Week 4: Regularized Regression and Combining Predictors

Lectures[*]**:** 1. Regularized regression, 2. Combining predictors, 3. Forecasting, 4. Unsupervised Prediction

Juan David Leongómez

13 April, 2021

Contents

[*]All lectures by Jeffrey Leek (John Hopkins Bloomberg Scool of Public Health).

1 Lecture 1: Regularized regression

This lecture is about Regularized regression. We learned about linear regression and generalized linear regression previously.

1.1 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 (when applied to prediction in the wild; for example, in Kaggle competitions)

1.2 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:

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

Note: it will not be exactly right, because $X1$ and $X2$ are not exactly the same variable. But it will be very close to right, if $X1$ and $X2$ are very similar to each other.

The result is:

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

1.3 Subset selection

Suppose we predict with all possible combinations of predictor variables. For the outcome where we build one regression model for every possible combination of vectors. As the number of predictors increases from left to right here, the training set error always goes down. As you include more predictors, the training set error will always decrease.

But this is a typical pattern of what you observe with real data, that the test set data on the other hand, as the number of predictors increases, the test set error goes down, which is good.

But then eventually it hits a plateau, and it starts to go back up again. This is because we're overfitting the data in the training set, and eventually, we may not want to include so many predictors in our model.

1.3.1 Most common pattern

This is an incredibly common pattern (see Fig. 1).

In the training set almost always the error goes monotonically down (i.e. as you build more and more complicated models, the training error will always decrease). But on a testing set, the error will decrease for a while, eventual hit a minimum. And then, start to increase again as the model gets too complex and over fits the data.

```r
knitr::include_graphics("trainingandtest.png")
```

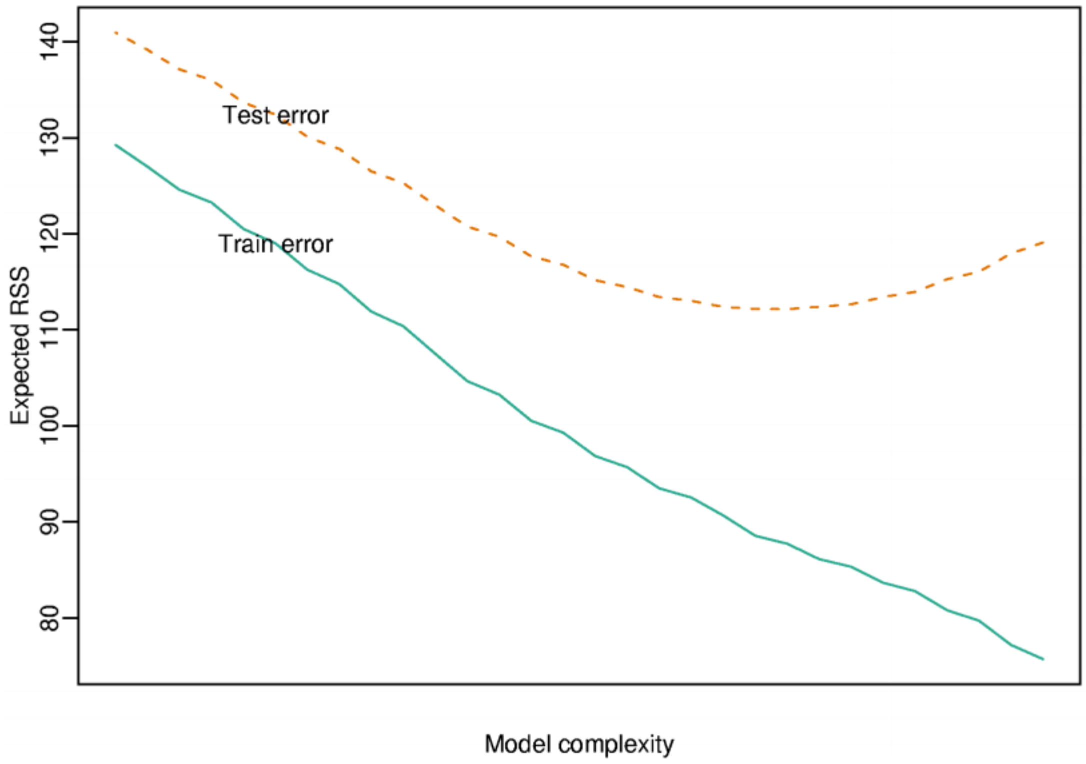


Figure 1: Comon pattern fro the association between model complexity and the expected Residual Sum of Squares (RSS). See http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/.

1.3.1.1 Prostate cancer example

This can be seen in other examples (see Fig. 2; Code here).

```r
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 ...
```

```r
knitr::include_graphics("prostate.png")
```

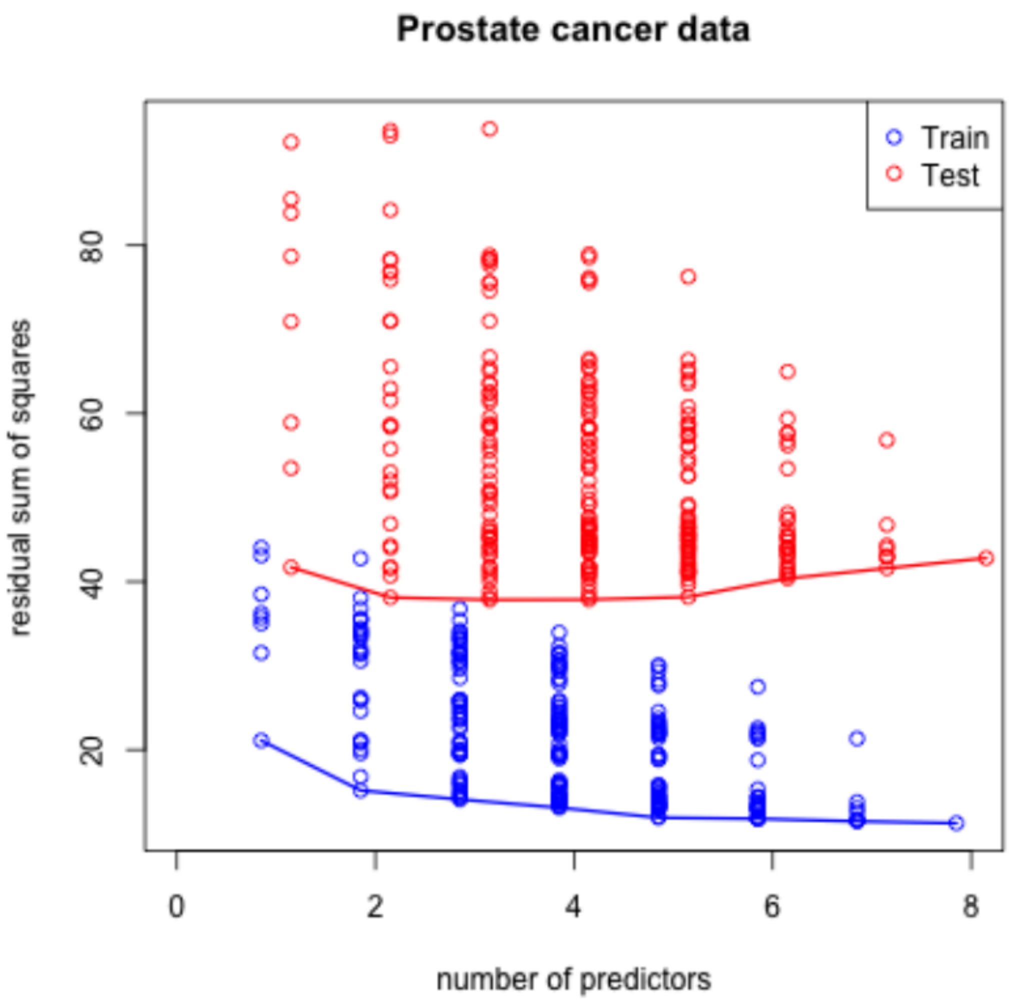


Figure 2: Prostate cancer data.

1.4 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 perform steps 1-3 again
- Two common problems
  - Limited data
  - Computational complexity

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

http://www.cbcb.umd.edu/~hcorrada/PracticalML/

1.4.1 Decomposing expected prediction error

Another approach is to try to decompose the prediction error, and see if there is another way that we can work directly get at including only the variable that need to be included in the model.

If we assume that the variable Y can be predicted as a function of X, plus some error term: $Y_i = f(X_i) + \epsilon_i$

Then the expected prediction error is the expected difference between the outcome and the prediction of the outcome squared: $EPE(\lambda) = E\left[\{Y - \hat{f}_\lambda(X)\}^2\right]$

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

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

$$= Irreducibleerror + Bias^2 + Variance$$

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

1.5 Another issue for high-dimensional data

Just a simple example of what happens when you have a lot of predictors.

So here I'm sub-setting just a small subset of the prostate data. Imagine that I only had five observations in my training set (It has more than five predictor variables).

So I fit a linear model relating the outcome to all of these predictor variables. Because there are more than five, some of them will get estimates (NA).

```
small <- prostate[1:5,]

lm(lpsa ~.,
   data = small)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = small)
##
## Coefficients:
## (Intercept)       lcavol      lweight          age         lbph          svi
##     9.60615      0.13901     -0.79142      0.09516           NA           NA
##         lcp      gleason        pgg45    trainTRUE
##          NA     -2.08710           NA           NA
```

In other words, R won't be able to estimate them because you have more predictors than you have samples. You have, design matrix that cannot be inverted.

