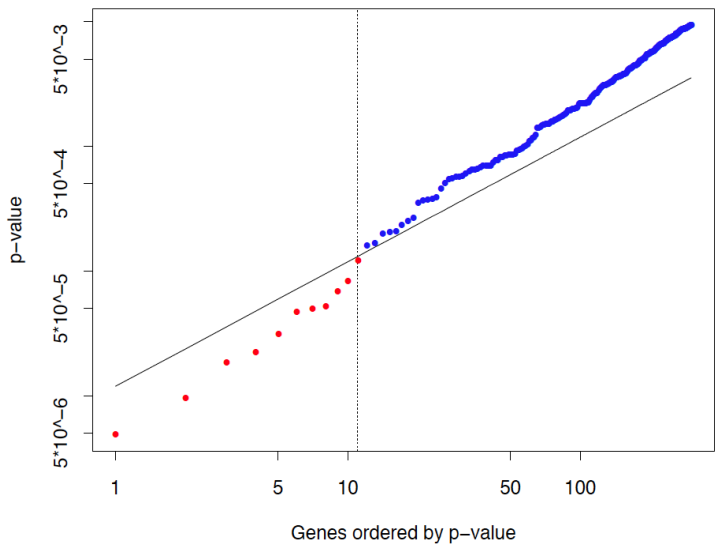
A Real Data Example

- With dependence correction, 0 rejections for FDR control ¡ 1
- Unclear how significant findings are...
  - Report K (10, 100, ...) most significant effects
  - Give FDR estimates from both “BH” and “BY”
  - Small sample-size and strange histogram shape add extra skepticism
  - We also need to check for batch effects and other potentially damaging factors

Example: Radiation Treatment Sensitivity

- Microarray data set on sensitivity of cancer patients to ionizing radiation treatment.
- Citation: Rieger, Hong et al, PNAS, 2004
- $M = 12,625$ genes, $n = 58$ samples: 44 samples with a normal reaction, 14 patients with severe reaction to radiation.
- Compute two-sample t-statistic for each gene's association with response to radiation.
- NO genes are declared significant after Bonferroni correction at level $\alpha = 0.05$.
- 11 genes had FDR below $\alpha = 0.15$.
- For each gene we can get a q-value, which is like a p-value for that gene, but quantifies the FDR associated with that gene.

Example: Radiation Treatment Sensitivity



Permutation Approach to FDR Estimation

- Another way to estimate FDRs: by permutation.
- Easy and natural – unlike Benjamini-Hochberg, can explain it pretty simply to a non-statistician.
- Does not require computing p-values, which can be helpful in settings where p-values are (1) difficult to compute, or (2) unreliable.
- Also known as "plug-in estimate for FDR".

Permutation Approach

1. Compute $t_1, \ldots, t_M$, the test statistic for each of the M features.
2. Create K permutations of the responses, and for each permutation from $k = 1, \ldots, K$, and for each feature $j = 1, \ldots, M$, compute $t_j^1, \ldots, t_j^K$.
3. For a range of values of the cut-point C, let

$$R_{obs} = \sum_{j=1}^{M} 1_{(|t_j| > C)},$$

and

$$\widehat{E(V)} = \frac{1}{K} \sum_{j=1}^{M} \sum_{k=1}^{K} 1_{(|t_j^k| > C)}.$$

4. Estimate the FDR by $\widehat{FDR} = \widehat{E(V)}/R_{obs}$.

Permutation Approach

- Based upon the approximation

$$E(V/R) \approx E(V)/E(R).$$

- We estimate $E(R)$ using R_{obs} – the actual number of test statistics that exceed the threshold.
- We estimate $E(V)$ by permuting the response and calculating the number of test statistics that exceed the threshold, in the absence of any real relationship between the features and the response.
- Actually, $\widehat{E(V)}$ estimates $(M/M_0)E(V)$, but since $M > M_0$, this over-estimate is conservative.

Permutation Approach

- An advantage of the permutation approach: can estimate FDR even when we don't know how to compute p-values.
- For instance, suppose we have protein level measurements for 100 proteins, as well as associated response measurements, in both pregnant and non-pregnant women.
- We want to develop a test statistic that combines the info from the pregnant women and from the non-pregnant women.
- For the jth gene, can compute $t_j^{pregnant}$, the test statistic for pregnant women, and $t_j^{non-pregnant}$, the test statistic for non-pregnant women.
- Can use $|t_j^{pregnant}| + |t_j^{non-pregnant}|$ as an overall test statistic.
- How to get p-value for this new test statistic? Who knows!
- But estimating FDR using a permutation approach is pretty straightforward.

At What Level to Control FDR?

- In biology, we typically control Type I Error at level $\alpha = 0.05$ or $\alpha = 0.01$. (We reject the null hypothesis if the p-value is less than 0.05 or 0.01.)
- However, there is no analogous default cut-off for rejecting the null hypothesis using FDR control.
- We might want to follow up on genes whose FDR is below 10% or 20% in a gene expression experiment.
- In general, the FDR threshold that we use will depend on the data set as well as the number of genes with small FDRs... as well as the number of rejected null hypotheses that we can afford to follow up on in the lab!

P-Values Using Permutations

- We could also compute p-values using a permutation-type approach: the p-value for the jth feature is given by

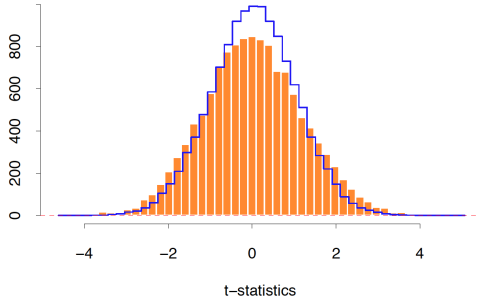
$$p_j = \frac{1}{MK} \sum_{k=1}^{K} \sum_{j'=1}^{M} 1_{(|t_{j'}^k| > |t_j|)}.$$

- Can also use the permutation p-values only

$$p_j = \frac{1}{K} \sum_{k=1}^{K} 1_{(|t_{j'}^k| > |t_j|)}.$$

P-Values Using Permutations

- Example on radiation sensitivity data: orange shows true t-statistics, and blue shows t-statistics for permuted data.



Asymmetric Cut-Points

- In previous examples, we used the absolute value of the test statistic, $|t_j|$, and hence the same cut-points for both positive and negative values
- In some cases, (most) differentially expressed genes change in the positive (or negative) direction. In such cases, it may be better to use different cut-points for negative and positive test statistics
- SAM (Significance Analysis of Microarrays) is such a procedure (Tusher et al, 2001):

SAM (Significance Analysis of Microarrays)

Tusher et al (2001)

- uses the values of test-statistics instead of p-values
- based on a permutation approach to determine the "expected" test-statistics
- uses a QQ-plot to find test statistics that are far from expected
- the cutoff value is determined by finding a band of length Δ around the expected test-statistics
- is implemented in the R-package `samr`: `SAM(x,y=NULL,...)`

SAM