



Motivation and pre-requisites

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

About this course

- This course covers the basic ideas behind machine learning/prediction
 - Study design - training vs. test sets
 - Conceptual issues - out of sample error, ROC curves
 - Practical implementation - the caret package
- What this course depends on
 - The Data Scientist's Toolbox
 - R Programming
- What would be useful
 - Exploratory analysis
 - Reporting Data and Reproducible Research
 - Regression models

Who predicts?

- Local governments -> pension payments
- Google -> whether you will click on an ad
- Amazon -> what movies you will watch
- Insurance companies -> what your risk of death is
- Johns Hopkins -> who will succeed in their programs

Why predict? Glory!



<http://www.zimbio.com/photos/Chris+Volinsky>

Why predict? Riches!



**Improve Healthcare,
Win \$3,000,000.**

COMPETITION GOAL

Identify patients who will be admitted to a hospital within the next year, using historical claims data.

<http://www.heritagehealthprize.com/c/hhp>

Why predict? For sport!



[Sign Up](#) [About](#) [Hosting Center](#) [All Competitions](#) [Users](#) [Forums](#) [Wiki](#) [Blog](#) [Data Science Jobs](#)

What's in your data?

Participate in competitions

Kaggle is an arena where you can match your data science skills against a global cadre of experts in statistics, mathematics, and machine learning. Whether you're a world-class algorithm wizard competing for prize money or a novice looking to learn from the best, here's your chance to jump in and geek out, for fame, fortune, or fun.

[Join as a participant](#)

(Need convincing?)

Create a competition

Kaggle is a platform for data prediction competitions that allows organizations to post their data and have it scrutinized by the world's best data scientists. In exchange for a prize, winning competitors provide the algorithms that beat all other methods of solving a data crunching problem. Most data problems can be framed as a competition.

[Learn more about hosting](#)

<http://www.kaggle.com/>

Why predict? To save lives!

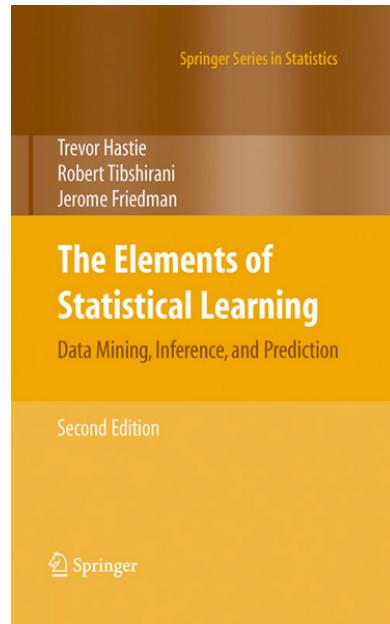
Oncotype DX® reveals
the underlying biology that
changes treatment decisions
37% of the time

Uncover the Unexpected™



<http://www.oncotypedx.com/en-US/Home>

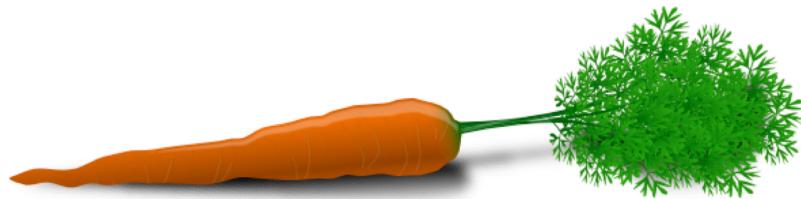
A useful (if a bit advanced) book



[The elements of statistical learning](#)

A useful package

the caret package



The **caret** package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

Links

[train Model List](#)

Topics

[Main Page](#)

[Data Sets](#)

[Visualizations](#)

[Pre-Processing](#)

<http://caret.r-forge.r-project.org/>

Machine learning (more advanced material)

Stanford Machine Learning

Andrew Ng

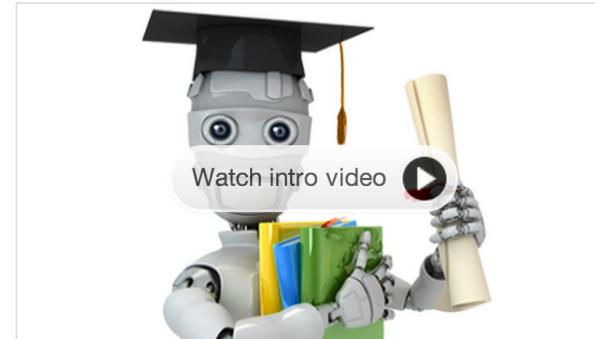
Taught In: English

Subtitles Available In: English

Sessions:

Oct 14th 2013 (10 weeks long) ▾

Learn for Free



3,794

12k

14k

Tweet

+1

Like

<https://www.coursera.org/course/ml>

Even more resources

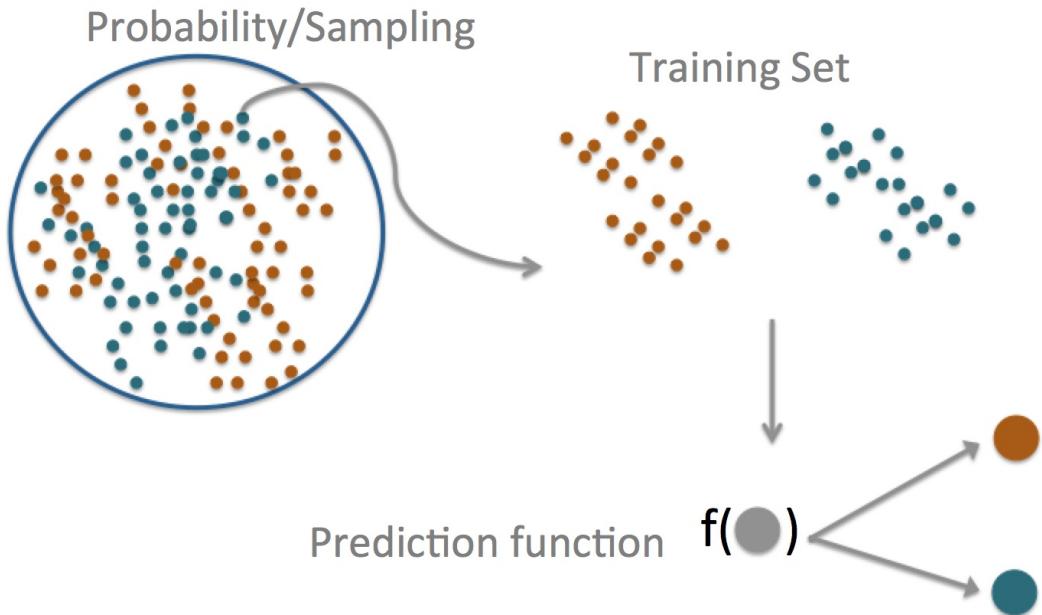
- [List of machine learning resources on Quora](#)
- [List of machine learning resources from Science](#)
- [Advanced notes from MIT open courseware](#)
- [Advanced notes from CMU](#)
- [Kaggle - machine learning competitions](#)



What is prediction?

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

The central dogma of prediction



What can go wrong

BIG DATA

The Parable of Google Flu: Traps in Big Data Analysis

David Lazer,^{1,2*} Ryan Kennedy,^{1,3,4} Gary King,³ Alessandro Vespignani^{5,6,3}

In February 2013, Google Flu Trends (GFT) made headlines but not for a reason that Google executives or the creators of the flu tracking system would have hoped. *Nature* reported that GFT was predicting more than double the proportion of doctor visits for influenza-like illness (ILI) than the Centers for Disease Control and Prevention (CDC), which bases its estimates on surveillance reports from laboratories across the United States (1, 2). This happened despite the fact that GFT was built to predict CDC reports. Given that GFT is often held up as an exemplary use of big data (3, 4), what lessons can we draw from this error?

The problems we identify are not limited to GFT. Research on whether search or social media can



Large errors in flu prediction were largely avoidable, which offers lessons for the use of big data.

run ever since, with a few changes announced in October 2013 (10, 15).

Although not widely reported until 2013, the new GFT has been persistently overestimating flu prevalence for a much longer time. GFT also missed by a very large margin in the 2011–2012 flu season and has missed high for 100 out of 108 weeks starting with August 2011 (see the graph). These errors are not randomly distributed. For example, last week's errors predict this week's errors (temporal autocorrelation), and the direction and magnitude of error varies with the time of year (seasonality). These patterns mean that GFT overlooks considerable information that could be extracted by traditional statistical methods.

<http://www.sciencemag.org/content/343/6176/1203.full.pdf>

Components of a predictor

question -> input data -> features -> algorithm -> parameters -> evaluation

SPAM Example

question -> input data -> features -> algorithm -> parameters -> evaluation

Start with a general question

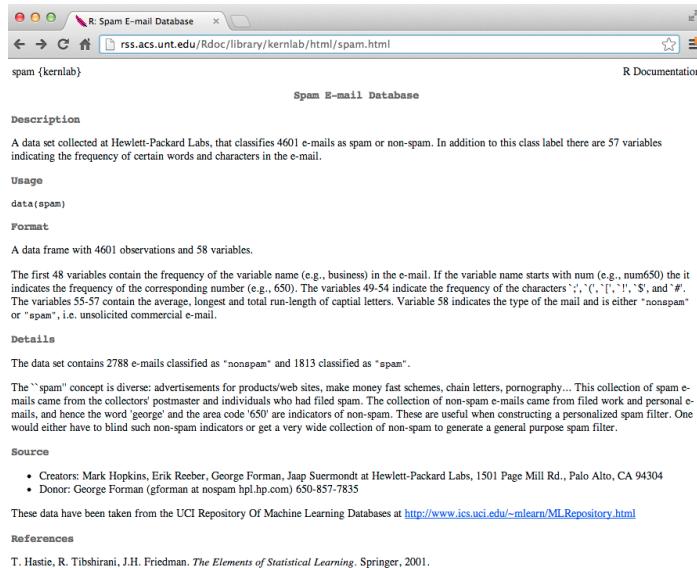
Can I automatically detect emails that are SPAM that are not?

Make it concrete

Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?

SPAM Example

question -> **input data** -> features -> algorithm -> parameters -> evaluation



The screenshot shows a web browser window with the title "R: Spam E-mail Database". The URL in the address bar is "rss.acs.unt.edu/Rdoc/library/kernlab/html/spam.html". The page content is the R documentation for the "spam" dataset, which is a data frame with 4601 observations and 58 variables. It includes sections for Description, Usage, Format, Details, and Source, along with a note about the UCI Repository and references to T. Hastie, R. Tibshirani, J.H. Friedman's book.

Description
A data set collected at Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam. In addition to this class label there are 57 variables indicating the frequency of certain words and characters in the e-mail.

Usage
`data(spam)`

Format
A data frame with 4601 observations and 58 variables.

The first 48 variables contain the frequency of the variable name (e.g., business) in the e-mail. If the variable name starts with num (e.g., num650) the it indicates the frequency of the corresponding number (e.g., 650). The variables 49-54 indicate the frequency of the characters ':', ',', '.', '!', '\$', and '#'. The variables 55-57 contain the average, longest and total run-length of capital letters. Variable 58 indicates the type of the mail and is either "nonspam" or "spam", i.e. unsolicited commercial e-mail.

Details
The data set contains 2788 e-mails classified as "nonspam" and 1813 classified as "spam".

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... This collection of spam e-mails came from the collectors' postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Source

- Creators: Mark Hopkins, Erik Reber, George Forman, Jaap Suermondt at Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304
- Donor: George Forman (gforman@nosspam.hpl.hp.com) 650-857-7835

These data have been taken from the UCI Repository Of Machine Learning Databases at <http://www.ics.uci.edu/~mlearn/MLRepository.html>

References

T. Hastie, R. Tibshirani, J.H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

<http://rss.acs.unt.edu/Rdoc/library/kernlab/html/spam.html>

SPAM Example

question -> input data -> **features** -> algorithm -> parameters -> evaluation

Dear Jeff,

Can you send me your address so I can send you the invitation?

Thanks,

Ben

SPAM Example

question -> input data -> **features** -> algorithm -> parameters -> evaluation

Dear Jeff,

Can **you**

send me your address so I can send **you** the invitation?