1.6 Hard thresholding

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

1.7 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.

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

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

1.7.1 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 $\sum_{j=1}^{p}\beta_j^2 \le s$ where s is inversely proportional to λ

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

Ridge coefficient paths

For every different choice of λ, that penalized regression problem on the previous page, as gambit increases.

That means that we penalize the big λ more and more.

So we start off with the betas being equal to a certain of values here when $\lambda = 0$. That's just a standard linear with regression values. And as you increase lambda, all of the coefficients get closer to 0.

```
knitr::include_graphics("ridgepath.png")
```

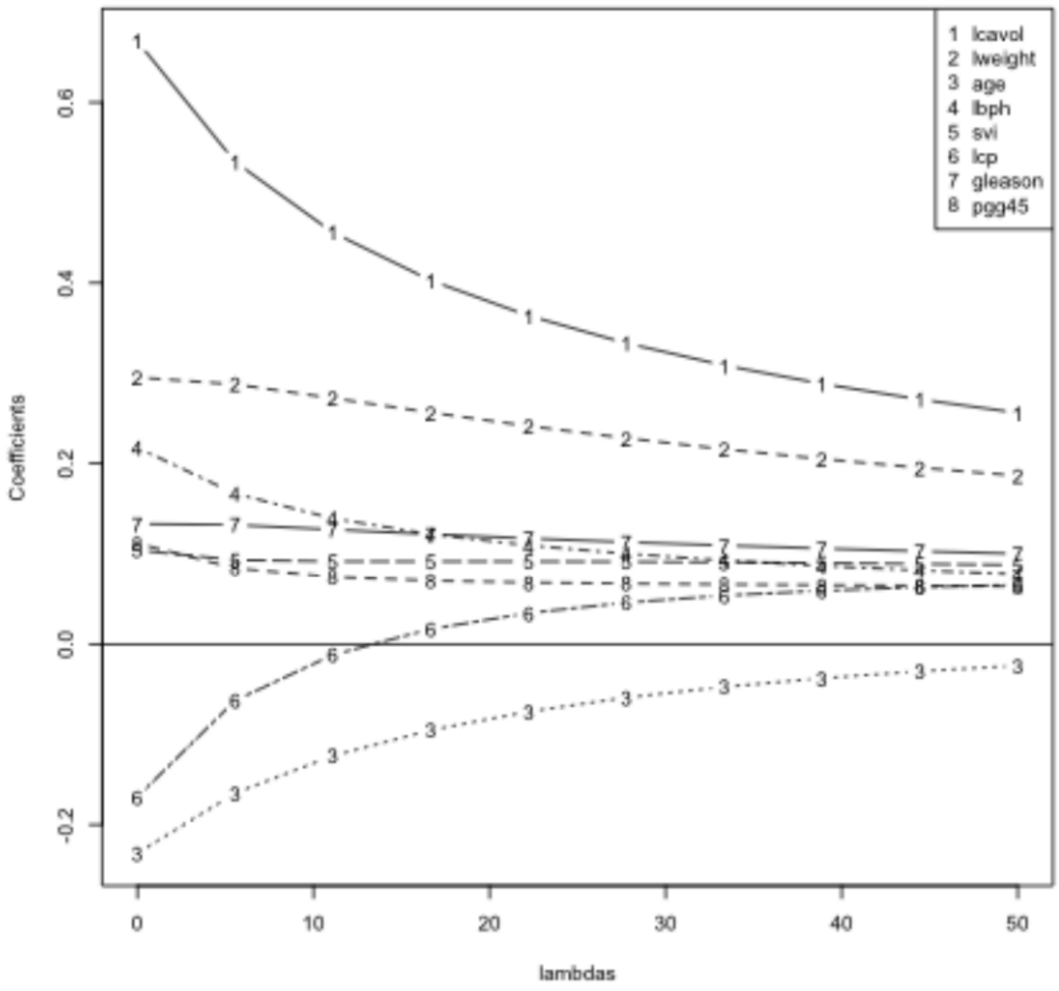


Figure 3: Association between λ and coefficients in regularised regression.

1.7.1.1 Tuning parameter λ

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

1.7.2 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

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

but not in general.

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

1.8 Notes and further reading

- Hector Corrada Bravo's Practical Machine Learning lecture notes
- Hector's penalized regression reading list
- Elements of Statistical Learning (Hastie et al., 2009)
- In `caret` methods are:
  - `ridge`
  - `lasso`
  - `relaxo`

---

2 Lecture 2: Combining predictors (ensembling)

This lecture is about combining predictors (sometimes called **ensembling methods** in learning).

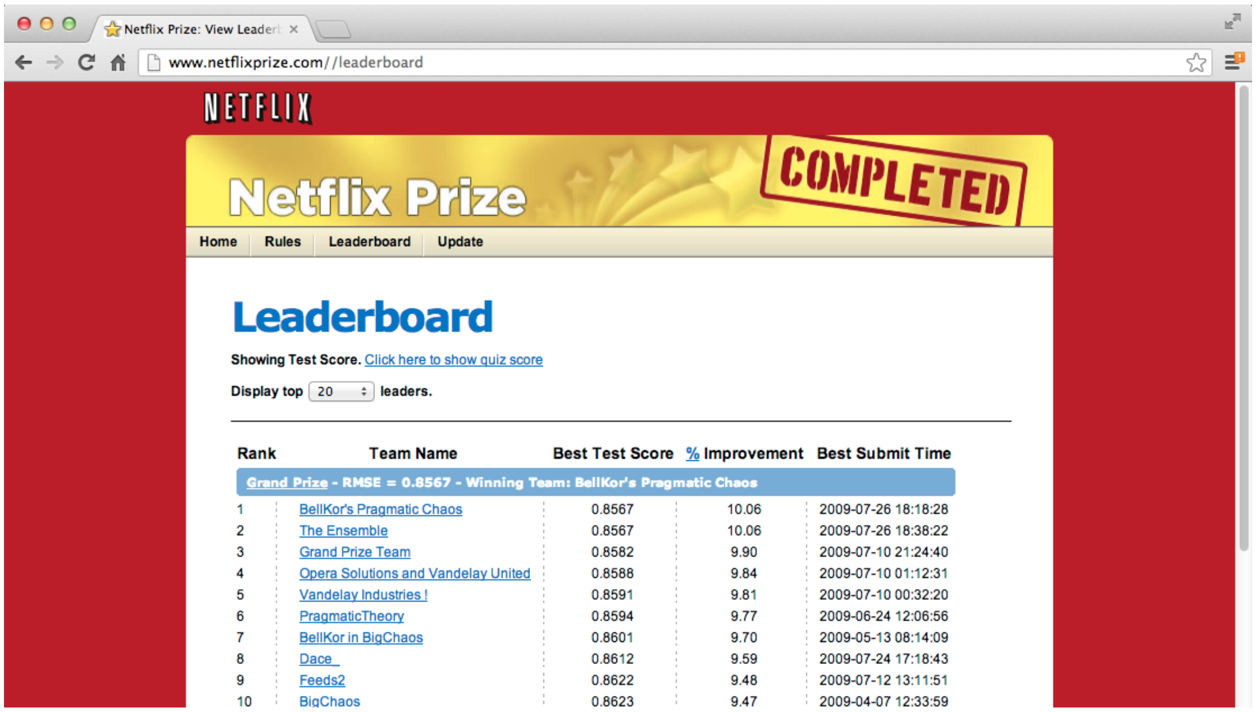
2.1 Key ideas

- You can **combine classifiers by averaging/voting**
  - these can be classifiers that are very different; for example you can combine a boosting classifier with a random forest with a linear regression model
- Combining classifiers **improves accuracy**
- Combining classifiers **reduces interpretability**
- Boosting, bagging, and random forests are variants on this theme

2.1.1 Example: Netflix prize

BellKor = Combination of 107 predictors

```r
knitr::include_graphics("netflix.png")
```


Netflix Prize: View Leaderb
www.netflixprize.com//leaderboard
NETFLIX
Netflix Prize
COMPLETED
Home
Rules
Leaderboard
Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59

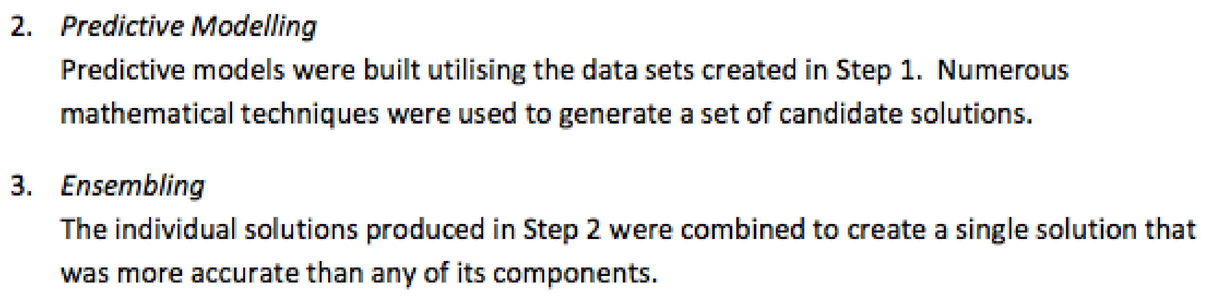
Figure 4: BellKor combined 107 predictors to produce the most accurate predictin of viewers' preferences and will the 1 million-dollar prize. See https://www.netflixprize.com/leaderboard.html.

