Regularized regression

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Basic idea

1. Fit a regression model
2. Penalize (or shrink) large coefficients

Pros:

- Can help with the bias/variance tradeoff
- Can help with model selection

Cons:

- May be computationally demanding on large data sets
- Does not perform as well as random forests and boosting

A motivating example

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

where X_1 and X_2 are nearly perfectly correlated (co-linear). You can approximate this model by:

$$Y = \beta_0 + (\beta_1 + \beta_2) X_1 + \epsilon$$

The result is:

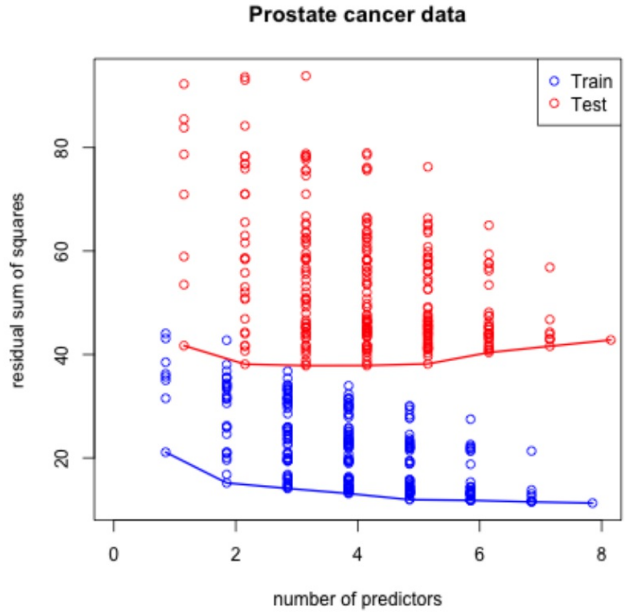
- You will get a good estimate of Y
- The estimate (of Y) will be biased
- We may reduce variance in the estimate

Prostate cancer

```
library(ElemStatLearn); data(prostate)
str(prostate)
```

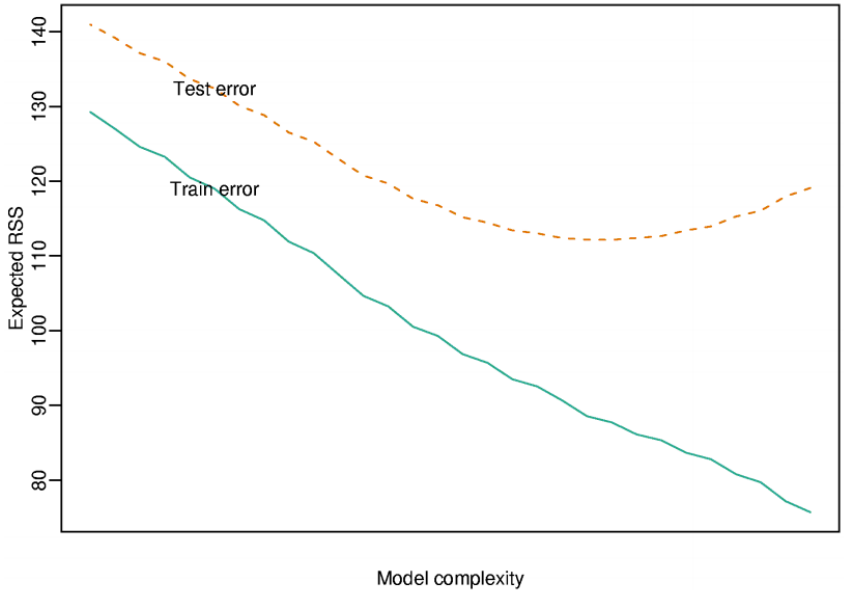
```
'data.frame':   97 obs. of  10 variables:
 $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
 $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
 $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
 $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
 $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
 $ train  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

Subset selection



Code here

Most common pattern



http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/

Model selection approach: split samples

- No method better when data/computation time permits it
- Approach
  1. Divide data into training/test/validation
  2. Treat validation as test data, train all competing models on the train data and pick the best one on validation.
  3. To appropriately assess performance on new data apply to test set
  4. You may re-split and reperform steps 1-3
- Two common problems
  - Limited data
  - Computational complexity

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Decomposing expected prediction error

Assume $Y_i = f(X_i) + \epsilon_i$

$$EPE(\lambda) = E\left[\{Y - \hat{f}_\lambda(X)\}^2\right]$$

Suppose $\hat{f}_\lambda$ is the estimate from the training data and look at a new data point $X = x^*$

$$E\left[\{Y - \hat{f}_\lambda(x^*)\}^2\right] = \sigma^2 + \{E[\hat{f}_\lambda(x^*)] - f(x^*)\}^2 + var[\hat{f}_\lambda(x_0)]$$

$$= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}$$

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Another issue for high-dimensional data