Thanks,

Ben

Frequency of **you** = $2/17 = 0.118$

SPAM Example

question -> input data -> **features** -> algorithm -> parameters -> evaluation

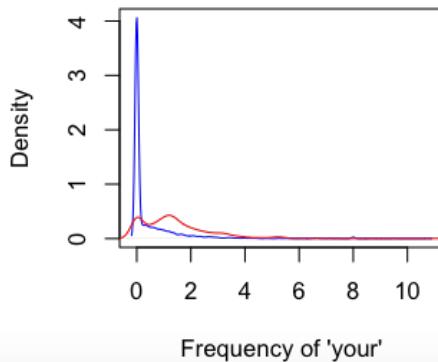
```
library(kernlab)
data(spam)
head(spam)
```

	make	address	all	num3d	our	over	remove	internet	order	mail	receive	will	people	report	addresses	
1	0.00	0.64	0.64	0	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.64	0.00	0.00	0.00	
2	0.21	0.28	0.50	0	0.14	0.28	0.21	0.07	0.00	0.94	0.21	0.79	0.65	0.21	0.14	
3	0.06	0.00	0.71	0	1.23	0.19	0.19	0.12	0.64	0.25	0.38	0.45	0.12	0.00	1.75	
4	0.00	0.00	0.00	0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	
5	0.00	0.00	0.00	0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	
6	0.00	0.00	0.00	0	1.85	0.00	0.00	1.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	free	business	email	you	credit	your	font	num000	money	hp	hpl	george	num650	lab	labs	telnet
1	0.32	0.00	1.29	1.93	0.00	0.96	0	0.00	0.00	0	0	0	0	0	0	0
2	0.14	0.07	0.28	3.47	0.00	1.59	0	0.43	0.43	0	0	0	0	0	0	0
3	0.06	0.06	1.03	1.36	0.32	0.51	0	1.16	0.06	0	0	0	0	0	0	0
4	0.31	0.00	0.00	3.18	0.00	0.31	0	0.00	0.00	0	0	0	0	0	0	0

SPAM Example

question -> input data -> features -> **algorithm** -> parameters -> evaluation

```
plot(density(spam$your[spam$type=="nonspam"]),
      col="blue",main="",xlab="Frequency of 'your'")
lines(density(spam$your[spam$type=="spam"]),col="red")
```



SPAM Example

question -> input data -> features -> **algorithm** -> parameters -> evaluation

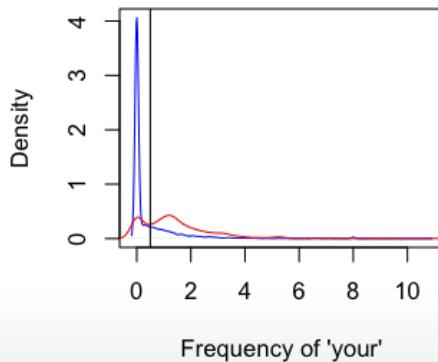
Our algorithm

- Find a value C.
- **frequency of 'your' > C** predict "spam"

SPAM Example

question -> input data -> features -> algorithm -> **parameters** -> evaluation

```
plot(density(spam$your[spam$type=="nonspam"]),
      col="blue",main="",xlab="Frequency of 'your'")
lines(density(spam$your[spam$type=="spam"]),col="red")
abline(v=0.5,col="black")
```



SPAM Example

question -> input data -> features -> algorithm -> parameters -> **evaluation**

```
prediction <- ifelse(spam$your > 0.5, "spam", "nonspam")
table(prediction,spam$type)/length(spam$type)
```

```
prediction nonspam    spam
nonspam   0.4590 0.1017
spam      0.1469 0.2923
```

Accuracy≈ 0.459 + 0.292 = 0.751



Relative importance of steps

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

Relative order of importance

question > data > features > algorithms

An important point

“ The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.”

John Tukey

Garbage in = Garbage out

question -> **input data** -> features -> algorithm -> parameters -> evaluation

1. May be easy (movie ratings -> new movie ratings)
2. May be harder (gene expression data -> disease)
3. Depends on what is a "good prediction".
4. Often more data > better models
5. The most important step!

Features matter!

question -> input data -> **features** -> algorithm -> parameters -> evaluation

Properties of good features

- Lead to data compression
- Retain relevant information
- Are created based on expert application knowledge

Common mistakes

- Trying to automate feature selection
- Not paying attention to data-specific quirks
- Throwing away information unnecessarily

May be automated with care

question -> input data -> **features** -> algorithm -> parameters -> evaluation



<http://arxiv.org/pdf/1112.6209v5.pdf>

Algorithms matter less than you'd think

question -> input data -> features -> **algorithm** -> parameters -> evaluation

TABLE 1

Performance of linear discriminant analysis and the best result we found on ten randomly selected data sets

Data set	Best method e.r.	Lindisc e.r.	Default rule	Prop linear
Segmentation	0.0140	0.083	0.760	0.907
Pima	0.1979	0.221	0.350	0.848
House-votes16	0.0270	0.046	0.386	0.948
Vehicle	0.1450	0.216	0.750	0.883
Satimage	0.0850	0.160	0.758	0.889
Heart Cleveland	0.1410	0.141	0.560	1.000
Splice	0.0330	0.057	0.475	0.945
Waveform21	0.0035	0.004	0.667	0.999
Led7	0.2650	0.265	0.900	1.000
Breast Wisconsin	0.0260	0.038	0.345	0.963

<http://arxiv.org/pdf/math/0606441.pdf>

Issues to consider

The “Best” Machine Learning Method

Interpretable

Simple

Accurate

Fast
(to train and test)

Scalable

<http://strata.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html>

Prediction is about accuracy tradeoffs

- Interpretability versus accuracy
- Speed versus accuracy
- Simplicity versus accuracy
- Scalability versus accuracy

Interpretability matters

```
if total cholesterol ≥160 and smoke then 10 year CHD risk ≥ 5%
else if smoke and systolic blood pressure≥140 then 10 year CHD risk ≥
5%
else 10 year CHD risk < 5%
```

<http://www.cs.cornell.edu/~chenhao/pub/mldg-0815.pdf>

Scalability matters



Innovation

by Mike Masnick

Fri, Apr 13th 2012
12:07am

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:

5

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

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



In sample and out of sample error

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

In sample versus out of sample

In Sample Error: The error rate you get on the same data set you used to build your predictor. Sometimes called resubstitution error.

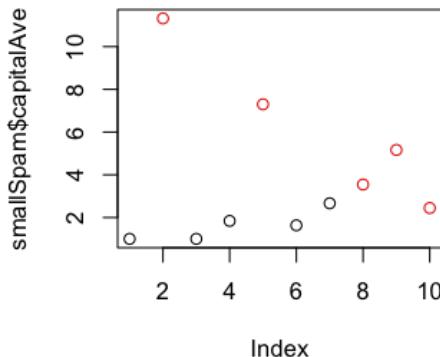
Out of Sample Error: The error rate you get on a new data set. Sometimes called generalization error.

Key ideas

1. Out of sample error is what you care about
2. In sample error < out of sample error
3. The reason is overfitting
 - Matching your algorithm to the data you have

In sample versus out of sample errors

```
library(kernlab); data(spam); set.seed(333)
smallSpam <- spam[sample(dim(spam)[1],size=10),]
spamLabel <- (smallSpam$type=="spam")*1 + 1
plot(smallSpam$capitalAve,col=spamLabel)
```



Prediction rule 1

- capitalAve > 2.7 = "spam"
- capitalAve < 2.40 = "nonspam"
- capitalAve between 2.40 and 2.45 = "spam"
- capitalAve between 2.45 and 2.7 = "nonspam"

Apply Rule 1 to smallSpam

```
rule1 <- function(x){  
  prediction <- rep(NA,length(x))  
  prediction[x > 2.7] <- "spam"  
  prediction[x < 2.40] <- "nonspam"  
  prediction[(x >= 2.40 & x <= 2.45)] <- "spam"  
  prediction[(x > 2.45 & x <= 2.70)] <- "nonspam"  
  return(prediction)  
}  
  
table(rule1(smallSpam$capitalAve),smallSpam$type)
```

	nonspam	spam
nonspam	5	0
spam	0	5

Prediction rule 2

- $\text{capitalAve} > 2.40 = \text{"spam"}$
- $\text{capitalAve} \leq 2.40 = \text{"nonspam"}$

Apply Rule 2 to smallSpam

```
rule2 <- function(x){  
  prediction <- rep(NA,length(x))  
  prediction[x > 2.8] <- "spam"  
  prediction[x <= 2.8] <- "nonspam"  
  return(prediction)  
}  
table(rule2(smallSpam$capitalAve),smallSpam$type)
```

	nonspam	spam
nonspam	5	1
spam	0	4

Apply to complete spam data

```
table(rule1(spam$capitalAve),spam$type)
```

	nonspam	spam
nonspam	2141	588
spam	647	1225

```
table(rule2(spam$capitalAve),spam$type)
```

	nonspam	spam
nonspam	2224	642
spam	564	1171

```
mean(rule1(spam$capitalAve)==spam$type)
```

Look at accuracy

```
sum(rule1(spam$capitalAve)==spam$type)
```

```
[1] 3366
```

```
sum(rule2(spam$capitalAve)==spam$type)
```

```
[1] 3395
```

What's going on?

Overfitting

- Data have two parts
 - Signal
 - Noise
- The goal of a predictor is to find signal
- You can always design a perfect in-sample predictor
- You capture both signal + noise when you do that
- Predictor won't perform as well on new samples

<http://en.wikipedia.org/wiki/Overfitting>



Prediction study design

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Prediction study design

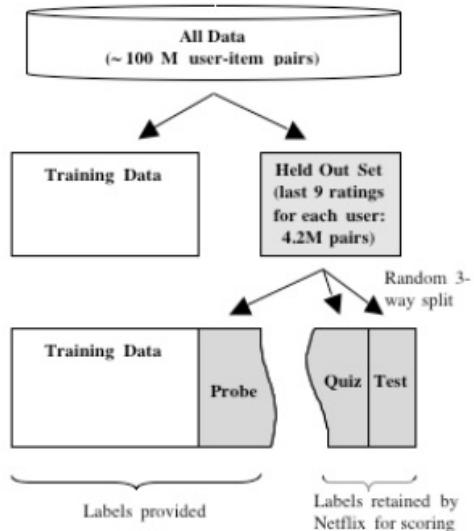
1. Define your error rate
2. Split data into:
 - Training, Testing, Validation (optional)
3. On the training set pick features
 - Use cross-validation
4. On the training set pick prediction function
 - Use cross-validation
5. If no validation
 - Apply 1x to test set
6. If validation
 - Apply to test set and refine
 - Apply 1x to validation

Know the benchmarks

Leaderboard – Heritage Health Prize					
#	User	Score	Rank	Date	Comments
1272	¶17 AceHack	0.521236	1	Wed, 25 May 2011 00:37:12	
1273	¶17 Shirohishi	0.521265	4	Wed, 25 Apr 2012 00:03:34	
1274	¶17 David Fu	0.521414	3	Mon, 23 Apr 2012 18:06:10	
1275	¶17 wiem	0.521603	3	Sat, 25 Aug 2012 10:20:53	
1276	¶17 Frostmourne	0.521865	4	Thu, 06 Sep 2012 11:50:22 (-23.9h)	
1277	¶17 John.Umbaugh	0.521902	3	Fri, 15 Jul 2011 04:24:03 (-5.2d)	
1278	¶17 Dow's team	0.521911	7	Thu, 16 Jun 2011 14:58:22 (-6.4d)	
All Zeros Benchmark					
1279	¶17 hyperdose	0.522226	11	Thu, 23 Jun 2011 21:23:27 (-37d)	
1279	¶17 matchstick314	0.522226	3	Sun, 06 Jun 2011 01:34:48 (-16.1d)	
1279	¶17 David Howden	0.522226	1	Mon, 23 May 2011 10:56:41	
1279	¶17 heritage	0.522226	1	Sat, 26 May 2011 20:54:17	
1279	¶17 Igor Kamenev	0.522226	5	Sat, 25 Jun 2011 13:03:11 (-24.2d)	
1279	¶17 iversonkxmd	0.522226	1	Wed, 01 Jun 2011 10:55:19	
1279	¶17 GEMc	0.522226	1	Sat, 11 Jun 2011 03:06:33	
1279	¶17 ay998470	0.522226	2	Mon, 13 Jun 2011 06:27:09	

<http://www.heritagehealthprize.com/c/hhp/leaderboard>

Study design



<http://www2.research.att.com/~volinsky/papers/ASAStatComp.pdf>

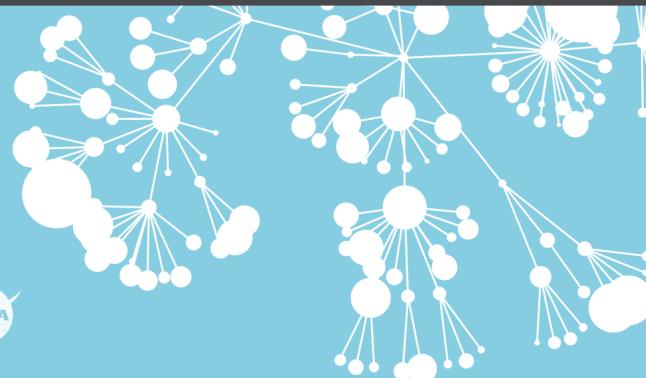
Used by the professionals

kaggle

Customer Solutions Competitions Community ▾

Sign Up Login

We're the global leader in solving business challenges through predictive analytics.



facebook. GE MasterCard. MERCK NASA

Compete as a data scientist for fortune, fame and fun »

<http://www.kaggle.com/>

Avoid small sample sizes

- Suppose you are predicting a binary outcome
 - Diseased/healthy
 - Click on ad/not click on ad
- One classifier is flipping a coin
- Probability of perfect classification is approximately:
 - $\left(\frac{1}{2}\right)^{\text{sample size}}$
 - $n = 1$ flipping coin 50% chance of 100% accuracy
 - $n = 2$ flipping coin 25% chance of 100% accuracy
 - $n = 10$ flipping coin 0.10% chance of 100% accuracy

Rules of thumb for prediction study design

- If you have a large sample size
 - 60% training
 - 20% test
 - 20% validation
- If you have a medium sample size
 - 60% training
 - 40% testing
- If you have a small sample size
 - Do cross validation
 - Report caveat of small sample size

Some principles to remember

- Set the test/validation set aside and *don't look at it*
- In general *randomly* sample training and test
- Your data sets must reflect structure of the problem
 - If predictions evolve with time split train/test in time chunks (called [backtesting](#) in finance)
- All subsets should reflect as much diversity as possible
 - Random assignment does this
 - You can also try to balance by features - but this is tricky



Types of errors

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Basic terms

In general, **Positive** = identified and **negative** = rejected. Therefore:

True positive = correctly identified

False positive = incorrectly identified

True negative = correctly rejected

False negative = incorrectly rejected

Medical testing example:

True positive = Sick people correctly diagnosed as sick

False positive = Healthy people incorrectly identified as sick

True negative = Healthy people correctly identified as healthy

False negative = Sick people incorrectly identified as healthy.