2.1.2 Example: Heritage health prize - Progress Prize 1

The Heritage Health prize was a $3 million prize. It was designed to try to predict whether people would go back to the hospital based on their hospitalization record.

Market Makers

```r
knitr::include_graphics("makers.png")
```



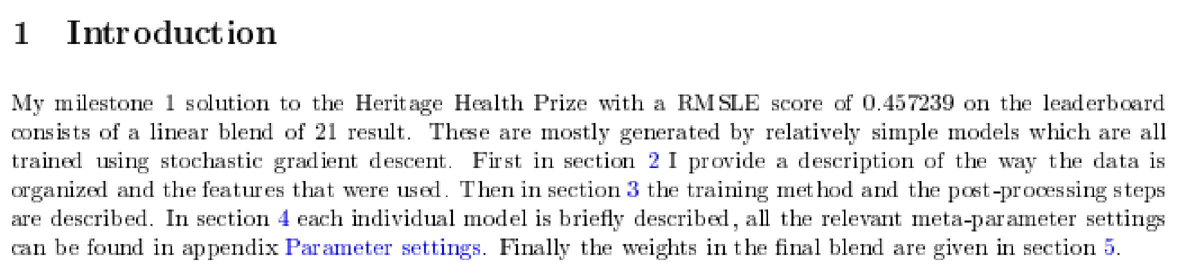
2. *Predictive Modelling*
   Predictive models were built utilising the data sets created in Step 1. Numerous mathematical techniques were used to generate a set of candidate solutions.

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.

Figure 5: Market Makers.

Mestrom

```r
knitr::include_graphics("mestrom.png")
```


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.

Figure 6: Mestrom.

2.2 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

2.3 Approaches for combining classifiers

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

2.4 Example: `Wage` data

Create training, test and validation sets

```
library(ISLR)
data(Wage)

library(ggplot2)
library(caret)

Wage <- subset(Wage,
               select = -c(logwage)) #logwage is too similar to wage

# 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)
# Create traininf and testing data sets from the building set
training <- buildData[inTrain,]
```

```
testing <- buildData[-inTrain,]

dim(training)
```

```
## [1] 1474   10
```

```
dim(testing)
```

```
## [1] 628  10
```

```
dim(validation)
```

```
## [1] 898  10
```

2.4.1 Build 2 different models

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

2.4.2 Predict on the testing set

Both models predictions are close to each other, but they do not perfectly agree with each other (much less in the lecture example).

And, neither of them perfectly correlates with the wage variable, which is the colour of the dots in Fig 7.

```
pred1 <- predict(mod1, testing)
pred2 <- predict(mod2, testing)

library(ggpubr)
```

```
## Loading required package: ggplot2
```

```
ggplot(testing, aes(x = pred1, y = pred2, colour = wage)) +
  geom_point(alpha = 0.3) +
  stat_cor(aes(label = paste(..rr.label..,
                             cut(..p..,
                                 breaks = c(-Inf,
                                            0.0001,
                                            0.001,
                                            0.01,
                                            0.05,
                                            Inf),
                                 labels = c("'****'",
                                            "'***'",
                                            "'**'",
                                            "'*'",
                                            "''")),
                             sep = "~")),
           label.y.npc = 0.9,
          color = "black") +
  theme_pubclean()
```

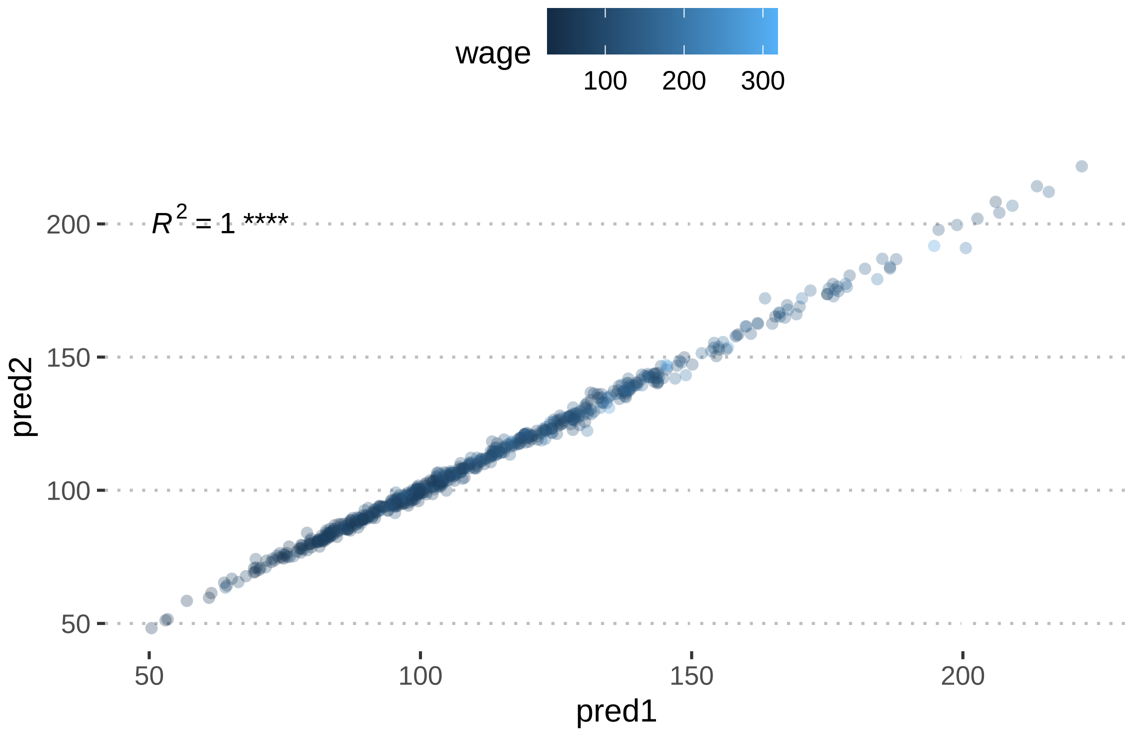


Figure 7: Association between model 1 and model 2.

2.4.3 Combining predictors

Create a data frame that combines the predictions from both models, and fit a new model from the predictions of the original models.

```
library(caret)
```

```
## Loading required package: lattice
```

```
predDF <- data.frame(pred1, pred2, wage = testing$wage)

combModFit <- train(wage ~.,
                    meyhod = "gam",
                    data = predDF)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

```
combPred <- predict(combModFit, predDF)
```

2.4.4 Testing errors

The error of the combined model, has a smaller error than any of the two models.

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

```
## [1] 827.2651
```

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

```
## [1] 826.0142
```

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

```
## [1] 388.155
```

2.4.5 Predict on validation data set

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

2.4.6 Evaluate on validation

This should be smaller using the combined model (it is on the lecture), but not in this case.

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

```
## [1] 1132.996
```

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

```
## [1] 1131.133
```

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

```
## [1] 1232.234
```

2.5 Notes and further resources

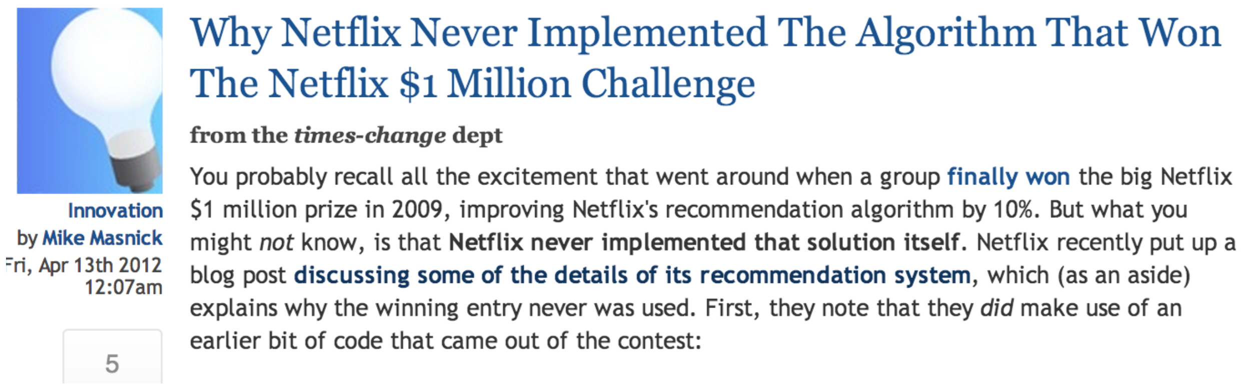
- Even simple blending can be useful
- Typical model for binary/multiclass data
  - Build an odd number of models
  - Predict with each model
  - Predict the class by majority vote
- This can get dramatically more complicated
  - Simple blending in caret: caretEnsemble (use at your own risk!)
  - Wikipedia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

2.6 Recall - scalability matters

A problem with ensembling is that this can lead to increases in computational complexity.

So it turns out that even though Netflix paid a million dollars to the team that won the prize, the Netflix million-dollar solution was never actually implemented, because it was too computational intensive to apply to specific data sets.