```
small = prostate[1:5,]
lm(lpsa ~ .,data =small)
```

```
Call:
lm(formula = lpsa ~ ., data = small)

Coefficients:
(Intercept)       lcavol      lweight          age         lbph          svi          lcp
     9.6061       0.1390      -0.7914       0.0952           NA           NA           NA
    gleason        pgg45    trainTRUE
    -2.0871           NA           NA
```

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Hard thresholding

- Model $Y = f(X) + \epsilon$
- Set $\hat{f}_\lambda(x) = x'\beta$
- Constrain only λ coefficients to be nonzero.
- Selection problem is after chosing λ figure out which $p - \lambda$ coefficients to make nonzero

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Regularization for regression

If the β_j's are unconstrained:

- They can explode
- And hence are susceptible to very high variance

To control variance, we might regularize/shrink the coefficients.

$$\text{PRSS}(\beta) = \sum_{j=1}^{n}(Y_j - \sum_{i=1}^{m} \beta_{1i} X_{ij})^2 + P(\lambda; \beta)$$

where PRSS is a penalized form of the sum of squares. Things that are commonly looked for

- Penalty reduces complexity
- Penalty reduces variance
- Penalty respects structure of the problem

Ridge regression

Solve:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

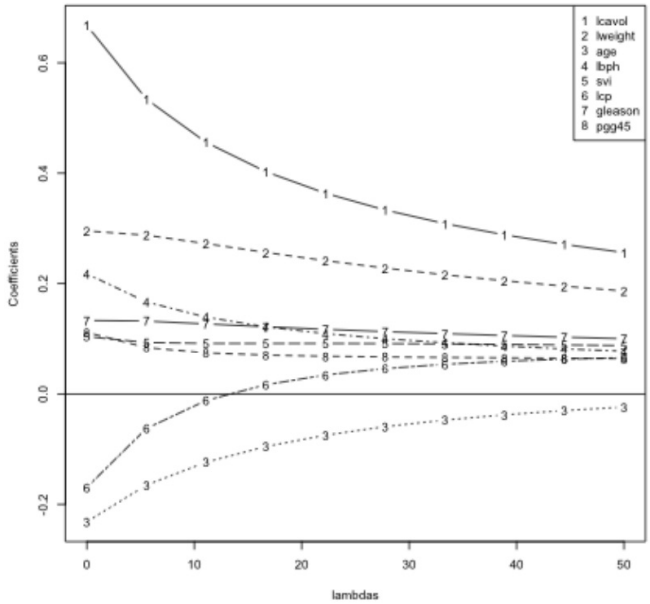
equivalent to solving

$\sum_{i=1}^{N}\left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)^2$ subject to $\sum_{j=1}^{p} \beta_j^2 \leq s$ where s is inversely proportional to λ

Inclusion of λ makes the problem non-singular even if $X^T X$ is not invertible.

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Ridge coefficient paths



http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Tuning parameter λ

- λ controls the size of the coefficients
- λ controls the amount of {\bf regularization}
- As $\lambda \to 0$ we obtain the least square solution
- As $\lambda \to \infty$ we have $\hat{\beta}^{\text{ridge}}_{\lambda=\infty} = 0$

Lasso

$\sum_{i=1}^{N} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$ subject to $\sum_{j=1}^{p} |\beta_j| \leq s$

also has a lagrangian form

$$\sum_{i=1}^{N} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

For orthonormal design matrices (not the norm!) this has a closed form solution

$$\hat{\beta}_j = \text{sign}(\hat{\beta}_j^0)(|\hat{\beta}_j^0| - \gamma)^+$$

but not in general.