Key quantities

		DISEASE	
		+	-
TEST	+	TP	FP
	-	FN	TN

Sensitivity

→ $\Pr(\text{positive test} \mid \text{disease})$

Specificity

→ $\Pr(\text{negative test} \mid \text{no disease})$

Positive Predictive Value

→ $\Pr(\text{disease} \mid \text{positive test})$

Negative Predictive Value

→ $\Pr(\text{no disease} \mid \text{negative test})$

Accuracy

→ $\Pr(\text{correct outcome})$

http://en.wikipedia.org/wiki/Sensitivity_and_specificity

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>

Key quantities as fractions

		DISEASE	
		+	-
TEST	+	TP	FP
	-	FN	TN

Sensitivity

→ $TP / (TP+FN)$

Specificity

→ $TN / (FP+TN)$

Positive Predictive Value

→ $TP / (TP+FP)$

Negative Predictive Value

→ $TN / (FN+TN)$

Accuracy

→ $(TP+TN) / (TP+FP+FN+TN)$

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>

Screening tests

Assume that some disease has a 0.1% prevalence in the population. Assume we have a test kit for that disease that works with 99% sensitivity and 99% specificity. What is the probability of a person having the disease **given the test result is positive**, if we randomly select a subject from

- ▶ the general population?
- ▶ a high risk sub-population with 10% disease prevalence?

General population

		DISEASE	
		+	-
TEST	+	99	999
	-	1	98901

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>

General population as fractions

		DISEASE	
		+	-
		+	99
TEST	+	99	999
	-	1	98901

Sensitivity

$$\rightarrow 99 / (99+1) = 99\%$$

Specificity

$$\rightarrow 98901 / (999+98901) = 99\%$$

Positive Predictive Value

$$\rightarrow 99 / (99+999) \approx 9\%$$

Negative Predictive Value

$$\rightarrow 98901 / (1+98901) > 99.9\%$$

Accuracy

$$\rightarrow (99+98901) / 100000 = 99\%$$

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>

At risk subpopulation

		DISEASE	
		+	-
TEST	+	9900	900
	-	100	89100

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>

At risk subpopulation as fraction

		DISEASE	
		+	-
TEST	+	9900	900
	-	100	89100

Sensitivity $\rightarrow 9900 / (9900+100) = 99\%$

Specificity $\rightarrow 89100 / (900+89100) = 99\%$

Positive Predictive Value $\rightarrow 9900 / (9900+900) \approx 92\%$

Negative Predictive Value $\rightarrow 89100 / (100+89100) \approx 99.9\%$

Accuracy $\rightarrow (9900+89100) / 100000 = 99\%$

Key public health issue

Vast Study Casts Doubts on Value of Mammograms

By GINA KOLATA FEB. 11, 2014



One of the largest and most meticulous studies of mammography ever done, involving 90,000 women and lasting a quarter-century, has added powerful new doubts about the value of the screening test for women of any age.



It found that the death rates from breast cancer and from all causes were the same in women who got mammograms and those who did not. And the screening had harms: One in five cancers found with mammography and treated was not a threat to the woman's health and did not need treatment such as chemotherapy, surgery or radiation.

[The study](#), published Tuesday in The British Medical Journal, is one of the few rigorous evaluations of mammograms in decades.

<http://www.biostat.jhsph.edu/~iruczins/teaching/140.615/>



Nearly 75 percent of American women 40 and over say they had a mammogram in the past year. Damian Dovarganes/Associated Press

Key public health issue

The New York Times

Business Day

WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION ARTS

Search Global DealBook Markets Economy Energy Media Technology

Looser Guidelines Issued on Prostate Screening

By ANDREW POLLACK

Published: May 3, 2013

In a major shift, the American Urological Association has pulled back its strong support of [prostate cancer](#) screening, saying that the testing should be considered primarily by men aged 55 to 69.

The association had staunchly defended the benefits of screening men with the prostate test, even after a government advisory committee, the United States Preventive Services Task Force, said in 2011 that healthy men should not be screened because far more men would be harmed by unnecessary prostate cancer treatments than would be saved from death.

 FACEBOOK

 TWITTER

 GOOGLE+

 SAVE

 EMAIL

 SHARE

 PRINT

 REPRINTS

For continuous data

Mean squared error (MSE):

$$\frac{1}{n} \sum_{i=1}^n (\text{Prediction}_i - \text{Truth}_i)^2$$

Root mean squared error (RMSE):


$$\sqrt{\frac{1}{n} \sum_{i=1}^n (\text{Prediction}_i - \text{Truth}_i)^2}$$

Common error measures

1. Mean squared error (or root mean squared error)

- Continuous data, sensitive to outliers

2. Median absolute deviation

- Continuous data, often more robust

3. Sensitivity (recall)

- If you want few missed positives

4. Specificity

- If you want few negatives called positives

5. Accuracy

- Weights false positives/negatives equally

6. Concordance

- One example is [kappa](#)



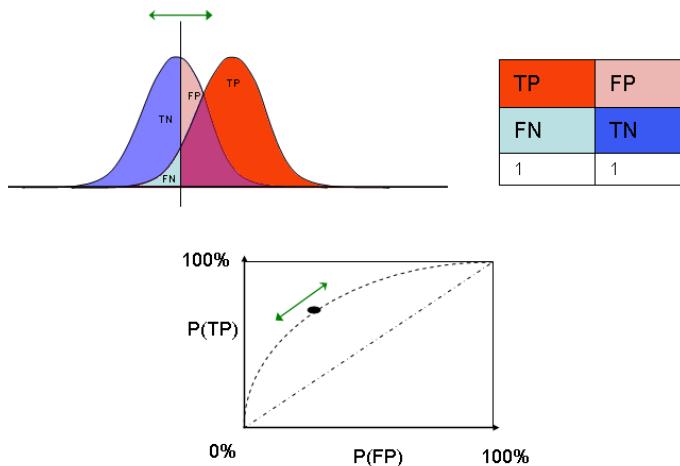
ROC curves

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Why a curve?

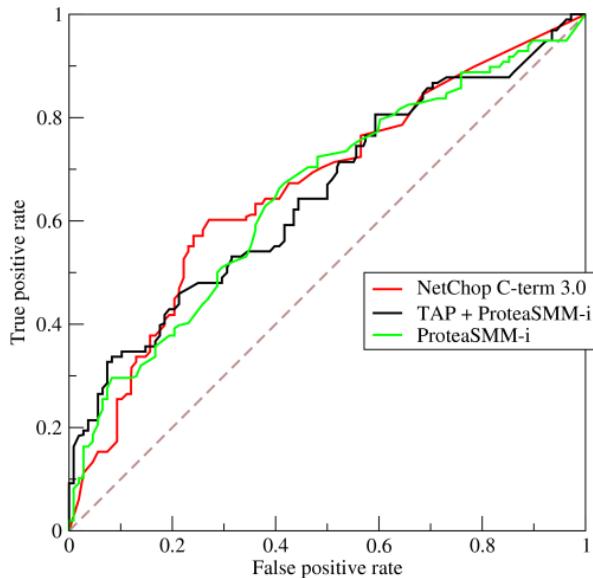
- In binary classification you are predicting one of two categories
 - Alive/dead
 - Click on ad/don't click
- But your predictions are often quantitative
 - Probability of being alive
 - Prediction on a scale from 1 to 10
- The *cutoff* you choose gives different results

ROC curves



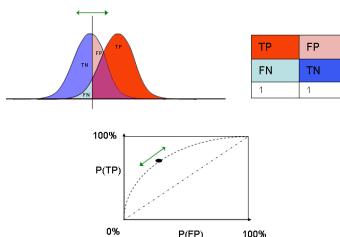
http://en.wikipedia.org/wiki/Receiver_operating_characteristic

An example



http://en.wikipedia.org/wiki/Receiver_operating_characteristic

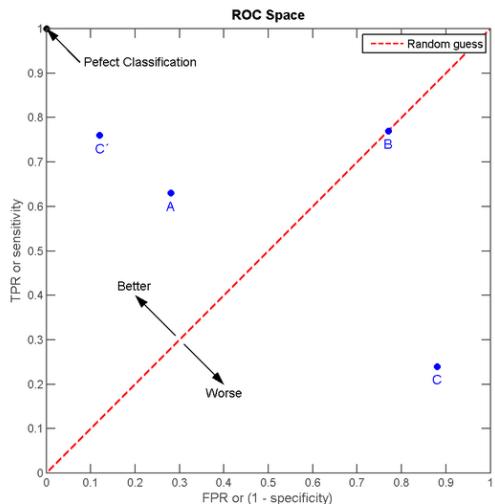
Area under the curve



- AUC = 0.5: random guessing
- AUC = 1: perfect classifier
- In general AUC of above 0.8 considered "good"

http://en.wikipedia.org/wiki/Receiver_operating_characteristic

What is good?



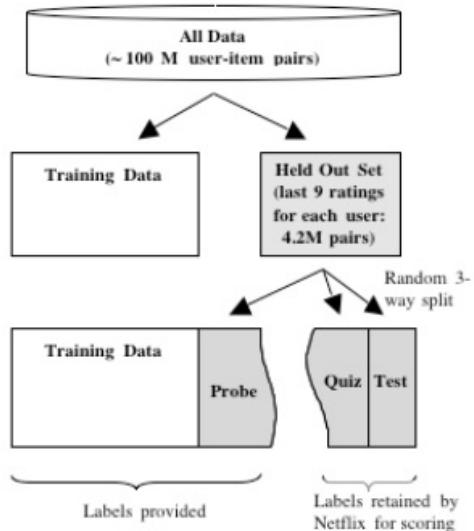
http://en.wikipedia.org/wiki/Receiver_operating_characteristic



Cross validation

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Study design



<http://www2.research.att.com/~volinsky/papers/ASAStatComp.pdf>

Key idea

1. Accuracy on the training set (resubstitution accuracy) is optimistic
2. A better estimate comes from an independent set (test set accuracy)
3. But we can't use the test set when building the model or it becomes part of the training set
4. So we estimate the test set accuracy with the training set.