```r
knitr::include_graphics("netflixno.png")
```


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

from the *times-change* dept

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

5

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:

Figure 8: The Netflix million-dollar solution was never actually implemented.

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

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

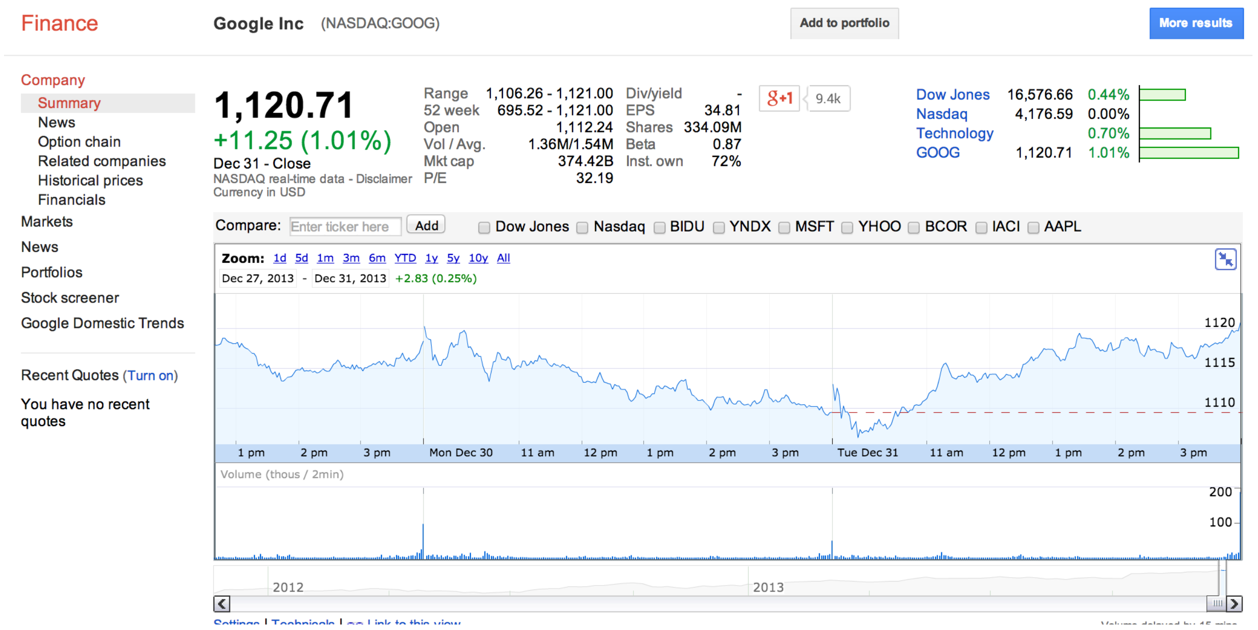
---

3 Lecture 3: Forecasting

This lecture is about forecasting, which is a very specific kind of prediction problem. And it's typically applied to things like time series data.

For example, this is the stock of information for Google on the NASDAQ. You can see over time that there's a price for this stock and it goes up and down (see Fig. 9). This introduces some very specific kinds of dependent structure and some additional challenges that must be taken into account when performing prediction.

```
knitr::include_graphics("GOOG.png")
```



Finance | Google Inc (NASDAQ:GOOG) | Add to portfolio | More results

Company
Summary
News
Option chain
Related companies
Historical prices
Financials
Markets
News
Portfolios
Stock screener
Google Domestic Trends

Recent Quotes (Turn on)

You have no recent quotes

1,120.71
+11.25 (1.01%)
Dec 31 - Close
NASDAQ real-time data - Disclaimer
Currency in USD

Range	1,106.26 - 1,121.00	Div/yield	-
52 week	695.52 - 1,121.00	EPS	34.81
Open	1,112.24	Shares	334.09M
Vol / Avg.	1.36M/1.54M	Beta	0.87
Mkt cap	374.42B	Inst. own	72%
P/E	32.19		

Dow Jones	16,576.66	0.44%
Nasdaq	4,176.59	0.00%
Technology		0.70%
GOOG	1,120.71	1.01%

Compare: Enter ticker here Add Dow Jones Nasdaq BIDU YNDX MSFT YHOO BCOR IACI AAPL

Zoom: 1d 5d 1m 3m 6m YTD 1y 5y 10y All
Dec 27, 2013 - Dec 31, 2013 +2.83 (0.25%)

Figure 9: Example of time series data. See https://www.google.com/finance.

3.1 What is different?

- Data are dependent over time
  - More challenging than when you have independent examples
- 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 (over a period that's longer than a year, for example)
- Subsampling into training/test is more complicated
  - you can't just randomly assign samples into training and test. You have to take advantage of the fact that there's actually specific times that are being sampled and that points are dependent in time
- 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!)

3.2 Beware spurious correlations!

One thing to be aware of is that you have to be careful of spurious correlations: time series can often be correlate for reasons that do not make them good for predicting one from the other (e.g. Fig. 10).

```r
knitr::include_graphics("spurious.jpg")
```

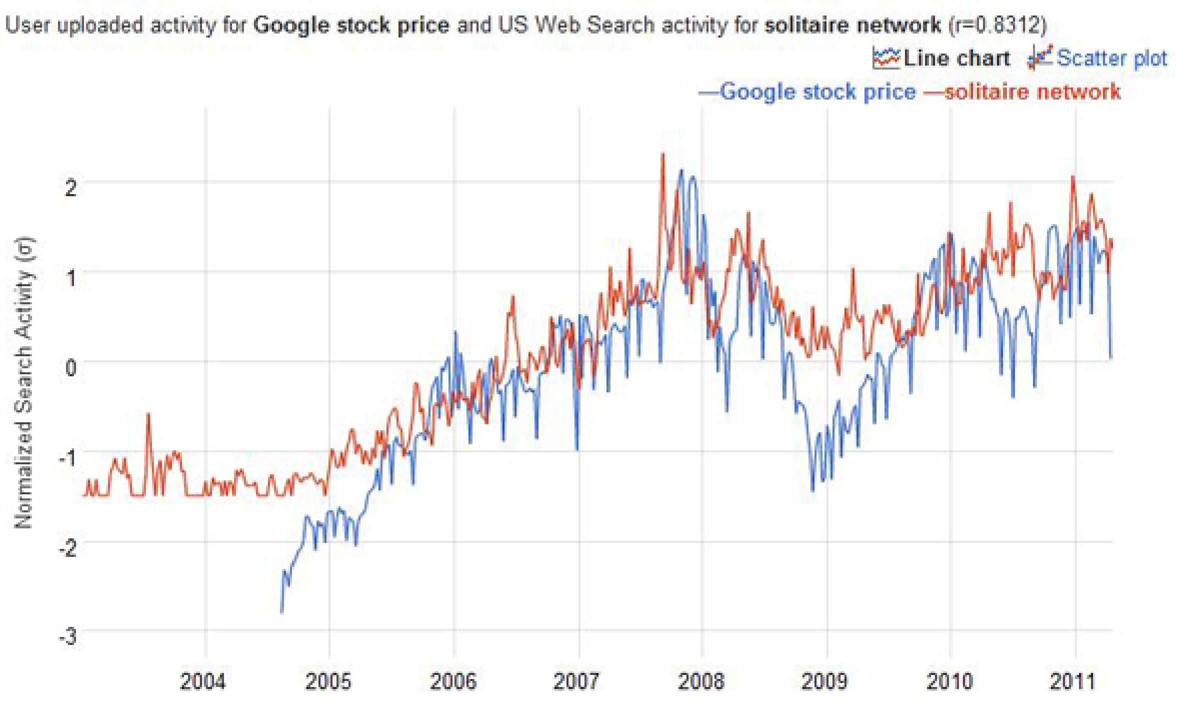


Figure 10: Example of spurious correlation in time series data. See http://www.google.com/trends/correlate.

3.2.1 Also common in geographic analyses

Fig 11 is a cartoon from *xkcd* that shows that heat maps particularly population-based heat maps had very similar shapes because of the place where many people live.

```r
knitr::include_graphics("heatmap.png")
```

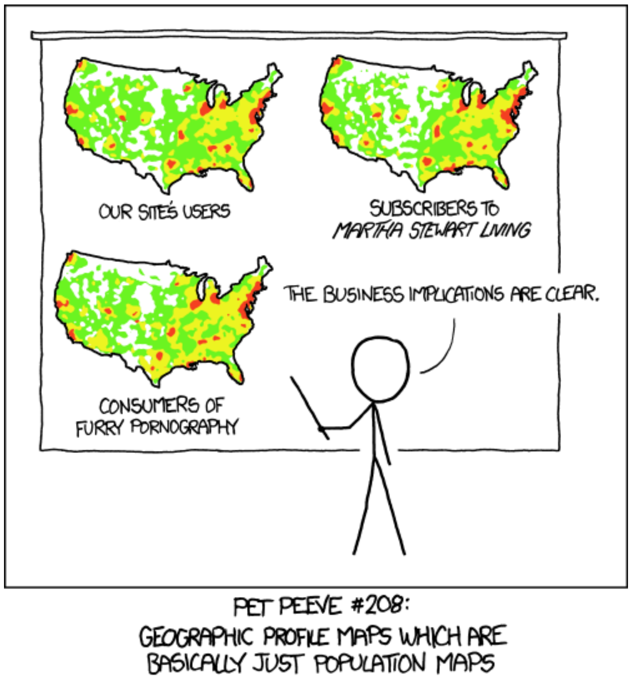

OUR SITE'S USERS
SUBSCRIBERS TO MARTHA STEWART LIVING
CONSUMERS OF FURRY PORNOGRAPHY
THE BUSINESS IMPLICATIONS ARE CLEAR.
PET PEEVE #208: GEOGRAPHIC PROFILE MAPS WHICH ARE BASICALLY JUST POPULATION MAPS

Figure 11: Example of spurious correlation in time series data. See http://www.google.com/trends/correlate.

For example, the users of a particular site or the subscribers to a particular magazine or the consumers of a particular type of website may all appear in the very similar places because the highest density in population in the United States.

3.3 Beware extrapolation!

Fig. 12 (Fig. 1 from Tatem et al., 2004) is a kind of a funny example that shows what happens if you extrapolate time series out without being careful about what could happen.