http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/ http://www.cbcb.umd.edu/~hcorrada/PracticalML/

Notes and further reading

- Hector Corrada Bravo's Practical Machine Learning lecture notes
- Hector's penalized regression reading list
- Elements of Statistical Learning
- In `caret` methods are:
  - `ridge`
  - `lasso`
  - `relaxo`

Combining predictors

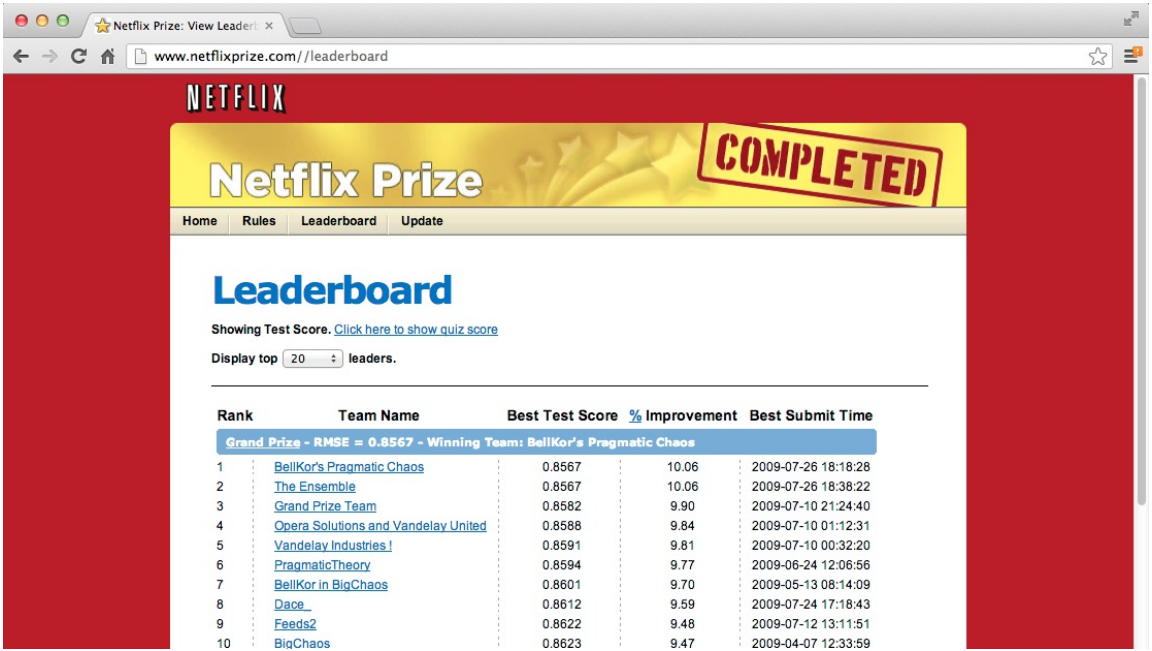
Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Key ideas

- You can combine classifiers by averaging/voting
- Combining classifiers improves accuracy
- Combining classifiers reduces interpretability
- Boosting, bagging, and random forests are variants on this theme

Netflix prize

BellKor = Combination of 107 predictors



http://www.netflixprize.com//leaderboard

Heritage health prize - Progress Prize 1

2. *Predictive Modelling*
   Predictive models were built utilising the data sets created in Step 1. Numerous mathematical techniques were used to generate a set of candidate solutions.

3. *Ensembling*
   The individual solutions produced in Step 2 were combined to create a single solution that was more accurate than any of its components.

Market Makers

1 Introduction

My milestone 1 solution to the Heritage Health Prize with a RMSLE score of 0.457239 on the leaderboard consists of a linear blend of 21 result. These are mostly generated by relatively simple models which are all trained using stochastic gradient descent. First in section 2 I provide a description of the way the data is organized and the features that were used. Then in section 3 the training method and the post-processing steps are described. In section 4 each individual model is briefly described, all the relevant meta-parameter settings can be found in appendix Parameter settings. Finally the weights in the final blend are given in section 5.

Mestrom

Basic intuition - majority vote

Suppose we have 5 completely independent classifiers

If accuracy is 70% for each:

- $10 \times (0.7)^3(0.3)^2 + 5 \times (0.7)^4(0.3)^2 + (0.7)^5$
- 83.7% majority vote accuracy

With 101 independent classifiers

- 99.9% majority vote accuracy

Approaches for combining classifiers

1. Bagging, boosting, random forests
   - Usually combine similar classifiers
2. Combining different classifiers
   - Model stacking
   - Model ensembling

Example with Wage data

Create training, test and validation sets