Cross-validation

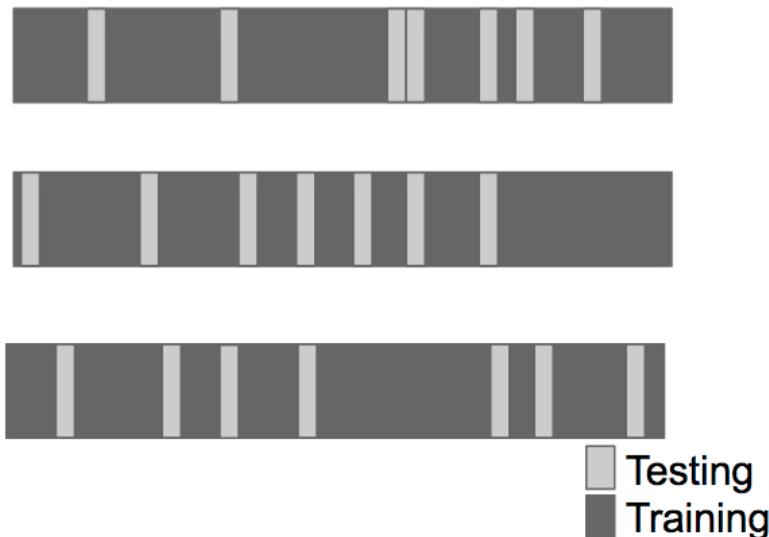
Approach:

1. Use the training set
2. Split it into training/test sets
3. Build a model on the training set
4. Evaluate on the test set
5. Repeat and average the estimated errors

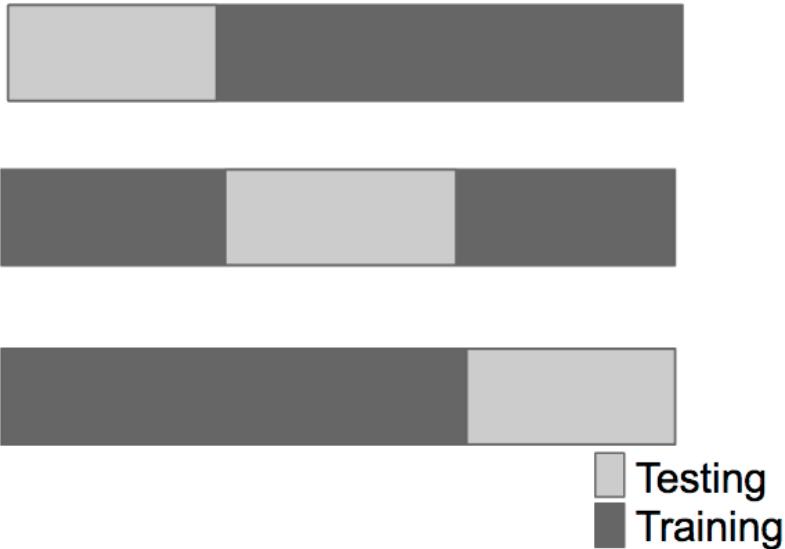
Used for:

1. Picking variables to include in a model
2. Picking the type of prediction function to use
3. Picking the parameters in the prediction function
4. Comparing different predictors

Random subsampling



K-fold



Leave one out



Considerations

- For time series data data must be used in "chunks"
- For k-fold cross validation
 - Larger k = less bias, more variance
 - Smaller k = more bias, less variance
- Random sampling must be done *without replacement*
- Random sampling with replacement is the *bootstrap*
 - Underestimates of the error
 - Can be corrected, but it is complicated ([0.632 Bootstrap](#))
- If you cross-validate to pick predictors estimate you must estimate errors on independent data.



What data should you use?

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

A successful predictor



fivethirtyeight.com

Polling data

Home GALLUP® Search Gallup.com

HOME POLITICS ECONOMY WELL-BEING WORLD GALLUP ANALYTICS

HOT TOPICS: Healthcare Law Guns U.S. Government Shutdown Iran Syria Russia U.S. Leadership Approval Race Relations T.

One in Four U.S. Uninsured Plan to Remain That Way



December 3, 2013

Twenty-eight percent of uninsured Americans say they are more likely to pay the fine for not having health insurance than to obtain insurance, as required by the healthcare law. Politics appear to be a major factor in that decision.

U.S. Economic Confidence Rises in November

December 3, 2013

U.S. Economic Confidence Index, Monthly Averages



Date	Index Value
Jan '12	-22
May '12	-26
Sep '12	-27
Jan '13	-16
May '13	-7
Sep '13	-13
Dec '13	-19
Jan '14	-25
May '14	-35

Inside Strategic Consulting

<http://www.gallup.com/>

Weighting the data

FiveThirtyEight

6.06.2010

Pollster Ratings v4.0: Methodology

by [Nate Silver](#)

Rating pollsters is at the core of FiveThirtyEight's mission, and forms the backbone of our forecasting models. But, it has been two years since we [last revised our ratings](#). Here, at last, is an update. We have both substantially increased the amount of data that we are evaluating, and significantly refined our methodology.

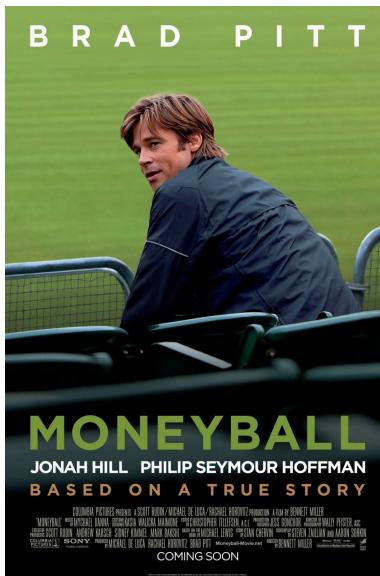
<http://www.fivethirtyeight.com/2010/06/pollster-ratings-v40-methodology.html>

Key idea

To predict X use data related to X

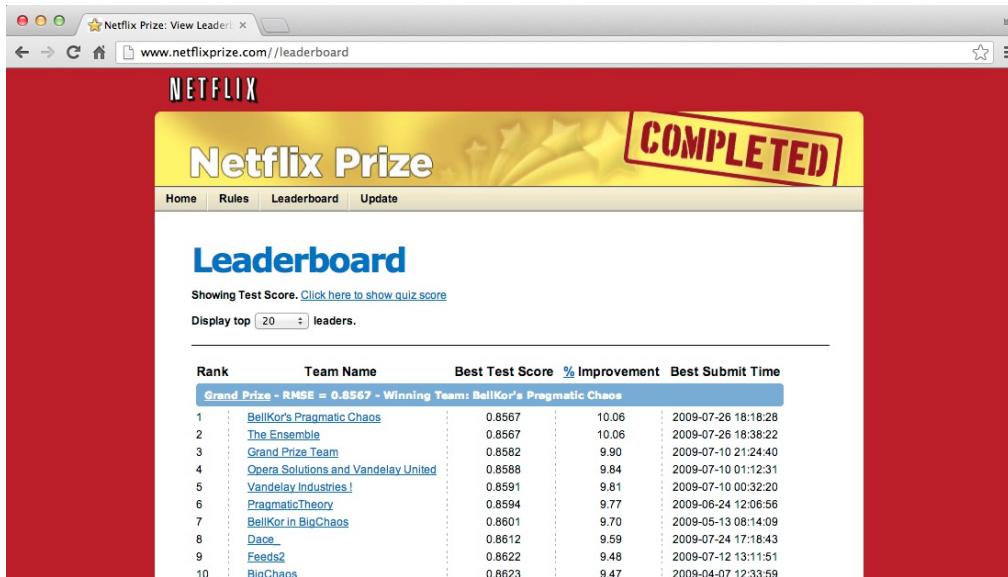
Key idea

To predict player performance use data about player performance



Key idea

To predict movie preferences use data about movie preferences



The screenshot shows a web browser window for the Netflix Prize Leaderboard at www.netflixprize.com//leaderboard. The page has a red header with the Netflix logo and a large yellow banner that says "COMPLETED". Below the banner, there's a navigation bar with links for Home, Rules, Leaderboard, and Update. The main section is titled "Leaderboard" and displays a table of top teams. The table has columns for Rank, Team Name, Best Test Score, % Improvement, and Best Submit Time. A blue header row indicates the winning team is BellKor's Pragmatic Chaos with an RMSE of 0.8567. The table lists 10 teams, starting with BellKor's Pragmatic Chaos at rank 1.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
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	0.8562	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8568	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

Key idea

To predict hospitalizations use data about hospitalizations

Information Data Forum Leaderboard



**Improve Healthcare,
Win \$3,000,000.**

COMPETITION GOAL

Identify patients who will be admitted to a hospital within the next year, using historical claims data.

Not a hard rule

To predict flu outbreaks use Google searches



<http://www.google.org/flutrends/>

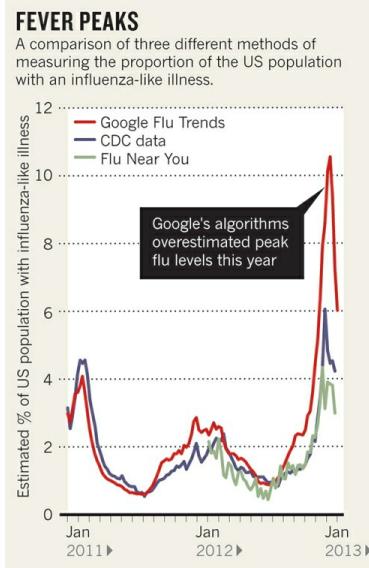
Looser connection = harder prediction

Oncotype DX® reveals
the underlying biology that
changes treatment decisions
37% of the time

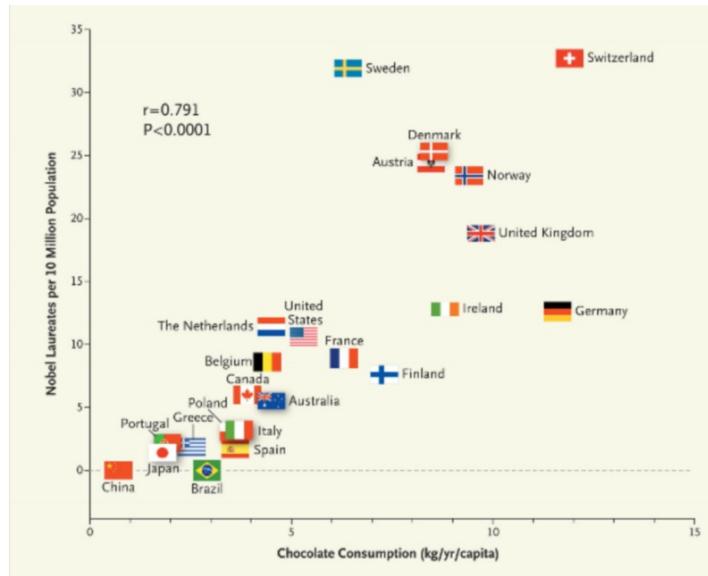
Uncover the Unexpected™



Data properties matter



Unrelated data is the most common mistake



<http://www.nejm.org/doi/full/10.1056/NEJMoa1211064>



The caret package

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

The caret R package

the caret package



The **caret** package (short for Classification And REgression Training) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

<http://caret.r-forge.r-project.org/>

Links

[train Model List](#)

Topics

[Main Page](#)

[Data Sets](#)

[Visualizations](#)

[Pre-Processing](#)

Caret functionality

- Some preprocessing (cleaning)
 - `preProcess`

- Data splitting
 - `createDataPartition`
 - `createResample`
 - `createTimeSlices`

- Training/testing functions
 - `train`
 - `predict`

- Model comparison
 - `confusionMatrix`

Machine learning algorithms in R

- Linear discriminant analysis
- Regression
- Naive Bayes
- Support vector machines
- Classification and regression trees
- Random forests
- Boosting
- etc.

Why caret?

obj	Class	Package	predict Function Syntax
lda	MASS		<code>predict(obj)</code> (no options needed)
glm	stats		<code>predict(obj, type = "response")</code>
gbm	gbm		<code>predict(obj, type = "response", n.trees)</code>
mda	mda		<code>predict(obj, type = "posterior")</code>
rpart	rpart		<code>predict(obj, type = "prob")</code>
Weka	RWeka		<code>predict(obj, type = "probability")</code>
LogitBoost	caTools		<code>predict(obj, type = "raw", nIter)</code>

http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user_caret_2up.pdf

SPAM Example: Data splitting

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain, ]
testing <- spam[-inTrain, ]
dim(training)
```

```
[1] 3451    58
```

SPAM Example: Fit a model

```
set.seed(32343)
modelFit <- train(type ~., data=training, method="glm")
modelFit
```

Generalized Linear Model

3451 samples
57 predictors
2 classes: 'nonspam', 'spam'

No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.9	0.8	0.02	0.04

SPAM Example: Final model

```
modelFit <- train(type ~., data=training, method="glm")
modelFit$finalModel
```

Call: NULL

Coefficients:

(Intercept)	make	address	all	num3d
-1.78e+00	-7.76e-01	-1.39e-01	3.68e-02	1.94e+00
our	over	remove	internet	order
7.61e-01	6.66e-01	2.34e+00	5.94e-01	4.10e-01
mail	receive	will	people	report
4.08e-02	2.71e-01	-1.08e-01	-2.28e-01	-1.14e-01
addresses	free	business	email	you
2.16e+00	8.78e-01	6.49e-01	1.38e-01	6.91e-02
credit	your	font	num000	money
8.00e-01	2.17e-01	2.17e-01	2.04e+00	1.95e+00
hp	hpl	george	num650	lab
-1.82e+00	-9.17e-01	-7.50e+00	3.33e-01	-1.89e+00
labs	telnet	num857	data	num415

SPAM Example: Prediction

```
predictions <- predict(modelFit,newdata=testing)
predictions
```

[1]	spam	spam	spam	nonspam	nonspam	nonspam	spam	spam	spam	spam	spam
[12]	spam	nonspam	spam	spam	spam						
[23]	nonspam	spam	nonspam	nonspam	spam	spam	spam	spam	spam	spam	spam
[34]	spam	spam	spam								
[45]	spam	spam	spam	spam	nonspam	spam	nonspam	spam	spam	spam	spam
[56]	spam	nonspam	nonspam	spam	spam	spam	spam	spam	nonspam	spam	spam
[67]	spam	spam	spam								
[78]	nonspam	nonspam	nonspam	spam	spam	nonspam	spam	nonspam	nonspam	spam	spam
[89]	spam	spam	spam	spam	spam	spam	nonspam	spam	spam	spam	spam
[100]	spam	spam	spam	nonspam	spam	nonspam	spam	spam	spam	spam	spam
[111]	spam	spam	spam	spam	nonspam	spam	spam	spam	spam	spam	spam
[122]	spam	nonspam	spam	spam	nonspam						
[133]	spam	spam	spam								
[144]	spam	spam	spam	nonspam	spam	spam	spam	spam	spam	spam	spam
[155]	nonspam	spam	nonspam	spam	nonspam	spam	spam	spam	spam	spam	spam
[166]	spam	spam	spam								
[177]	spam	spam	spam								

SPAM Example: Confusion Matrix