```r
knitr::include_graphics("extrapolation.jpg")
```

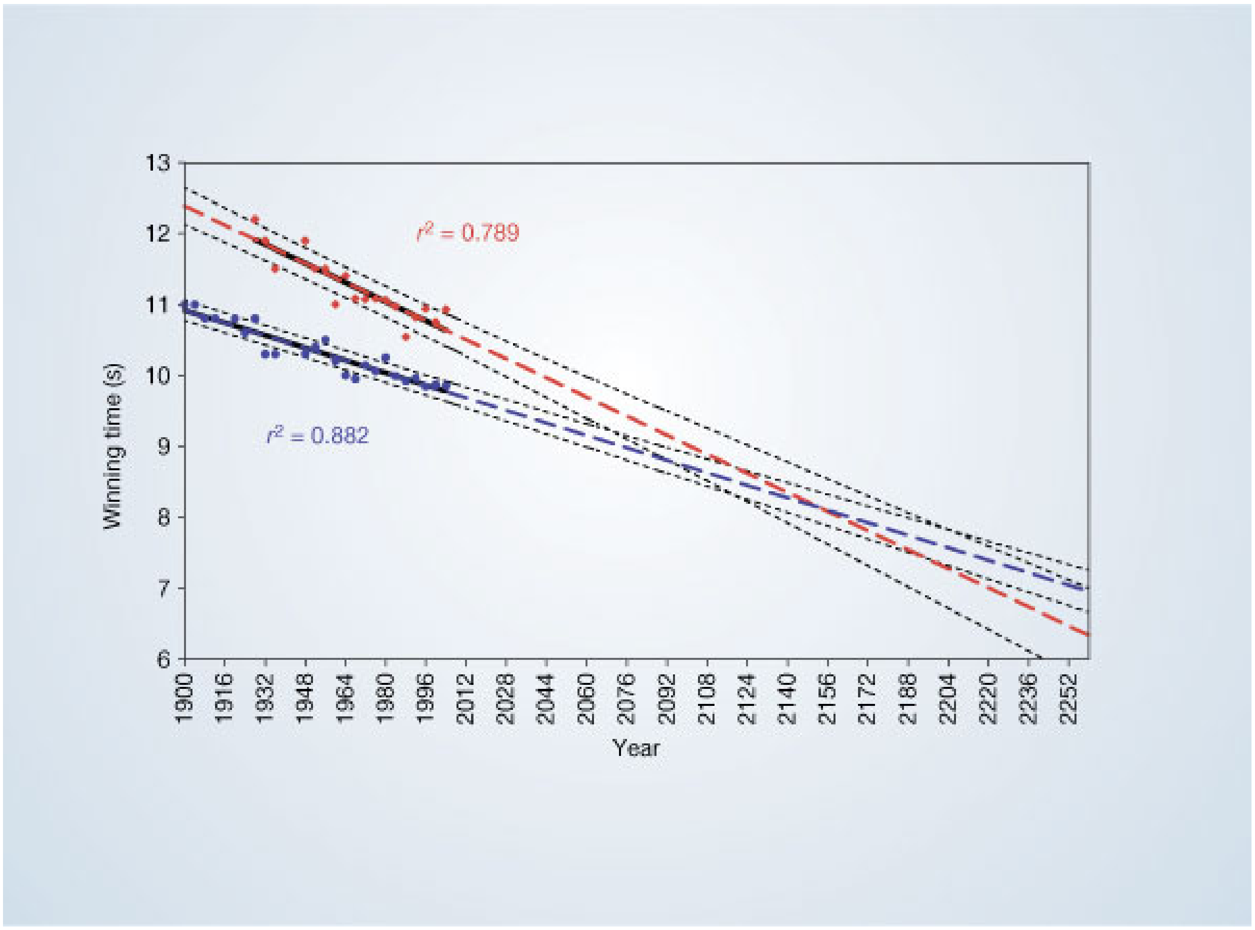


Figure 12: Example of a ridiculuos extrapolation of a long scale the winning time of a large number of races that occurred at the Olympics. The extcolorblueblue times are men and the extcolorredred times are women. Taken from Tatem et al. (2004).

The authors of this paper extrapolated out into the future and said that in 2156 that would be when women would run faster than men in the sprint. And while we don't know when that may or may not occur, one thing that was pointed out is that this kind of extrapolation is very dangerous.

Eventually at some time in the future, both men and women will be predicted to run negative times for the 100 meters.

3.4 Example: Google data

Example of some forecasting using the `quantmod` package and some Google data.

If I load this `quantmod` package and I can load in a bunch of data from the Google stock symbol, and from the Google finance data set.

So, if I look at this Google variable, I get the open, high, low, close, and volume information for a particular Google stock from the 1st of January, 2008 to December 31st, 2013.

```
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
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 = "yahoo", from = from.dat, to = to.dat)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## [1] "GOOG"
```

```
getSymbols("GOOGL", src = "yahoo", from = from.dat, to = to.dat)
```

```
## [1] "GOOGL"
```

```
GOOG <- GOOG + GOOGL
GOOG$GOOG.Volume <- GOOGL$GOOGL.Volume/2
head(GOOG)
```

```
##            GOOG.Open GOOG.High GOOG.Low GOOG.Close GOOG.Volume GOOG.Adjusted
## 2008-01-02  691.9231  696.4170 676.8038   684.2536     4302593      684.2536
## 2008-01-03  684.3235  685.9113 675.5955   684.3934     3249248      684.3934
## 2008-01-04  678.7611  680.0294 654.1049   656.1021     5354440      656.1021
## 2008-01-07  653.0463  661.3749 636.4790   648.3627     6396997      648.3627
## 2008-01-08  652.1076  659.0581 630.1377   630.8167     5333761      630.8167
## 2008-01-09  629.1790  652.4471 621.6592   652.3073     6732961      652.3073
```

3.4.1 Summarise 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")
```

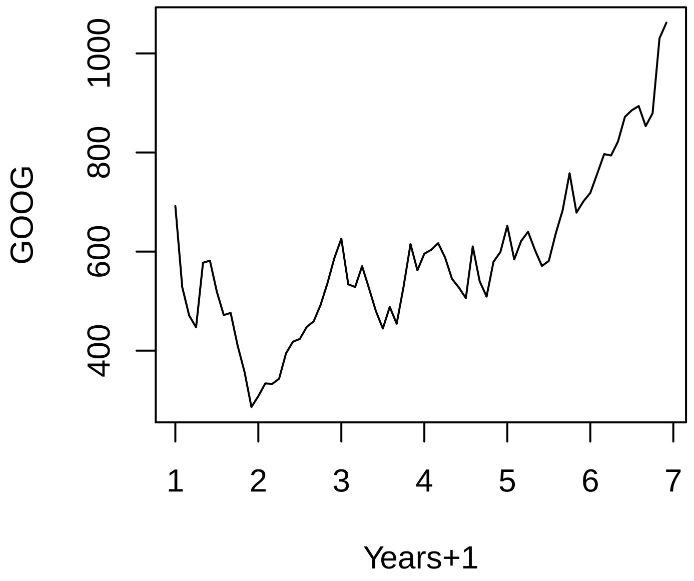


Figure 13: Time series plot .

Plotting time series in ‘ggplot2‘.

```
library(ggplot2)
mGoog2 <- as.data.frame(GOOG)
mGoog2$date <- as.Date(row.names(mGoog2))
ggplot(mGoog2, aes(x = date, y = GOOG.Open, colour = GOOG.Open)) +
  geom_line() +
  labs(x = "Year", y = "GOOG", color = "GOOG") +
  theme_pubclean()
```

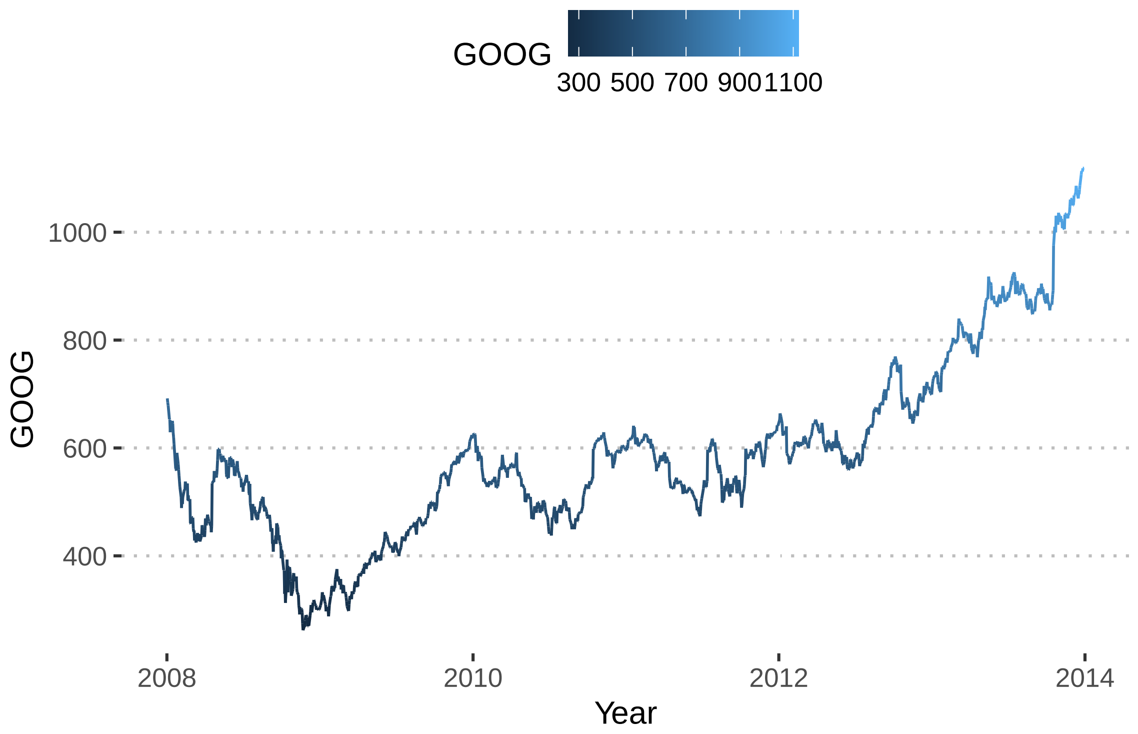


Figure 14: Time series plot in `ggplot2`.