```
library(ISLR); data(Wage); library(ggplot2); library(caret);
Wage <- subset(Wage,select=-c(logwage))

# Create a building data set and validation set
inBuild <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
validation <- Wage[-inBuild,]; buildData <- Wage[inBuild,]

inTrain <- createDataPartition(y=buildData$wage,
                               p=0.7, list=FALSE)
training <- buildData[inTrain,]; testing <- buildData[-inTrain,]
```

Wage data sets

Create training, test and validation sets

```
dim(training)
```

```
[1] 1474   11
```

```
dim(testing)
```

```
[1] 628  11
```

```
dim(validation)
```

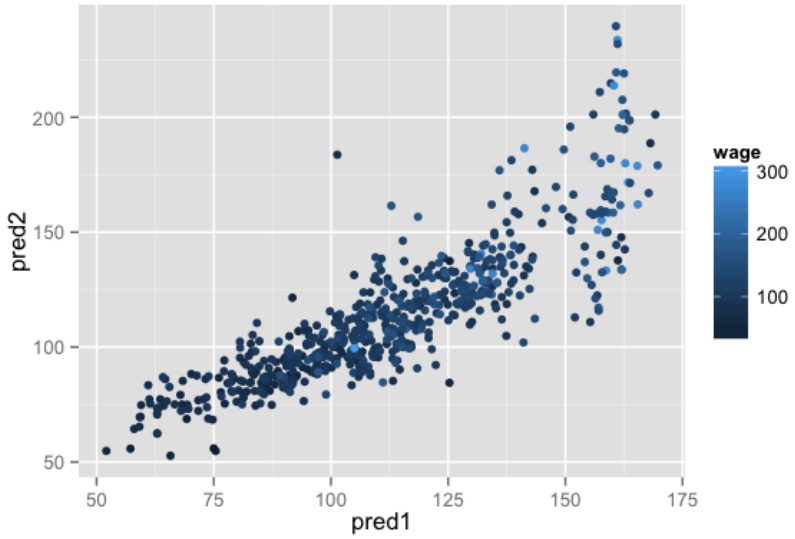
```
[1] 898  11
```

Build two different models

```
mod1 <- train(wage ~.,method="glm",data=training)
mod2 <- train(wage ~.,method="rf",
              data=training,
              trControl = trainControl(method="cv"),number=3)
```

Predict on the testing set

```
pred1 <- predict(mod1,testing); pred2 <- predict(mod2,testing)
qplot(pred1,pred2,colour=wage,data=testing)
```



Fit a model that combines predictors

```
predDF <- data.frame(pred1,pred2,wage=testing$wage)
combModFit <- train(wage ~.,method="gam",data=predDF)
combPred <- predict(combModFit,predDF)
```

Testing errors

```
sqrt(sum((pred1-testing$wage)^2))
```

```
[1] 827.1
```

```
sqrt(sum((pred2-testing$wage)^2))
```

```
[1] 866.8
```

```
sqrt(sum((combPred-testing$wage)^2))
```

```
[1] 813.9
```

Predict on validation data set

```
pred1V <- predict(mod1,validation); pred2V <- predict(mod2,validation)
predVDF <- data.frame(pred1=pred1V,pred2=pred2V)
combPredV <- predict(combModFit,predVDF)
```

Evaluate on validation