```
confusionMatrix(predictions,testing$type)
```

Confusion Matrix and Statistics

		Reference	
		nonspam	spam
Prediction	nonspam	665	54
	spam	32	399

Accuracy : 0.925
95% CI : (0.908, 0.94)

No Information Rate : 0.606

P-Value [Acc > NIR] : <2e-16

Kappa : 0.842
McNemar's Test P-Value : 0.0235

Sensitivity : 0.954
Specificity : 0.881
Pos Pred Value : 0.925

Further information

- Caret tutorials:
 - http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user_caret_2up.pdf
 - <http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>
- A paper introducing the caret package
 - <http://www.jstatsoft.org/v28/i05/paper>



Data slicing

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

SPAM Example: Data splitting

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain, ]
testing <- spam[-inTrain, ]
dim(training)
```

```
[1] 3451    58
```

SPAM Example: K-fold

```
set.seed(32323)
folds <- createFolds(y=spam$type, k=10,
                      list=TRUE, returnTrain=TRUE)
sapply(folds, length)
```

```
Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
4141    4140    4141    4142    4140    4142    4141    4141    4140    4141
```

```
folds[[1]][1:10]
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

SPAM Example: Return test

```
set.seed(32323)
folds <- createFolds(y=spam$type,k=10,
                      list=TRUE,returnTrain=FALSE)
sapply(folds,length)
```

```
Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
 460     461     460     459     461     459     460     460     461     460
```

```
folds[[1]][1:10]
```

```
[1] 24 27 32 40 41 43 55 58 63 68
```

SPAM Example: Resampling

```
set.seed(32323)
folds <- createResample(y=spam$type,times=10,
                        list=TRUE)
sapply(folds,length)
```

```
Resample01 Resample02 Resample03 Resample04 Resample05 Resample06 Resample07 Resample08 Resample09
    4601      4601      4601      4601      4601      4601      4601      4601      4601
Resample10
    4601
```

```
folds[[1]][1:10]
```

```
[1] 1 2 3 3 5 5 7 8 12
```

SPAM Example: Time Slices

```
set.seed(32323)
tme <- 1:1000
folds <- createTimeSlices(y=tme,initialWindow=20,
                           horizon=10)
names(folds)
```

```
[1] "train" "test"
```

```
folds$train[ [1]]
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```
folds$test[ [1]]
```

```
[1] 21 22 23 24 25 26 27 28 29 30
```

Further information

- Caret tutorials:
 - http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user_caret_2up.pdf
 - <http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf>
- A paper introducing the caret package
 - <http://www.jstatsoft.org/v28/i05/paper>



Training options

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

SPAM Example

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain, ]
testing <- spam[-inTrain, ]
modelFit <- train(type ~., data=training, method="glm")
```

Train options

```
args(train.default)
```

```
function (x, y, method = "rf", preProcess = NULL, ..., weights = NULL,
metric = ifelse(is.factor(y), "Accuracy", "RMSE"), maximize = ifelse(metric ==
"RMSE", FALSE, TRUE), trControl = trainControl(), tuneGrid = NULL,
tuneLength = 3)
NULL
```

Metric options

Continuous outcomes:

- $RMSE$ = Root mean squared error
- $RSquared$ = R^2 from regression models

Categorical outcomes:

- $Accuracy$ = Fraction correct
- $Kappa$ = A measure of [concordance](#)

trainControl

```
args(trainControl)
```

```
function (method = "boot", number = ifelse(method %in% c("cv",
"repeatedcv"), 10, 25), repeats = ifelse(method %in% c("cv",
"repeatedcv"), 1, number), p = 0.75, initialWindow = NULL,
horizon = 1, fixedWindow = TRUE, verboseIter = FALSE, returnData = TRUE,
returnResamp = "final", savePredictions = FALSE, classProbs = FALSE,
summaryFunction = defaultSummary, selectionFunction = "best",
custom = NULL, preProcOptions = list(thresh = 0.95, ICAcomp = 3,
k = 5), index = NULL, indexOut = NULL, timingSamps = 0,
predictionBounds = rep(FALSE, 2), seeds = NA, allowParallel = TRUE)
NULL
```

trainControl resampling

- *method*
 - *boot* = bootstrapping
 - *boot632* = bootstrapping with adjustment
 - *cv* = cross validation
 - *repeatedcv* = repeated cross validation
 - *LOOCV* = leave one out cross validation
- *number*
 - For boot/cross validation
 - Number of subsamples to take
- *repeats*
 - Number of times to repeat subsampling
 - If big this can *slow things down*

Setting the seed

- It is often useful to set an overall seed
- You can also set a seed for each resample
- Seeding each resample is useful for parallel fits

seed example

```
set.seed(1235)
modelFit2 <- train(type ~., data=training, method="glm")
modelFit2
```

Generalized Linear Model

3451 samples
57 predictors
2 classes: 'nonspam', 'spam'

No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.9	0.8	0.007	0.01

seed example

```
set.seed(1235)
modelFit3 <- train(type ~., data=training, method="glm")
modelFit3
```

Generalized Linear Model

3451 samples
57 predictors
2 classes: 'nonspam', 'spam'

No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.9	0.8	0.007	0.01

Further resources

- [Caret tutorial](#)
- [Model training and tuning](#)



Plotting predictors

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Example: predicting wages



Image Credit <http://www.cahs-media.org/the-high-cost-of-low-wages>

Data from: [ISLR package](#) from the book: [Introduction to statistical learning](#)

Example: Wage data

```
library(ISLR); library(ggplot2); library(caret);  
data(Wage)  
summary(Wage)
```

year	age	sex	maritl	race
Min. :2003	Min. :18.0	1. Male :3000	1. Never Married: 648	1. White:2480
1st Qu.:2004	1st Qu.:33.8	2. Female: 0	2. Married :2074	2. Black: 293
Median :2006	Median :42.0		3. Widowed : 19	3. Asian: 190
Mean :2006	Mean :42.4		4. Divorced : 204	4. Other: 37
3rd Qu.:2008	3rd Qu.:51.0		5. Separated : 55	
Max. :2009	Max. :80.0			
education		region	jobclass	health
1. < HS Grad	:268	2. Middle Atlantic :3000	1. Industrial :1544	1. <=Good : 858
2. HS Grad	:971	1. New England : 0	2. Information:1456	2. >=Very Good:2142
3. Some College	:650	3. East North Central: 0		
4. College Grad	:685	4. West North Central: 0		
5. Advanced Degree	:426	5. South Atlantic : 0		
		6. East South Central: 0		
		(Other) : 0		

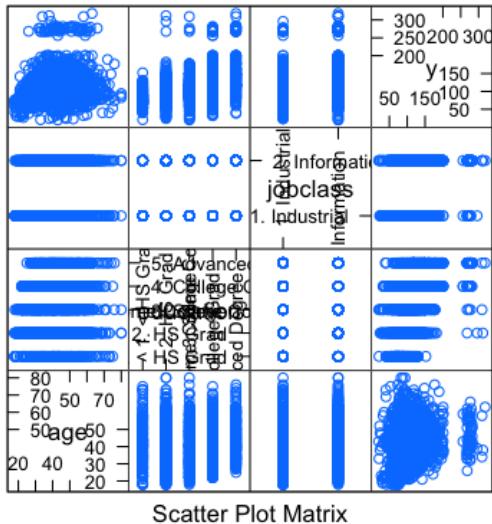
Get training/test sets

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

```
[1] 898 12
```

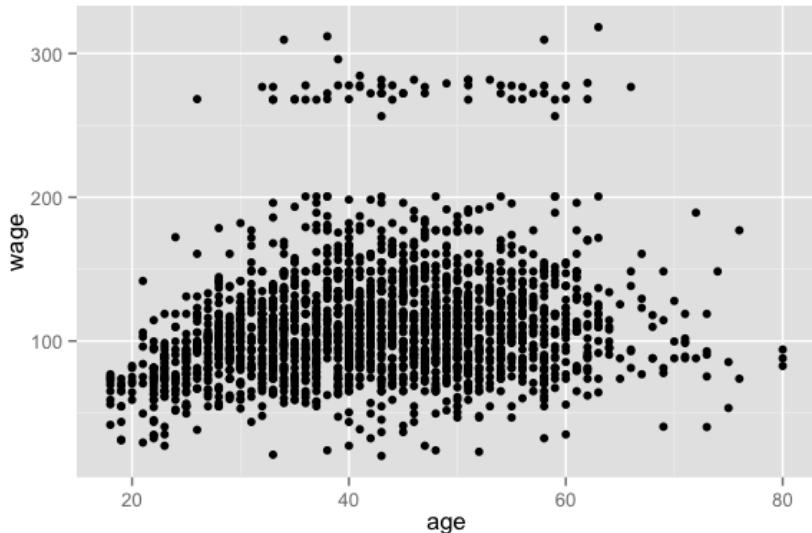
Feature plot (*caret* package)

```
featurePlot(x=training[,c("age","education","jobclass")],  
            y = training$wage,  
            plot="pairs")
```



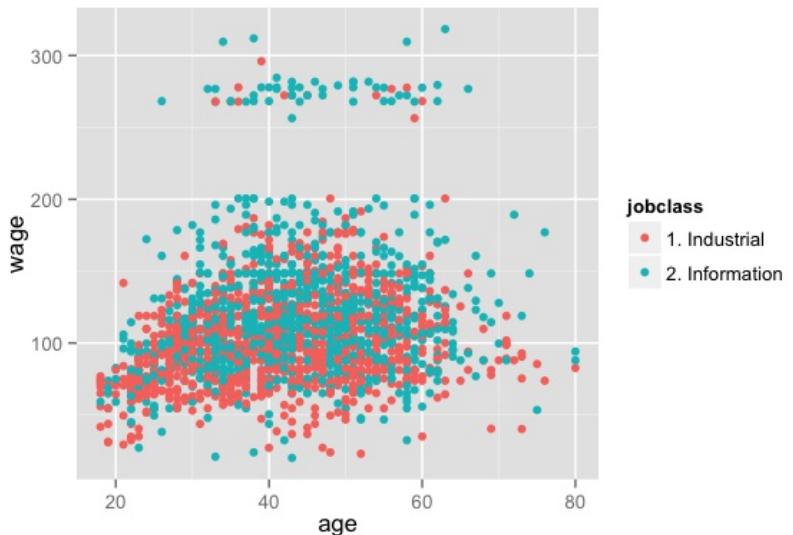
Qplot (*ggplot2* package)

```
qplot(age,wage,data=training)
```



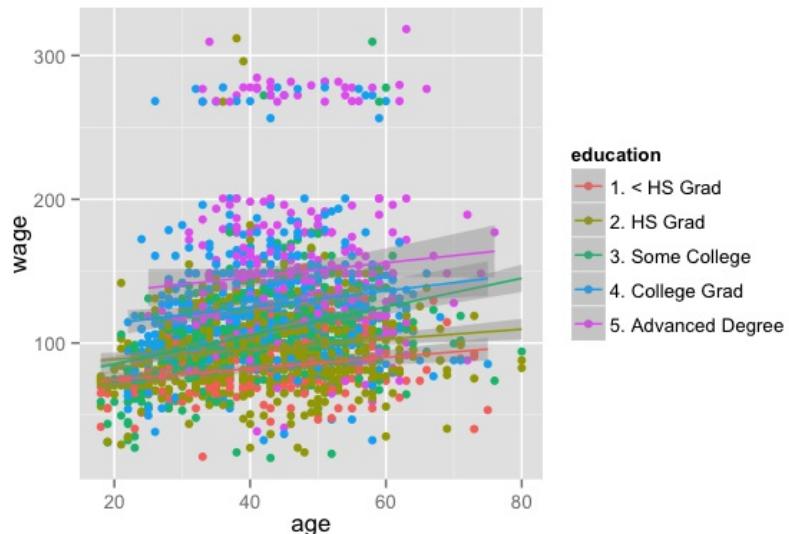
Qplot with color (*ggplot2* package)

```
qplot(age,wage,colour=jobclass,data=training)
```



Add regression smoothers (*ggplot2* package)

```
qq <- qplot(age,wage,colour=education,data=training)  
qq + geom_smooth(method='lm',formula=y~x)
```