Or in `ggplot2`, by using `ggfortify::autoplot` (to use the exact same data as in Fig. 13) from a `ts` object.

```
library(ggfortify)
autoplot(ts1) +
  theme_pubclean()
```

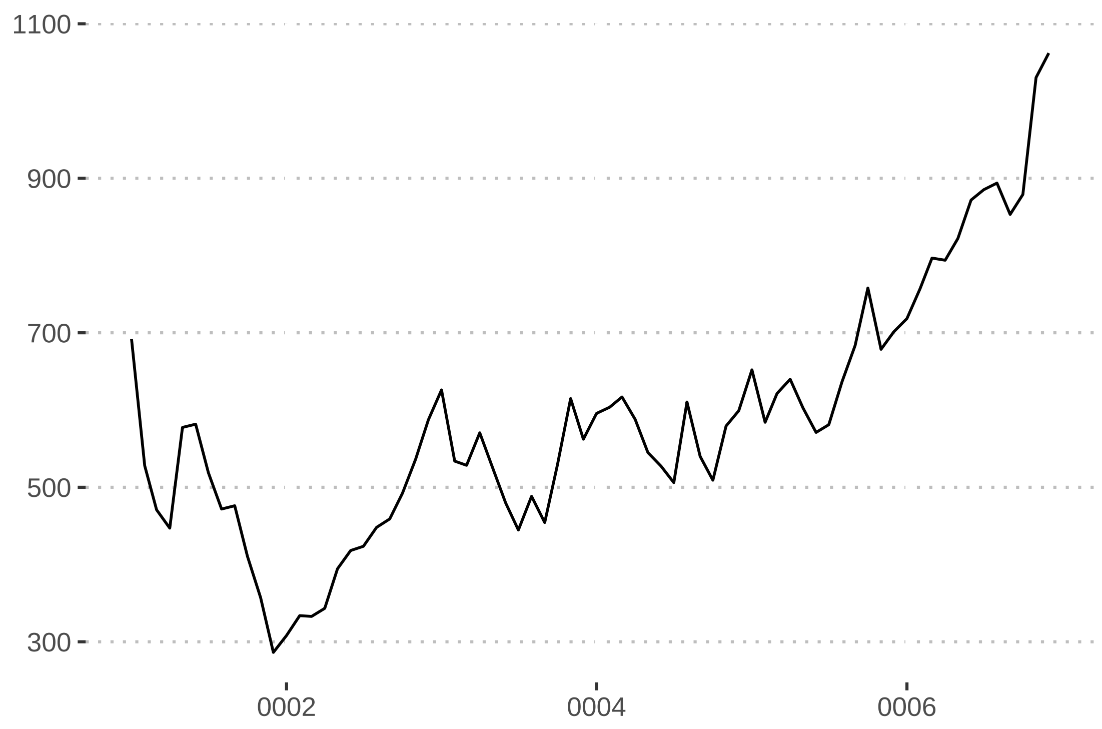


Figure 15: Time series plot in `ggplot2` using `ggfortify`.

3.4.2 Example time series decomposition

For more info, check <https://www.otexts.org/fpp/6/1> (Hyndman & Athanasopoulos, 2018)

- **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

3.4.2.1 Decompose a time series into parts If I decompose this in an additive way, then I can see that there's a trend variable that appears to be an upward trend of the Google stock price (Figs. 16 and 17).

There also appears to be a seasonal pattern, as well as a more of a random cyclical pattern in the data set. So this is decomposing this series here into a series of different types of patterns in the data.

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

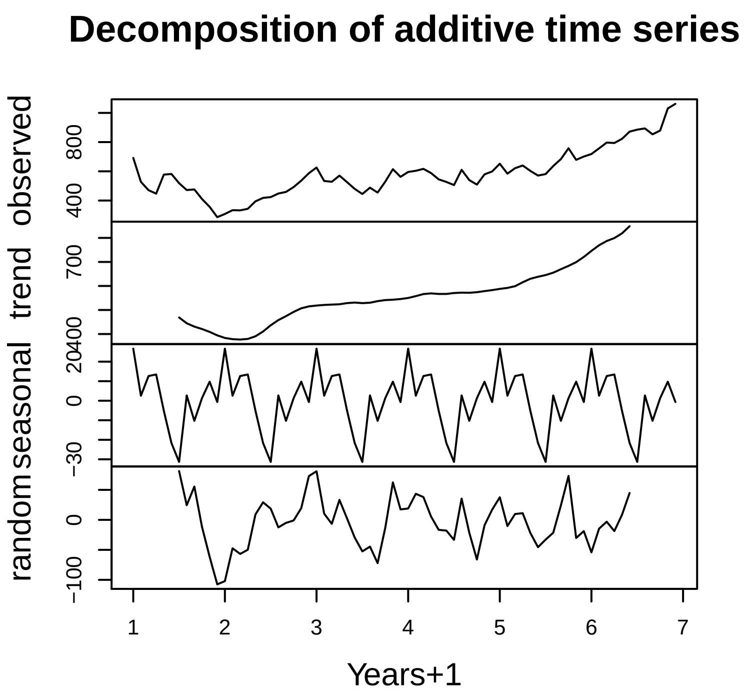


Figure 16: Decomposed time series plot.

Or in `ggplot2`, by using `autoplot`.

```r
ts1 %>%
  decompose() %>%
  autoplot() +
  theme_pubclean()
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```
## Warning: Removed 24 row(s) containing missing values (geom_path).
```

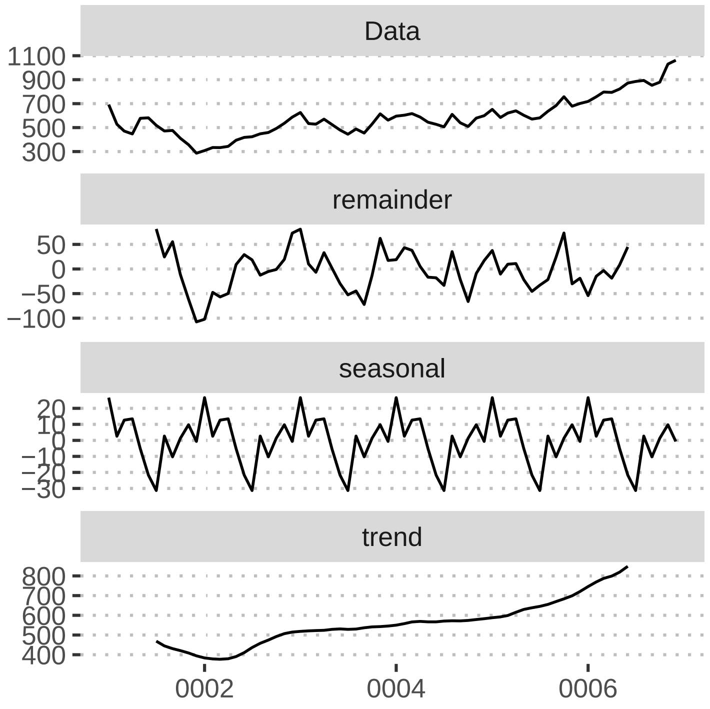


Figure 17: Decomposed time series plot in `ggplot2`.

3.4.3 Training and test sets

I have to build training and test sets that have consecutive time points. So here I am building a training set that starts at time point 1 and ends at time point 5. And then a test set that is the next consecutive sets of points after that.