```
sqrt(sum((pred1V-validation$wage)^2))
```

```
[1] 1003
```

```
sqrt(sum((pred2V-validation$wage)^2))
```

```
[1] 1068
```

```
sqrt(sum((combPredV-validation$wage)^2))
```

```
[1] 999.9
```

Notes and further resources

- Even simple blending can be useful
- Typical model for binary/multiclass data
  - Build an odd number of models
  - Predict with each model
  - Predict the class by majority vote
- This can get dramatically more complicated
  - Simple blending in caret: caretEnsemble (use at your own risk!)
  - Wikipedia ensemlbe learning

Recall - scalability matters

Innovation
by Mike Masnick
Fri, Apr 13th 2012
12:07am

5

Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge

from the *times-change* dept

You probably recall all the excitement that went around when a group **finally won** the big Netflix $1 million prize in 2009, improving Netflix's recommendation algorithm by 10%. But what you might *not* know, is that **Netflix never implemented that solution itself.** Netflix recently put up a blog post **discussing some of the details of its recommendation system**, which (as an aside) explains why the winning entry never was used. First, they note that they *did* make use of an earlier bit of code that came out of the contest:

http://www.techdirt.com/blog/innovation/articles/20120409/03412518422/

http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

Unsupervised prediction

Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Sometimes you don't know the labels for prediction
- To build a predictor
  - Create clusters
  - Name clusters
  - Build predictor for clusters
- In a new data set
  - Predict clusters

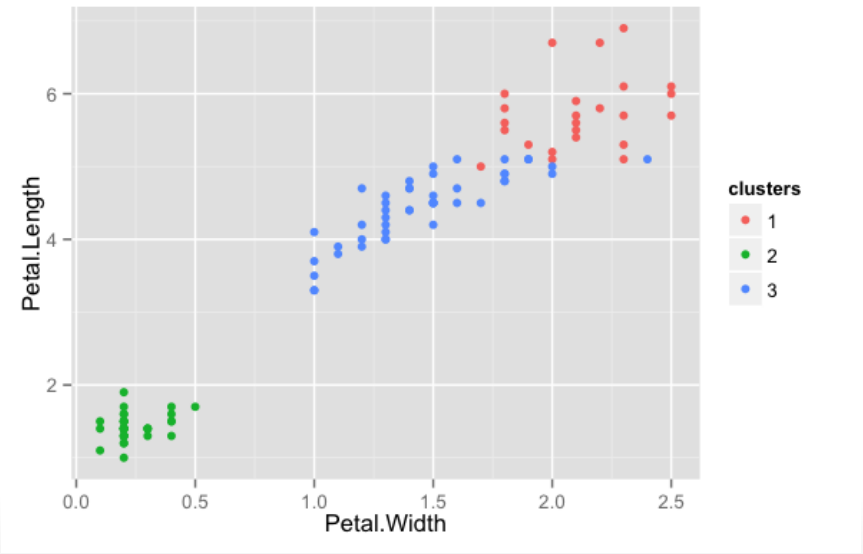
Iris example ignoring species labels

```
data(iris); library(ggplot2)
inTrain <- createDataPartition(y=iris$Species,
                               p=0.7, list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
dim(training); dim(testing)
```

```
[1] 45  5
```

Cluster with k-means

```
kMeans1 <- kmeans(subset(training,select=-c(Species)),centers=3)
training$clusters <- as.factor(kMeans1$cluster)
qplot(Petal.Width,Petal.Length,colour=clusters,data=training)
```



Compare to real labels

```
table(kMeans1$cluster,training$Species)
```

```
   setosa versicolor virginica
1       0          1        23
2      35          0         0
3       0         34        12
```

Build predictor

```
modFit <- train(clusters ~.,data=subset(training,select=-c(Species)),method="rpart")
table(predict(modFit,training),training$Species)
```

```
   setosa versicolor virginica
1      0          0        21
2     35          0         0
3      0         35        14
```

Apply on test

```
testClusterPred <- predict(modFit,testing)
table(testClusterPred ,testing$Species)
```