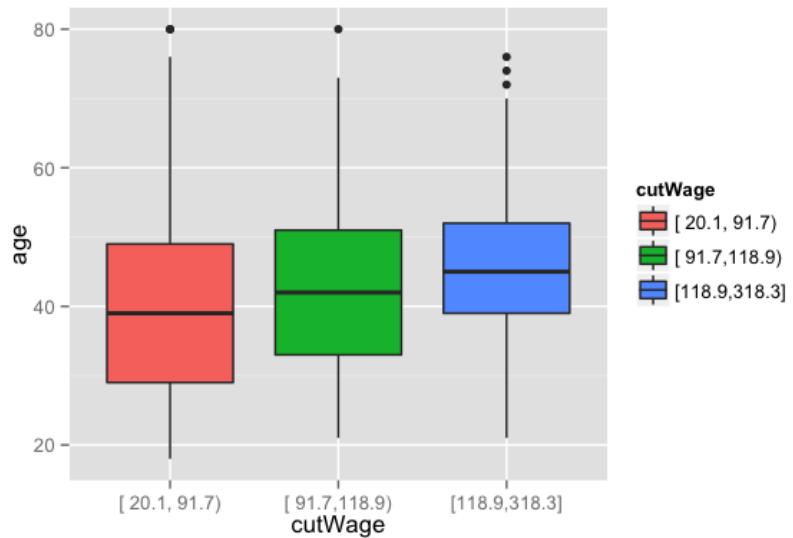
cut2, making factors (*Hmisc* package)

```
cutWage <- cut2(training$wage,g=3)  
table(cutWage)
```

```
cutWage  
[ 20.1, 91.7) [ 91.7,118.9) [118.9,318.3]  
    704          725          673
```

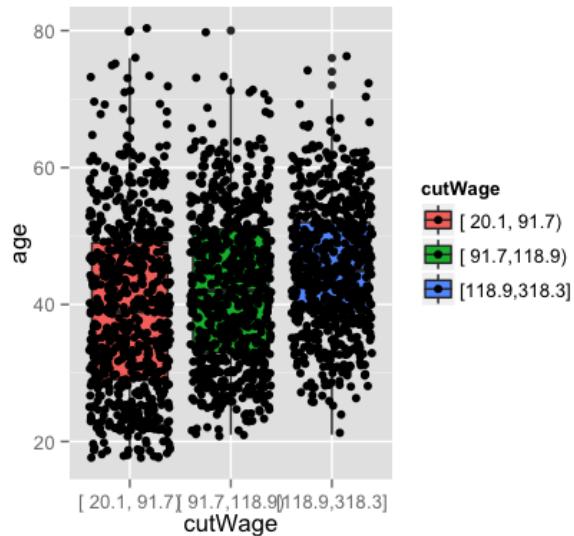
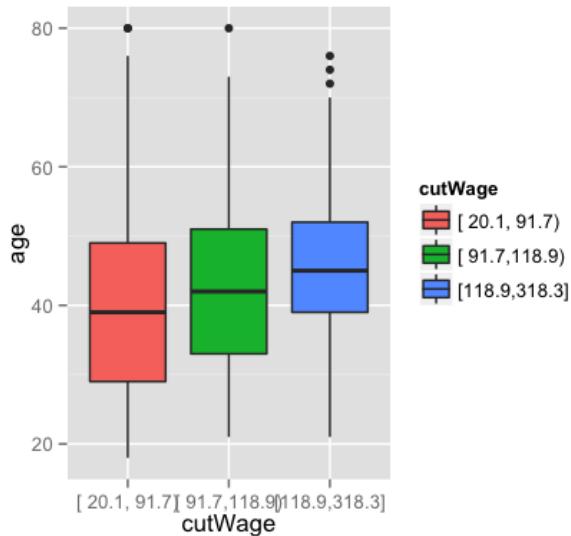
Boxplots with cut2

```
p1 <- qplot(cutWage, age, data=training, fill=cutWage,  
            geom=c("boxplot"))  
p1
```



Boxplots with points overlaid

```
p2 <- qplot(cutWage, age, data=training, fill=cutWage,  
            geom=c("boxplot", "jitter"))  
grid.arrange(p1,p2,ncol=2)
```



Tables

```
t1 <- table(cutWage,training$jobclass)  
t1
```

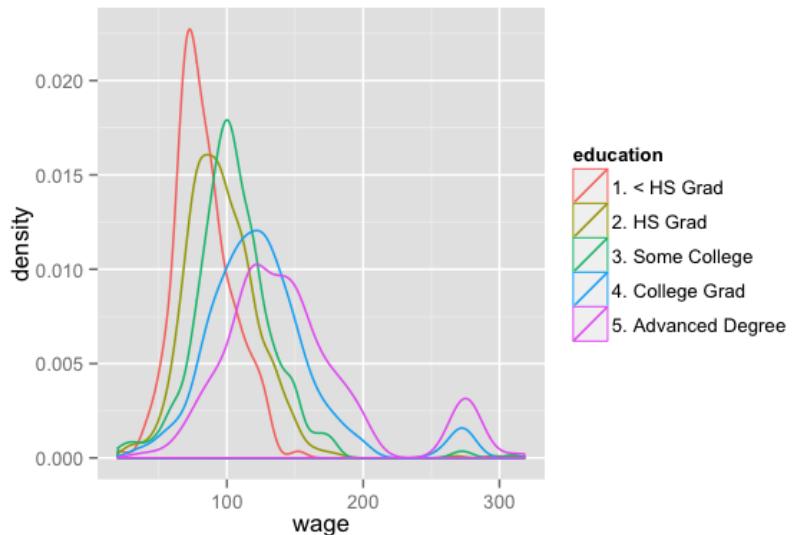
```
cutWage      1. Industrial 2. Information  
[ 20.1, 91.7)      437      267  
[ 91.7,118.9)      365      360  
[118.9,318.3]      263      410
```

```
prop.table(t1,1)
```

```
cutWage      1. Industrial 2. Information  
[ 20.1, 91.7)      0.6207      0.3793  
[ 91.7,118.9)      0.5034      0.4966  
[118.9,318.3]      0.3908      0.6092
```

Density plots

```
qplot(wage, colour=education, data=training, geom="density")
```



Notes and further reading

- Make your plots only in the training set
 - Don't use the test set for exploration!
- Things you should be looking for
 - Imbalance in outcomes/predictors
 - Outliers
 - Groups of points not explained by a predictor
 - Skewed variables
- [ggplot2 tutorial](#)
- [caret visualizations](#)

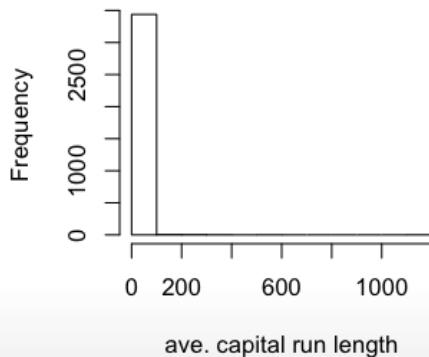


Preprocessing

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Why preprocess?

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain, ]
testing <- spam[-inTrain, ]
hist(training$capitalAve,main="",xlab="ave. capital run length")
```



Why preprocess?

```
mean(training$capitalAve)
```

```
[1] 4.709
```

```
sd(training$capitalAve)
```

```
[1] 25.48
```

Standardizing

```
trainCapAve <- training$capitalAve  
trainCapAveS <- (trainCapAve - mean(trainCapAve))/sd(trainCapAve)  
mean(trainCapAveS)
```

```
[1] 5.862e-18
```

```
sd(trainCapAveS)
```

```
[1] 1
```

Standardizing - test set

```
testCapAve <- testing$capitalAve  
testCapAveS <- (testCapAve - mean(trainCapAve))/sd(trainCapAve)  
mean(testCapAveS)
```

```
[1] 0.07579
```

```
sd(testCapAveS)
```

```
[1] 1.79
```

Standardizing - *preProcess* function

```
preObj <- preProcess(training[,-58],method=c("center","scale"))
trainCapAveS <- predict(preObj,training[,-58])$capitalAve
mean(trainCapAveS)
```

```
[1] 5.862e-18
```

```
sd(trainCapAveS)
```

```
[1] 1
```

Standardizing - *preProcess* function

```
testCapAveS <- predict(preObj,testing[,-58])$capitalAve  
mean(testCapAveS)
```

```
[1] 0.07579
```

```
sd(testCapAveS)
```

```
[1] 1.79
```

Standardizing - *preProcess* argument

```
set.seed(32343)
modelFit <- train(type ~., data=training,
                    preProcess=c("center", "scale"), method="glm")
modelFit
```

3451 samples
57 predictors
2 classes: 'nonspam', 'spam'

Pre-processing: centered, scaled
Resampling: Bootstrap (25 reps)

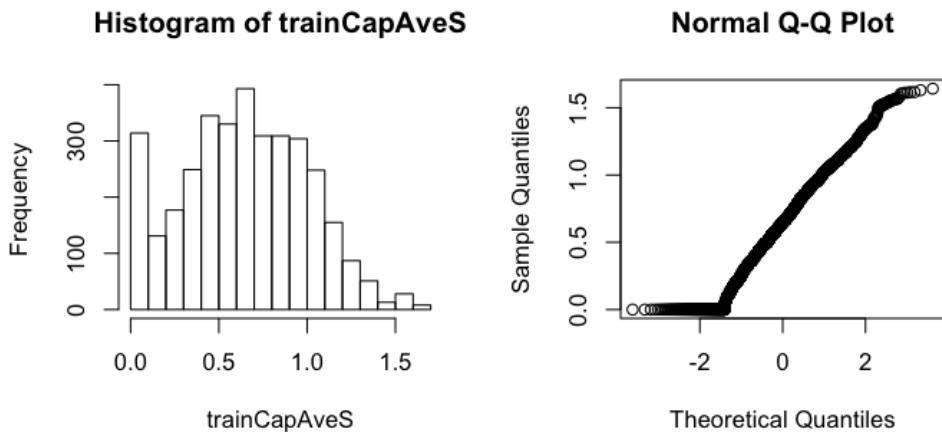
Summary of sample sizes: 3451, 3451, 3451, 3451, 3451, 3451, ...

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.9	0.8	0.007	0.01

Standardizing - Box-Cox transforms

```
preObj <- preProcess(training[,-58],method=c("BoxCox"))
trainCapAveS <- predict(preObj,training[,-58])$capitalAve
par(mfrow=c(1,2)); hist(trainCapAveS); qqnorm(trainCapAveS)
```



Standardizing - Imputing data

```
set.seed(13343)

# Make some values NA
training$capAve <- training$capitalAve
selectNA <- rbinom(dim(training)[1],size=1,prob=0.05)==1
training$capAve[selectNA] <- NA

# Impute and standardize
preObj <- preProcess(training[,-58],method="knnImpute")
capAve <- predict(preObj,training[,-58])$capAve

# Standardize true values
capAveTruth <- training$capitalAve
capAveTruth <- (capAveTruth-mean(capAveTruth))/sd(capAveTruth)
```

Standardizing - Imputing data

```
quantile(capAve - capAveTruth)
```

0%	25%	50%	75%	100%
-1.1324388	-0.0030842	-0.0015074	-0.0007467	0.2155789

```
quantile( (capAve - capAveTruth)[selectNA] )
```

0%	25%	50%	75%	100%
-0.9243043	-0.0125489	-0.0001968	0.0194524	0.2155789

```
quantile( (capAve - capAveTruth)[ !selectNA ] )
```

0%	25%	50%	75%	100%
-1.1324388	-0.0030033	-0.0015115	-0.0007938	-0.0001968

Notes and further reading

- Training and test must be processed in the same way
- Test transformations will likely be imperfect
 - Especially if the test/training sets collected at different times
- Careful when transforming factor variables!
- [preprocessing with caret](#)



Covariate creation

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Two levels of covariate creation

Level 1: From raw data to covariate

Hi

WE'VE DISCOVERED YOU ARE THE
HEIR TO AN INCREDIBLE FORTUNE.
PLEASE SUBMIT YOUR NAME,
ADDRESS AND BANK ACCOUNT SO
WE CAN SEND YOU \$\$\$\$\$\$.



	capitalAve	you	numDollar	...
1		2	8	...
2				...
3				...

JOE JOHNSON

Level 2: Transforming tidy covariates

```
library(kernlab); data(spam)  
spam$capitalAveSq <- spam$capitalAve^2
```

Level 1, Raw data -> covariates

- Depends heavily on application
- The balancing act is summarization vs. information loss
- Examples:
 - Text files: frequency of words, frequency of phrases ([Google ngrams](#)), frequency of capital letters.
 - Images: Edges, corners, blobs, ridges ([computer vision feature detection](#))
 - Webpages: Number and type of images, position of elements, colors, videos ([A/B Testing](#))
 - People: Height, weight, hair color, sex, country of origin.
- The more knowledge of the system you have the better the job you will do.
- When in doubt, err on the side of more features
- Can be automated, but use caution!

Level 2, Tidy covariates -> new covariates

- More necessary for some methods (regression, svms) than for others (classification trees).
- Should be done *only on the training set*
- The best approach is through exploratory analysis (plotting/tables)
- New covariates should be added to data frames

Load example data

```
library(ISLR); library(caret); data(Wage);
inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]
```

Common covariates to add, dummy variables

Basic idea - convert factor variables to [indicator variables](#)

```
table(training$jobclass)
```

1. Industrial 2. Information

1090 1012

```
dummies <- dummyVars(wage ~ jobclass,data=training)
head(predict(dummies,newdata=training))
```

jobclass.1. Industrial jobclass.2. Information

231655	1	0
86582	0	1
11141	0	1

Removing zero covariates