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

```
## Warning in window.default(x, ...): 'end' value not changed
```

```
ts1Train
```

```
##          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1 691.9231 527.9475 470.8656 447.1281 577.5197 581.7039 518.8699 471.8643
## 2 308.1783 333.8332 332.8745 343.3102 394.4901 418.1577 423.6203 448.1267
## 3 626.0932 533.8694 528.4768 570.5692 525.7805 479.7734 444.6815 488.3217
## 4 595.6649 603.6639 616.9357 587.9554 544.9542 527.3184 506.0475 610.3847
## 5 652.0477
##          Sep      Oct      Nov      Dec
## 1 476.1184 410.5881 357.0913 286.2882
## 2 459.0518 492.3263 536.3460 587.3262
## 3 454.3582 529.2757 614.8886 562.2306
## 4 540.0110 509.1532 579.3072 599.1800
## 5
```

3.4.4 Simple moving average

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

```r
library(forecast)
```

```
## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify

##
## Attaching package: 'forecast'

## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

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

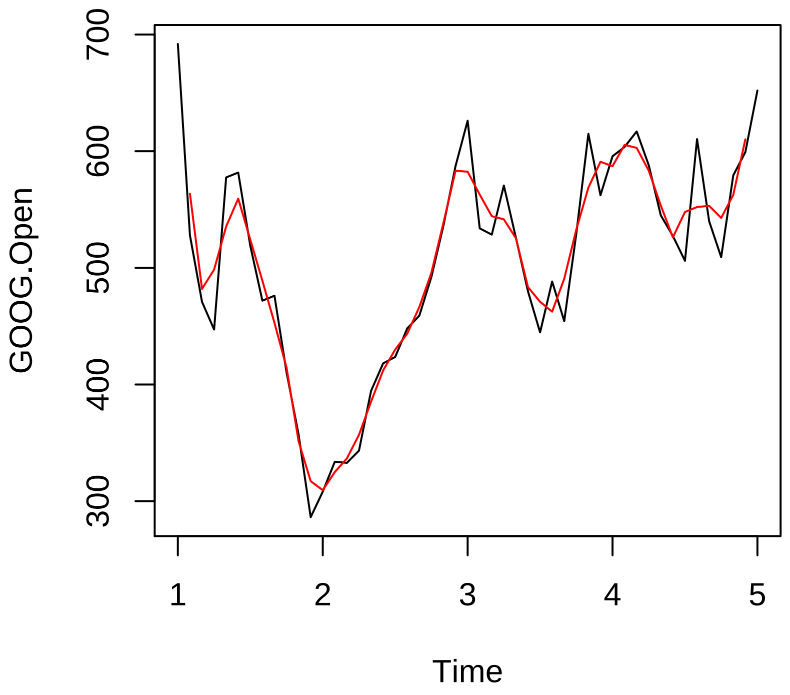


Figure 18: Moving average plot in the training set.

Or in ggplot2.

```r
ts1Train %>%
  autoplot() +
  geom_line(aes(y = ma(ts1Train, order = 3)),
            color = "red") +
  theme_pubclean()
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```

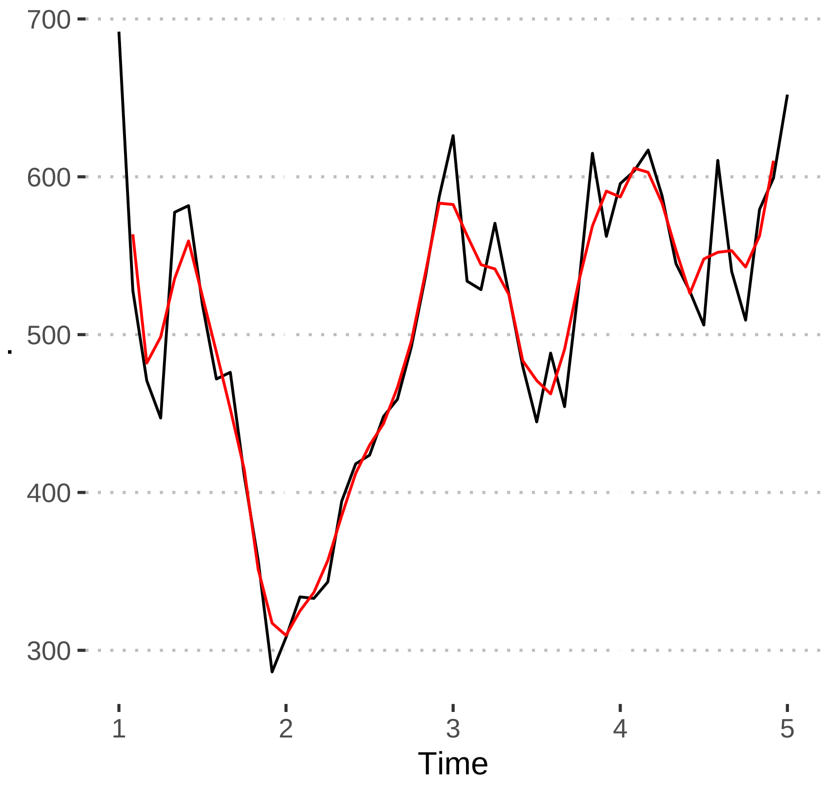


Figure 19: Moving average plot in the training set in `ggplot2`.

3.4.5 Exponential smoothing

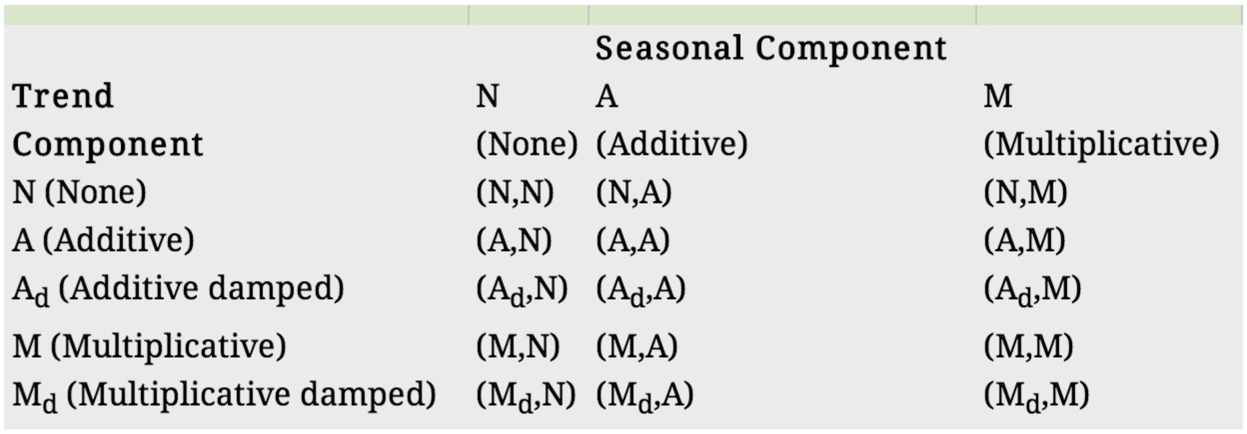
Example - simple exponential smoothing

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

Basically, we weight near-by time points as higher values or by more heavily than time points that are farther away.

There is a large number of different classes of smoothing models that you can choose (Fig. 20):

```
knitr::include_graphics("expsmooth.png")
```



		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)

Figure 20: Example of exponential smoothing. .

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

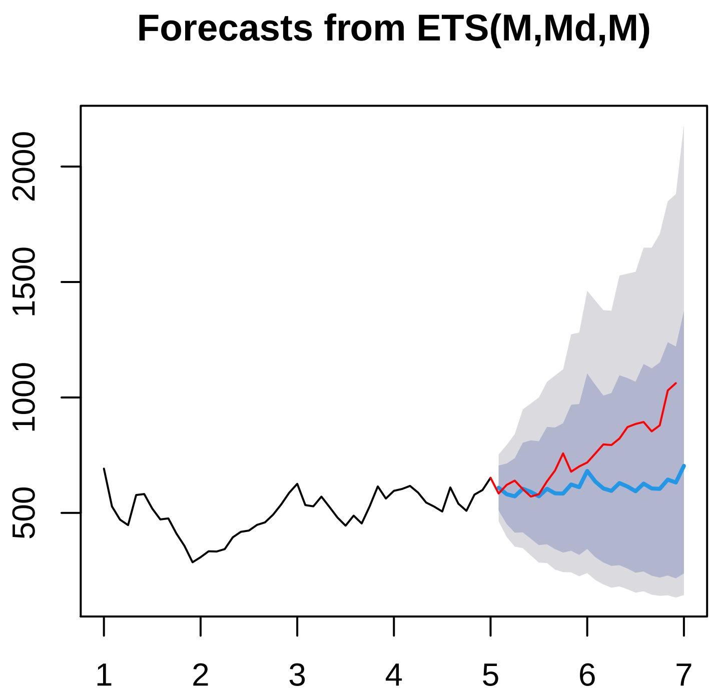


Figure 21: Exponential smoothing. Forecast on the training data set in blue, and forecast on the test data set in red.

Or in `ggplot2`.

```r
fcast %>%
  autoplot() +
  autolayer(ts1Test, color = "red") +
  theme_pubclean() +
  theme(legend.position = "none")
```

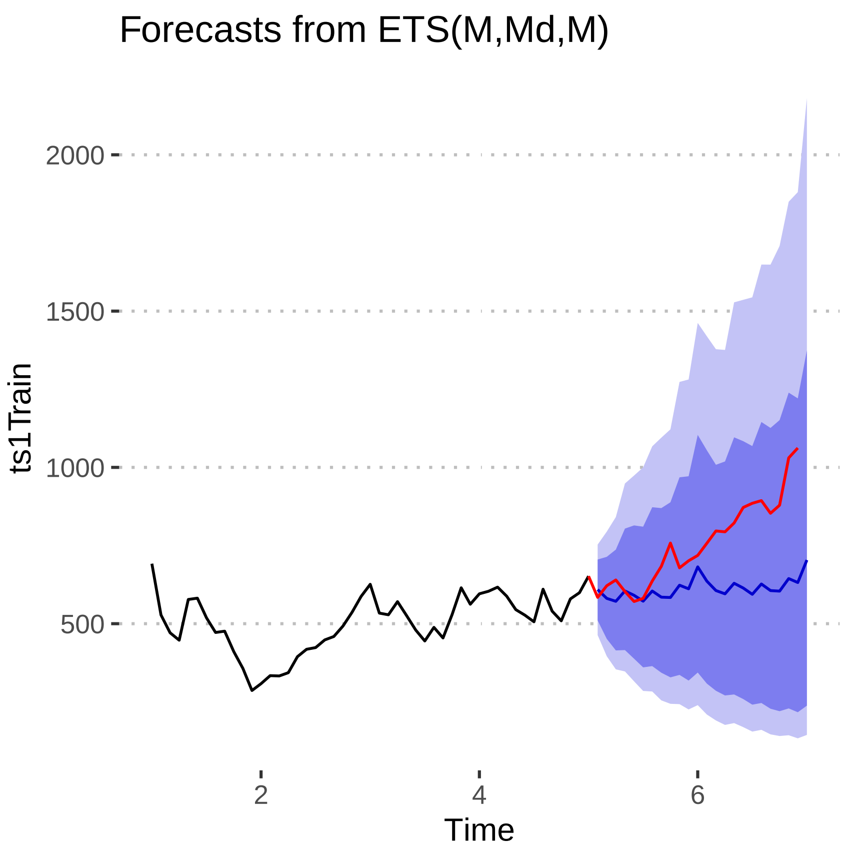


Figure 22: Exponential smoothing plot in `ggplot2`. Forecast on the training data set in blue, and forecast on the test data set in red.