```
testClusterPred setosa versicolor virginica
              1      0          0        13
              2     15          0         0
              3      0         15         2
```

Notes and further reading

- The cl_predict function in the clue package provides similar functionality
- Beware over-interpretation of clusters!
- This is one basic approach to recommendation engines
- Elements of statistical learning
- Introduction to statistical learning

Forecasting

Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

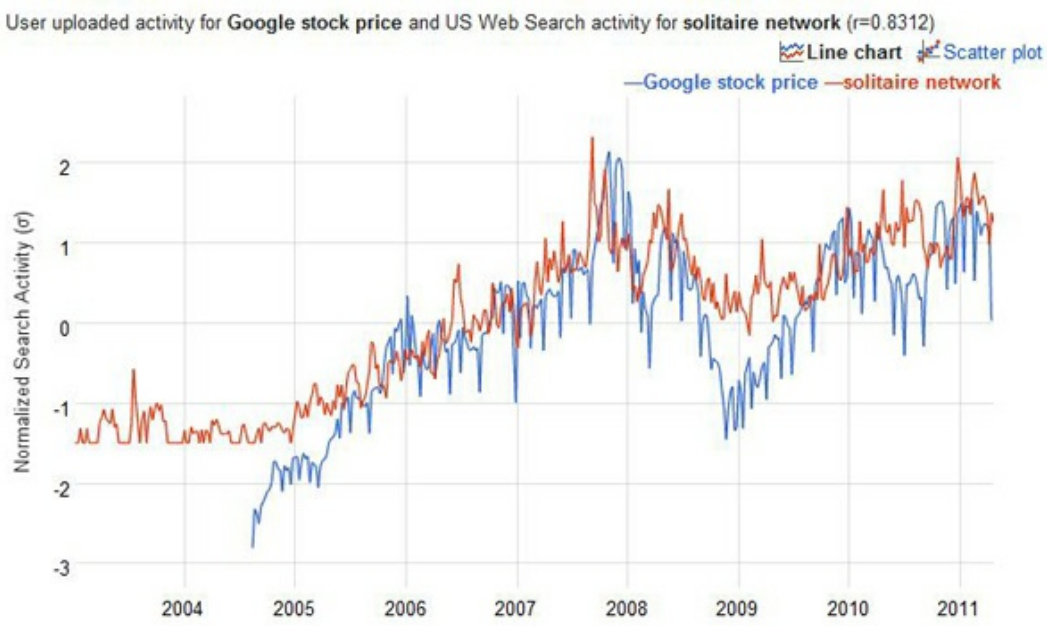
Time series data

https://www.google.com/finance

What is different?

- Data are dependent over time
- Specific pattern types
  - Trends - long term increase or decrease
  - Seasonal patterns - patterns related to time of week, month, year, etc.
  - Cycles - patterns that rise and fall periodically
- Subsampling into training/test is more complicated
- Similar issues arise in spatial data
  - Dependency between nearby observations
  - Location specific effects
- Typically goal is to predict one or more observations into the future.
- All standard predictions can be used (with caution!)

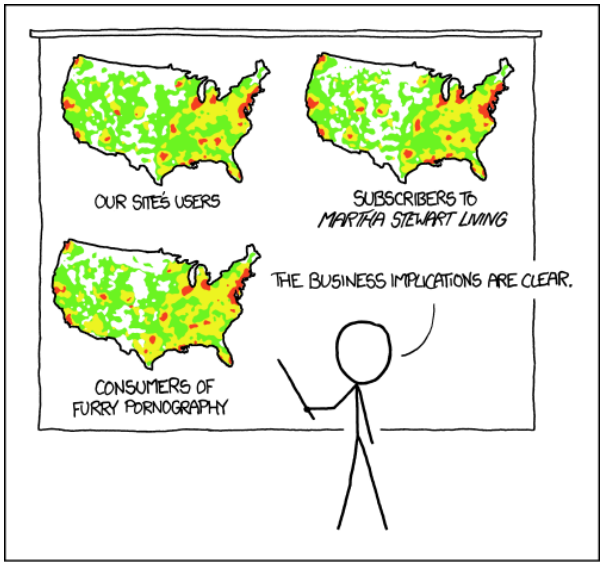
Beware spurious correlations!



http://www.google.com/trends/correlate

http://www.newscientist.com/blogs/onepercent/2011/05/google-correlate-passes-our-we.html

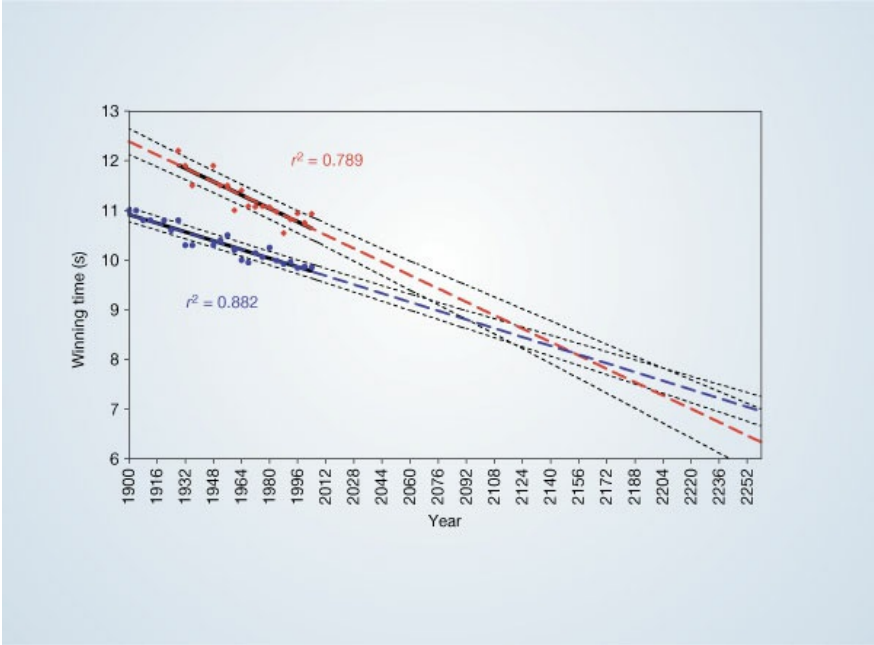
Also common in geographic analyses



PET PEEVE #208:
GEOGRAPHIC PROFILE MAPS WHICH ARE BASICALLY JUST POPULATION MAPS

http://xkcd.com/1138/

Beware extrapolation!



http://www.nature.com/nature/journal/v431/n7008/full/431525a.html

Google data