```
nsv <- nearZeroVar(training,saveMetrics=TRUE)  
nsv
```

	freqRatio	percentUnique	zeroVar	nzv
year	1.029	0.33302	FALSE	FALSE
age	1.122	2.80685	FALSE	FALSE
sex	0.000	0.04757	TRUE	TRUE
maritl	3.159	0.23787	FALSE	FALSE
race	8.529	0.19029	FALSE	FALSE
education	1.492	0.23787	FALSE	FALSE
region	0.000	0.04757	TRUE	TRUE
jobclass	1.077	0.09515	FALSE	FALSE
health	2.452	0.09515	FALSE	FALSE
health_ins	2.269	0.09515	FALSE	FALSE
logwage	1.198	17.26927	FALSE	FALSE
wage	1.185	18.07802	FALSE	FALSE

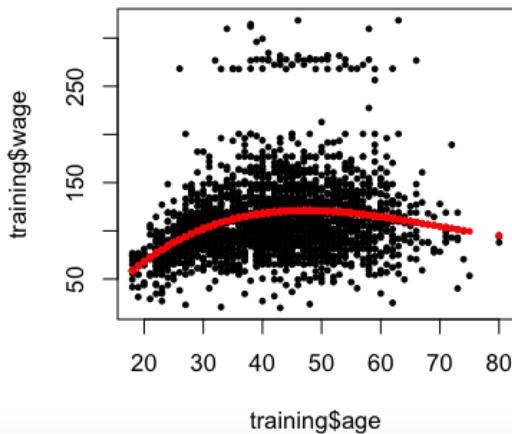
Spline basis

```
library(splines)
bsBasis <- bs(training$age,df=3)
bsBasis
```

	1	2	3
[1,]	0.00000 0.0000000 0.000e+00		
[2,]	0.23685 0.0253768 9.063e-04		
[3,]	0.44309 0.2436978 4.468e-02		
[4,]	0.43081 0.2910904 6.556e-02		
[5,]	0.42617 0.1482327 1.719e-02		
[6,]	0.41709 0.1331149 1.416e-02		
[7,]	0.31823 0.0540390 3.059e-03		
[8,]	0.36253 0.3866940 1.375e-01		
[9,]	0.44436 0.2275981 3.886e-02		
[10,]	0.20449 0.0179375 5.245e-04		
[11,]	0.07768 0.3601465 5.566e-01		
[12,]	0.13145 0.0066841 1.133e-04		
[13,]	0.39290 0.1042387 9.218e-03		
[14,]	0.26654 0.0339238 1.439e-03		
[15,]	0.20449 0.0179375 5.245e-04		

Fitting curves with splines

```
lm1 <- lm(wage ~ bsBasis,data=training)
plot(training$age,training$wage,pch=19,cex=0.5)
points(training$age,predict(lm1,newdata=training),col="red",pch=19,cex=0.5)
```



Splines on the test set

```
predict(bsBasis,age=testing$age)
```

	1	2	3
[1,]	0.00000	0.0000000	0.000e+00
[2,]	0.23685	0.0253768	9.063e-04
[3,]	0.44309	0.2436978	4.468e-02
[4,]	0.43081	0.2910904	6.556e-02
[5,]	0.42617	0.1482327	1.719e-02
[6,]	0.41709	0.1331149	1.416e-02
[7,]	0.31823	0.0540390	3.059e-03
[8,]	0.36253	0.3866940	1.375e-01
[9,]	0.44436	0.2275981	3.886e-02
[10,]	0.20449	0.0179375	5.245e-04
[11,]	0.07768	0.3601465	5.566e-01
[12,]	0.13145	0.0066841	1.133e-04
[13,]	0.39290	0.1042387	9.218e-03
[14,]	0.26654	0.0339238	1.439e-03
[15,]	0.20449	0.0179375	5.245e-04
[16,]	0.29109	0.4308138	2.125e-01
[17,]	0.23685	0.0253768	9.063e-04

Notes and further reading

- Level 1 feature creation (raw data to covariates)
 - Science is key. Google "feature extraction for [data type]"
 - Err on overcreation of features
 - In some applications (images, voices) automated feature creation is possible/necessary
 - <http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf>
- Level 2 feature creation (covariates to new covariates)
 - The function *preProcess* in *caret* will handle some preprocessing.
 - Create new covariates if you think they will improve fit
 - Use exploratory analysis on the training set for creating them
 - Be careful about overfitting!
- [preprocessing with caret](#)
 - If you want to fit spline models, use the *gam* method in the *caret* package which allows smoothing of multiple variables.
 - More on feature creation/data tidying in the Obtaining Data course from the Data Science *course*



Preprocessing with Principal Components Analysis (PCA)

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Correlated predictors

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain, ]
testing <- spam[-inTrain, ]

M <- abs(cor(training[, -58]))
diag(M) <- 0
which(M > 0.8, arr.ind=T)
```

	row	col
num415	34	32
num857	32	34

Correlated predictors

```
names(spam)[c(34,32)]
```

```
[1] "num415" "num857"
```

```
plot(spam[,34],spam[,32])
```

Basic PCA idea

- We might not need every predictor
- A weighted combination of predictors might be better
- We should pick this combination to capture the "most information" possible
- Benefits
 - Reduced number of predictors
 - Reduced noise (due to averaging)

We could rotate the plot

$$X = 0.71 \times \text{num415} + 0.71 \times \text{num857}$$

$$Y = 0.71 \times \text{num415} - 0.71 \times \text{num857}$$

```
X <- 0.71*training$num415 + 0.71*training$num857  
Y <- 0.71*training$num415 - 0.71*training$num857  
plot(X,Y)
```

Related problems

You have multivariate variables X_1, \dots, X_n so $X_1 = (X_{11}, \dots, X_{1m})$

- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is **statistical** and the second goal is **data compression**.

Related solutions - PCA/SVD

SVD

If X is a matrix with each variable in a column and each observation in a row then the SVD is a "matrix decomposition"

$$X = UDV^T$$

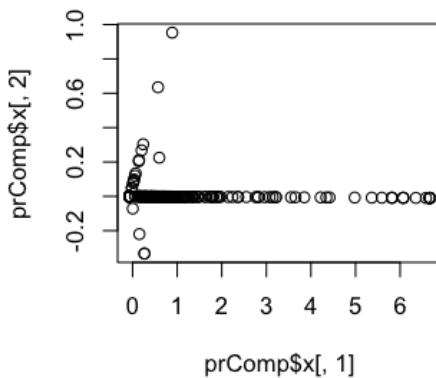
where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singular vectors) and D is a diagonal matrix (singular values).

PCA

The principal components are equal to the right singular values if you first scale (subtract the mean, divide by the standard deviation) the variables.

Principal components in R - prcomp

```
smallSpam <- spam[,c(34,32)]  
prComp <- prcomp(smallSpam)  
plot(prComp$x[,1],prComp$x[,2])
```



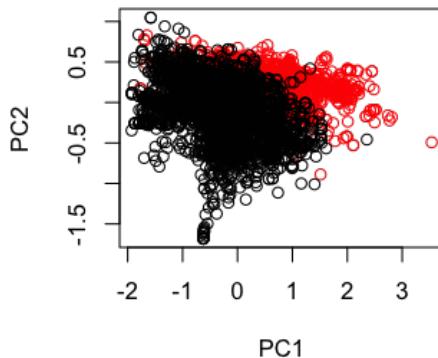
Principal components in R - prcomp

```
prComp$rotation
```

	PC1	PC2
num415	0.7081	0.7061
num857	0.7061	-0.7081

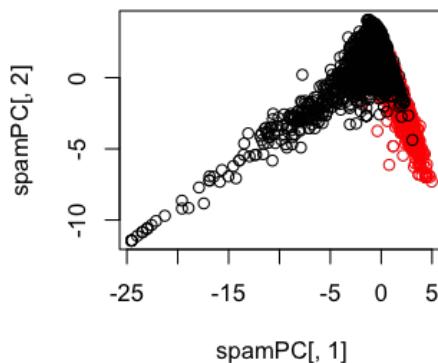
PCA on SPAM data

```
typeColor <- ((spam$type=="spam")*1 + 1)
prComp <- prcomp(log10(spam[, -58]+1))
plot(prComp$x[, 1], prComp$x[, 2], col=typeColor, xlab="PC1", ylab="PC2")
```



PCA with caret

```
preProc <- preProcess(log10(spam[, -58]+1), method="pca", pcaComp=2)
spamPC <- predict(preProc, log10(spam[, -58]+1))
plot(spamPC[, 1], spamPC[, 2], col=typeColor)
```



Preprocessing with PCA

```
preProc <- preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC <- predict(preProc,log10(training[,-58]+1))
modelFit <- train(training$type ~ .,method="glm",data=trainPC)
```

Preprocessing with PCA

```
testPC <- predict(preProc,log10(testing[,-58]+1))
confusionMatrix(testing$type,predict(modelFit,testPC))
```

Confusion Matrix and Statistics

		Reference	
		nonspam	spam
Prediction	nonspam	646	51
	spam	64	389

Accuracy : 0.9
95% CI : (0.881, 0.917)

No Information Rate : 0.617
P-Value [Acc > NIR] : <2e-16

Kappa : 0.79
McNemar's Test P-Value : 0.263

Sensitivity : 0.910
Specificity : 0.884

Alternative (sets # of PCs)

```
modelFit <- train(training$type ~ .,method="glm",preProcess="pca",data=training)
confusionMatrix(testing$type,predict(modelFit,testing))
```

Confusion Matrix and Statistics

Reference
Prediction nonspam spam

nonspam	660	37
spam	54	399

Accuracy : 0.921
95% CI : (0.904, 0.936)

No Information Rate : 0.621
P-Value [Acc > NIR] : <2e-16

Kappa : 0.833
McNemar's Test P-Value : 0.0935

Sensitivity : 0.924
Specificity : 0.915

Final thoughts on PCs

- Most useful for linear-type models
- Can make it harder to interpret predictors
- Watch out for outliers!
 - Transform first (with logs/Box Cox)
 - Plot predictors to identify problems
- For more info see
 - Exploratory Data Analysis
 - [Elements of Statistical Learning](#)



Predicting with regression

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Fit a simple regression model
- Plug in new covariates and multiply by the coefficients
- Useful when the linear model is (nearly) correct

Pros:

- Easy to implement
- Easy to interpret

Cons:

- Often poor performance in nonlinear settings

Example: Old faithful eruptions



(c) Wally Pacholka / AstroPics.com

Image Credit/Copyright Wally Pacholka <http://www.astropics.com/>

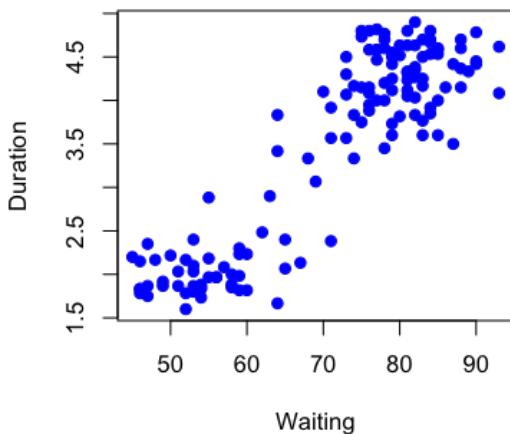
Example: Old faithful eruptions

```
library(caret); data(faithful); set.seed(333)
inTrain <- createDataPartition(y=faithful$waiting,
                               p=0.5, list=FALSE)
trainFaith <- faithful[inTrain,]; testFaith <- faithful[-inTrain,]
head(trainFaith)
```

	eruptions	waiting
6	2.883	55
11	1.833	54
16	2.167	52
19	1.600	52
22	1.750	47
27	1.967	55

Eruption duration versus waiting time

```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
```



Fit a linear model

$$ED_i = b_0 + b_1 WT_i + e_i$$

```
lm1 <- lm(eruptions ~ waiting,data=trainFaith)
summary(lm1)
```

Call:

```
lm(formula = eruptions ~ waiting, data = trainFaith)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.2699	-0.3479	0.0398	0.3659	1.0502

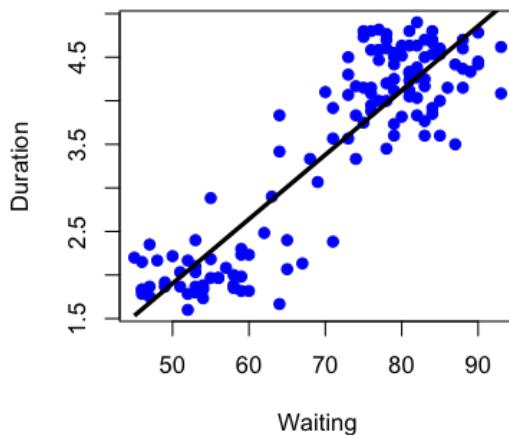
Coefficients:

	Estimate	Std. Error	t value	Pr(> t)							
(Intercept)	-1.79274	0.22787	-7.87	1e-12 ***							
waiting	0.07390	0.00315	23.47	<2e-16 ***							

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

Model fit