3.4.6 Get the accuracy

You can get the accuracy of your forecast using your test set, and it will give you root mean square to error and other metrics that are more appropriate for forecasting.

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

```
##                      ME      RMSE       MAE        MPE      MAPE      MASE
## Training set  -1.969684  51.27863  41.70078 -0.9271222  8.370485 0.3960809
## Test set     148.481075 195.80989 152.52015 17.3090760 18.006488 1.4486616
##                    ACF1 Theil's U
## Training set 0.02356608        NA
## Test set     0.75601592  3.549695
```

3.5 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 (Hyndman & Athanasopoulos, 2018)
- 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.

---

4 Lecture 4: Unsupervised Prediction

So far, in the examples we have talked about, you know what the labels are. In other words, you're trying to do supervised classification: you're trying to predict an outcome that you know what it is.

4.1 Key ideas

- Sometimes you don't know the labels for prediction
- To build a predictor
  - Create clusters
    - It is not a perfectly noiseless process
  - Name clusters
    - Coming up with the right names (interpreting the clusters well) is an incredibly challenging problem
  - Build predictor for clusters
    - Using the algorithms that we have learned, as well as predicting on those clusters
- In a new data set
  - Predict clusters
    - (and apply the name that you come up with in the previous data set)

4.2 Example: Iris dataset ignoring species labels

```r
data(iris)
library(ggplot2)
library(caret)

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

```
## [1] 105   5
```

```r
dim(testing)
```

```
## [1] 45  5
```

4.2.1 Cluster with k-means

I could perform a k-means clustering.

If you remember that k-means clustering from the exploratory data analysis section of the data science specialization. And the basic idea here is to basically create three different clusters. So I was telling it to create three different clusters, ignoring the species information.

Here, we will create 3 clusters, ignoring the species.

```
kMeans1 <- kmeans(subset(training,
                         select= -c(Species)), #ignore the species
                  centers = 3) #create 3 clusters

training$clusters <- as.factor(kMeans1$cluster)

panA <- qplot(x = Petal.Width, y = Petal.Length,
              colour = clusters,
              data = training) +
  labs(title = "Predicted clustering (species) from k-means") +
  theme_pubclean()

panB <- qplot(x = Petal.Width, y = Petal.Length,
              colour = Species,
              data = training) +
  labs(title = "Actual clustering (species)") +
  theme_pubclean()

ggarrange(panA, panB,
          labels = "AUTO")
```

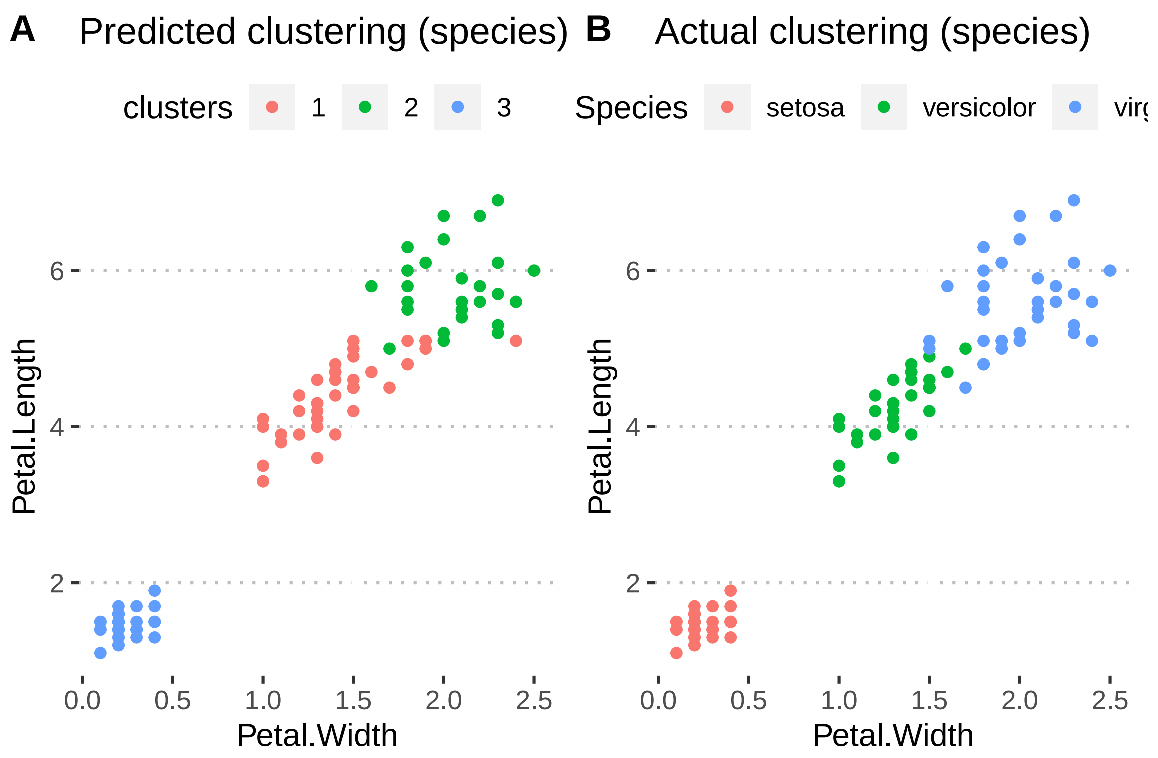


Figure 23: Comparison of k-clusterin prediction os Iris species, and actual species. **A** Predicted clusters (species). **B** Actual clusters (species).

Compare to real labels

I wouldn't know what those species names were, and I would have to come up with names for each of my clusters. However, in this case:

- Cluster 1 = versicolor
- Cluster 2 = setosa
- Cluster 3 = virginica

```
library(kableExtra)
kable(table(kMeans1$cluster,
            training$Species),
      booktabs = TRUE,
      row.names = TRUE) %>%
  kable_styling(position = "center",
                latex_options = "HOLD_position")
```

	setosa	versicolor	virginica
1	0	34	9
2	0	1	26
3	35	0	0

4.2.2 Build predictor

Then I can fit a model that relates the cluster variable that I have created, to all the predictor variables in the training set.

In this case, I am doing it with a classification tree (`method = "rpart"`). I can then do a prediction in a, a training set.

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

kable(table(predict(modFit,
                    training),
            training$Species),
      booktabs = TRUE,
      row.names = TRUE) %>%
  kable_styling(position = "center",
                latex_options = "HOLD_position")
```

	setosa	versicolor	virginica
1	0	35	10
2	0	0	25
3	35	0	0

It did a reasonably good job of predicting clusters 1 and 2, but cluster 1 and cluster 3 sometimes get mix, mixed up (for virginica) in my prediction model.

This is because I have both error or variation in my prediction building and error and variation in my cluster building, so it ends up being a quite a challenging problem to do unsupervised prediction in this way.

4.2.3 Apply on test

If I predict on the test data set, in general I wouldn't know what the labels are, but here I'm showing what the labels are.

```r
testClusterPred <- predict(modFit,
                           testing)

kable(table(testClusterPred,
            testing$Species),
      booktabs = TRUE,
      row.names = TRUE) %>%
  kable_styling(position = "center",
                latex_options = "HOLD_position")
```

	setosa	versicolor	virginica
1	0	15	6
2	0	0	9
3	15	0	0

Here I'm predicting on a new data set and making a table versus the actual known species. And so I can see this actually does quite a reasonable job here of predicting the different species into different clustered labels.

4.3 Notes and further reading

- The `cl_predict` function in the `clue` package provides similar functionality
- Beware over-interpretation of clusters!
  - This is in fact an exploratory technique. And so the clusters may change depend on the way that you sample the data
- This is one basic approach to recommendation engines
- Elements of statistical learning (**hastie2009?**)
- Introduction to statistical learning (James et al., 2013)

References

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed). Springer.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (Second). OTexts. https://otexts.com/fpp2/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. Springer. https://www.statlearning.com/s/ISLR-Seventh-Printing-xwa7.pdf

Tatem, A. J., Guerra, C. A., Atkinson, P. M., & Hay, S. I. (2004). Momentous sprint at the 2156 Olympics? *Nature*, *431*(7008), 525–525. https://doi.org/10.1038/431525a