```
library(quantmod)
from.dat <- as.Date("01/01/08", format="%m/%d/%y")
to.dat <- as.Date("12/31/13", format="%m/%d/%y")
getSymbols("GOOG", src="google", from = from.dat, to = to.dat)
```

```
[1] "GOOG"
```

```
head(GOOG)
```

```
           GOOG.Open GOOG.High GOOG.Low GOOG.Close GOOG.Volume
2008-01-02     692.9     697.4    677.7      685.2     4306848
2008-01-03     685.3     686.9    676.5      685.3     3252846
2008-01-04     679.7     681.0    655.0      657.0     5359834
2008-01-07     653.9     662.3    637.4      649.2     6404945
2008-01-08     653.0     660.0    631.0      631.7     5341949
2008-01-09     630.0     653.3    622.5      653.2     6744242
```

Summarize monthly and store as time series

```
mGoog <- to.monthly(GOOG)
googOpen <- Op(mGoog)
ts1 <- ts(googOpen,frequency=12)
plot(ts1,xlab="Years+1", ylab="GOOG")
```

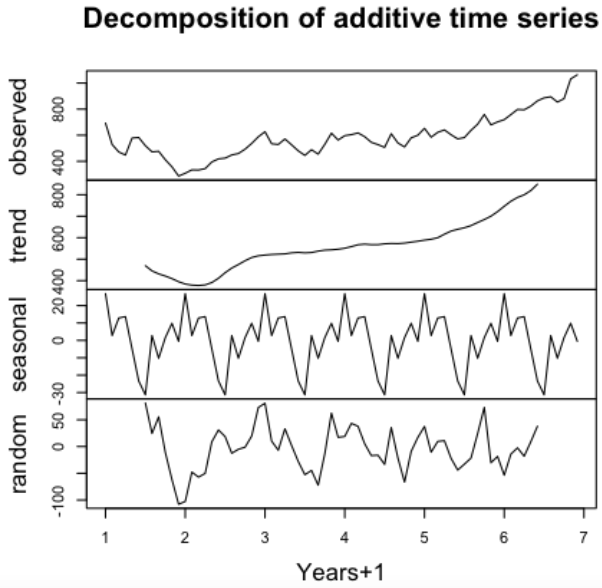
Example time series decomposition

- **Trend** - Consistently increasing pattern over time
- **Seasonal** - When there is a pattern over a fixed period of time that recurs.
- **Cyclic** - When data rises and falls over non fixed periods

https://www.otexts.org/fpp/6/1

Decompose a time series into parts

```
plot(decompose(ts1),xlab="Years+1")
```



Training and test sets

```
ts1Train <- window(ts1,start=1,end=5)
ts1Test <- window(ts1,start=5,end=(7-0.01))
ts1Train
```

```
    Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1 692.9 528.7 471.5 447.7 578.3 582.5 519.6 472.5 476.8 412.1 357.6 286.7
2 308.6 334.3 333.3 343.8 395.0 418.7 424.2 448.7 459.7 493.0 537.1 588.1
3 627.0 534.6 529.2 571.4 526.5 480.4 445.3 489.0 455.0 530.0 615.7 563.0
4 596.5 604.5 617.8 588.8 545.7 528.0 506.7 611.2 540.8 509.9 580.1 600.0
5 652.9
```

Simple moving average

$$Y_t = \frac{1}{2 * k + 1} \sum_{j=-k}^{k} y_{t+j}$$

```
plot(ts1Train)
lines(ma(ts1Train,order=3),col="red")
```

Exponential smoothing

Example - simple exponential smoothing

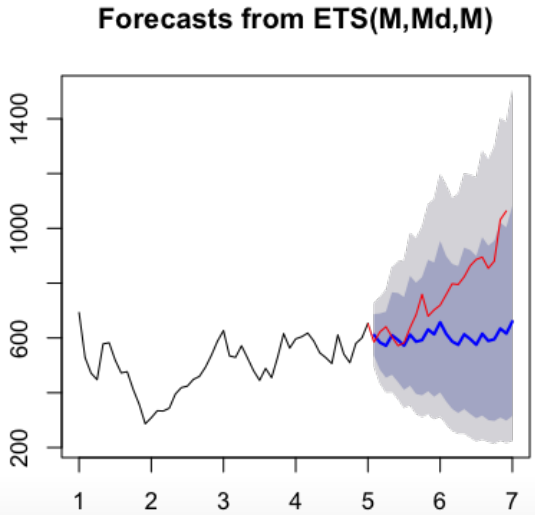
$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_{t-1}$$

		Seasonal Component	
Trend Component	N (None)	A (Additive)	M (Multiplicative)
N (None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
A_d (Additive damped)	(A_d,N)	(A_d,A)	(A_d,M)
M (Multiplicative)	(M,N)	(M,A)	(M,M)
M_d (Multiplicative damped)	(M_d,N)	(M_d,A)	(M_d,M)

https://www.otexts.org/fpp/7/6

Exponential smoothing

```
ets1 <- ets(ts1Train,model="MMM")
fcast <- forecast(ets1)
plot(fcast); lines(ts1Test,col="red")
```



Get the accuracy

```
accuracy(fcast,ts1Test)
```

```
                    ME    RMSE     MAE      MPE    MAPE    MASE     ACF1 Theil's U
Training set   0.9464   48.78   39.35 -0.3297   7.932 0.3733 0.07298        NA
Test set     156.1890 205.76 160.78 18.1819 18.971 1.5254 0.77025     3.745
```

Notes and further resources

- Forecasting and timeseries prediction is an entire field
- Rob Hyndman's Forecasting: principles and practice is a good place to start
- Cautions
  - Be wary of spurious correlations
  - Be careful how far you predict (extrapolation)
  - Be wary of dependencies over time
- See quantmod or quandl packages for finance-related problems.