```
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")  
lines(trainFaith$waiting,lm1$fitted,lwd=3)
```



Predict a new value

$$\hat{ED} = \hat{b}_0 + \hat{b}_1 WT$$

```
coef(lm1)[1] + coef(lm1)[2]*80
```

```
(Intercept)
```

```
4.119
```

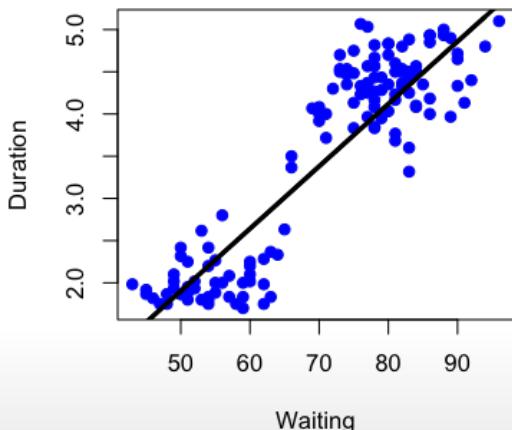
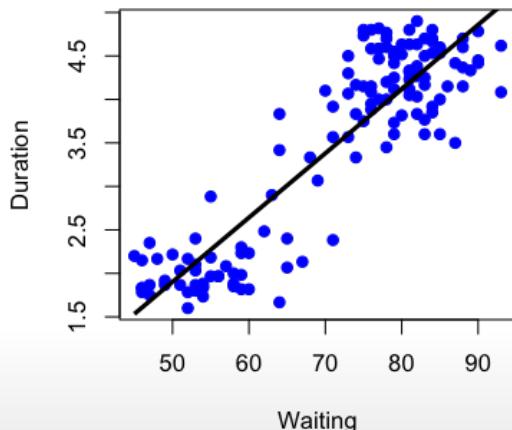
```
newdata <- data.frame(waiting=80)  
predict(lm1,newdata)
```

```
1
```

```
4.119
```

Plot predictions - training and test

```
par(mfrow=c(1,2))
plot(trainFaith$waiting,trainFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(trainFaith$waiting,predict(lm1),lwd=3)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue",xlab="Waiting",ylab="Duration")
lines(testFaith$waiting,predict(lm1,newdata=testFaith),lwd=3)
```



Get training set/test set errors

```
# Calculate RMSE on training  
sqrt(sum((lm1$fitted-trainFaith$eruptions)^2))
```

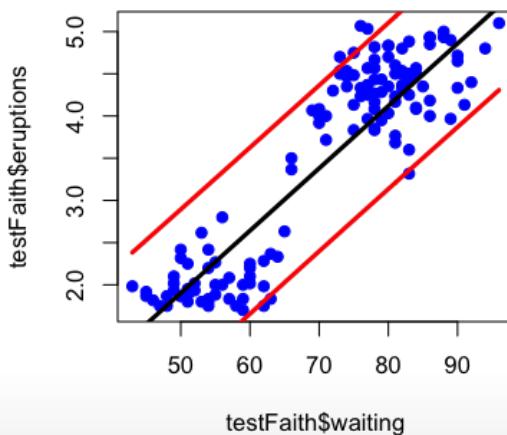
```
[1] 5.752
```

```
# Calculate RMSE on test  
sqrt(sum((predict(lm1,newdata=testFaith)-testFaith$eruptions)^2))
```

```
[1] 5.839
```

Prediction intervals

```
pred1 <- predict(lm1,newdata=testFaith,interval="prediction")
ord <- order(testFaith$waiting)
plot(testFaith$waiting,testFaith$eruptions,pch=19,col="blue")
matlines(testFaith$waiting[ord],pred1[ord,],type="l",,col=c(1,2,2),lty = c(1,1,1), lwd=3)
```



Same process with caret

```
modFit <- train(eruptions ~ waiting,data=trainFaith,method="lm")
summary(modFit$finalModel)
```

Call:

```
lm(formula = modFormula, data = data)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.2699	-0.3479	0.0398	0.3659	1.0502

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)							
(Intercept)	-1.79274	0.22787	-7.87	1e-12 ***							
waiting	0.07390	0.00315	23.47	<2e-16 ***							

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

Residual standard error: 0.495 on 135 degrees of freedom

Multiple R-squared: 0.803, Adjusted R-squared: 0.802

Notes and further reading

- Regression models with multiple covariates can be included
- Often useful in combination with other models
- [Elements of statistical learning](#)
- [Modern applied statistics with S](#)
- [Introduction to statistical learning](#)



Predicting with regression, multiple covariates

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Example: predicting wages

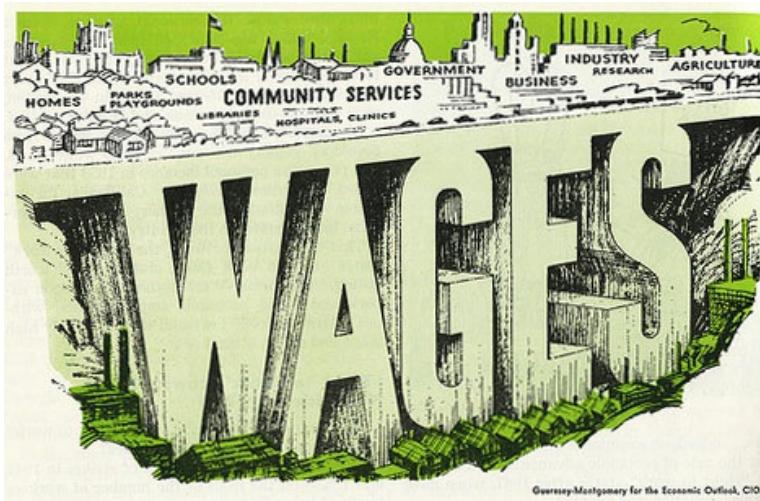


Image Credit <http://www.cahs-media.org/the-high-cost-of-low-wages>

Data from: [ISLR package](#) from the book: [Introduction to statistical learning](#)

Example: Wage data

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

year	age	sex	maritl	race
Min. :2003	Min. :18.0	1. Male :3000	1. Never Married: 648	1. White:2480
1st Qu.:2004	1st Qu.:33.8	2. Female: 0	2. Married :2074	2. Black: 293
Median :2006	Median :42.0		3. Widowed : 19	3. Asian: 190
Mean :2006	Mean :42.4		4. Divorced : 204	4. Other: 37
3rd Qu.:2008	3rd Qu.:51.0		5. Separated : 55	
Max. :2009	Max. :80.0			
education		region	jobclass	health
1. < HS Grad	:268	2. Middle Atlantic :3000	1. Industrial :1544	1. <=Good : 858
2. HS Grad	:971	1. New England : 0	2. Information:1456	2. >=Very Good:2142
3. Some College	:650	3. East North Central: 0		
4. College Grad	:685	4. West North Central: 0		
5. Advanced Degree	:426	5. South Atlantic : 0		
		6. East South Central: 0		
		(Other) : 0		

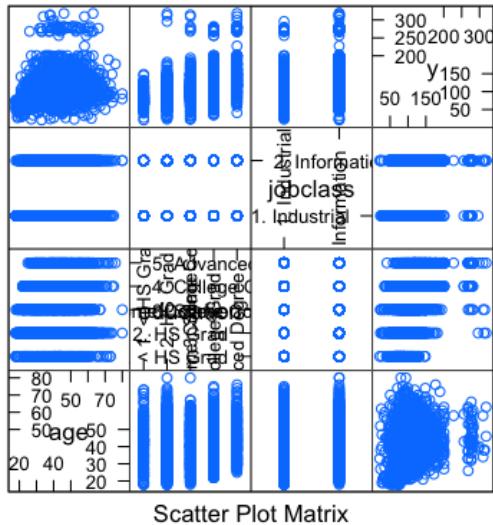
Get training/test sets

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

```
[1] 898 12
```

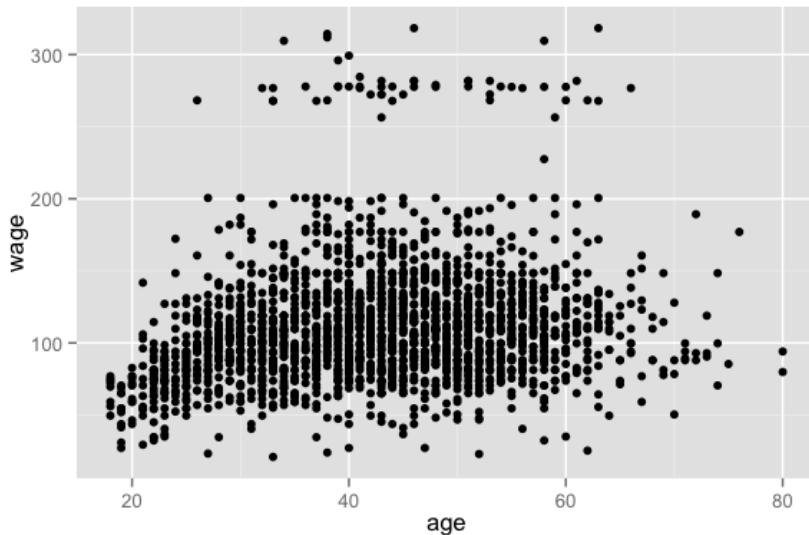
Feature plot

```
featurePlot(x=training[,c("age","education","jobclass")],  
            y = training$wage,  
            plot="pairs")
```



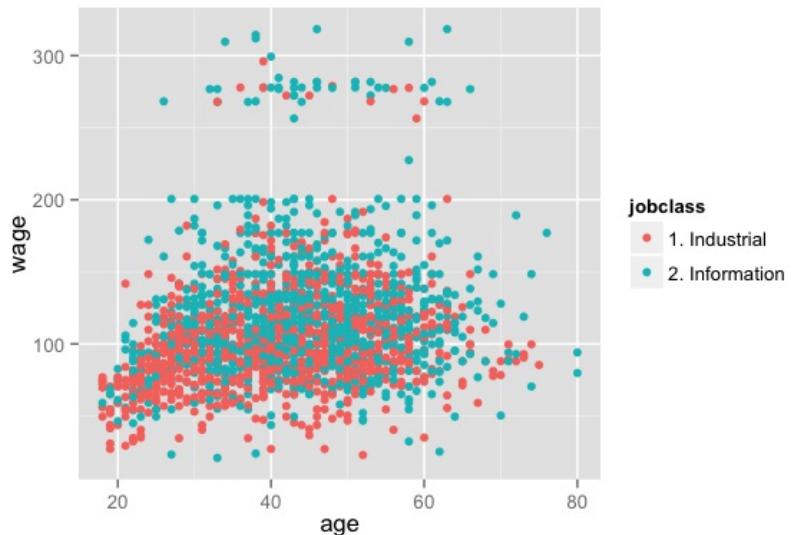
Plot age versus wage

```
qplot(age,wage,data=training)
```



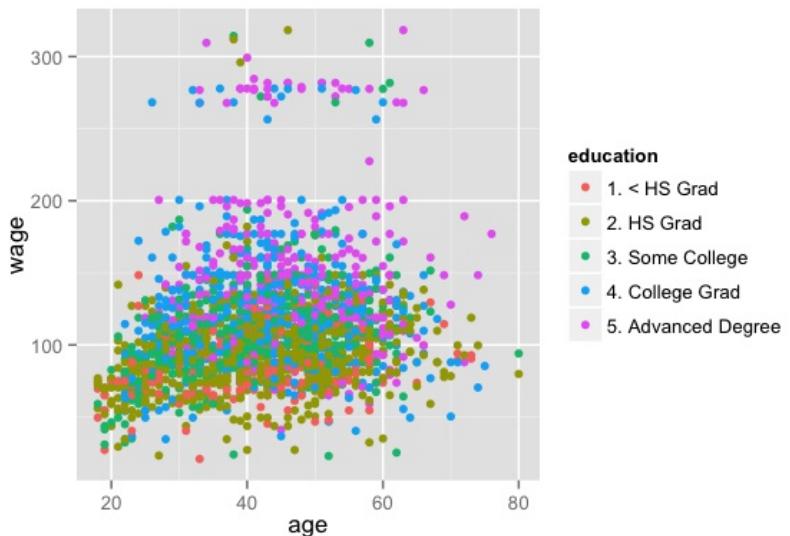
Plot age versus wage colour by jobclass

```
qplot(age,wage,colour=jobclass,data=training)
```



Plot age versus wage colour by education

```
qplot(age,wage,colour=education,data=training)
```



Fit a linear model

$$ED_i = b_0 + b_1 \text{age} + b_2 I(\text{Jobclass}_i = "Information") + \sum_{k=1}^4 \gamma_k I(\text{education}_i = \text{level}k)$$

```
modFit<- train(wage ~ age + jobclass + education,  
                 method = "lm", data=training)  
finMod <- modFit$finalModel  
print(modFit)
```

Linear Regression

2102 samples
11 predictors

No pre-processing

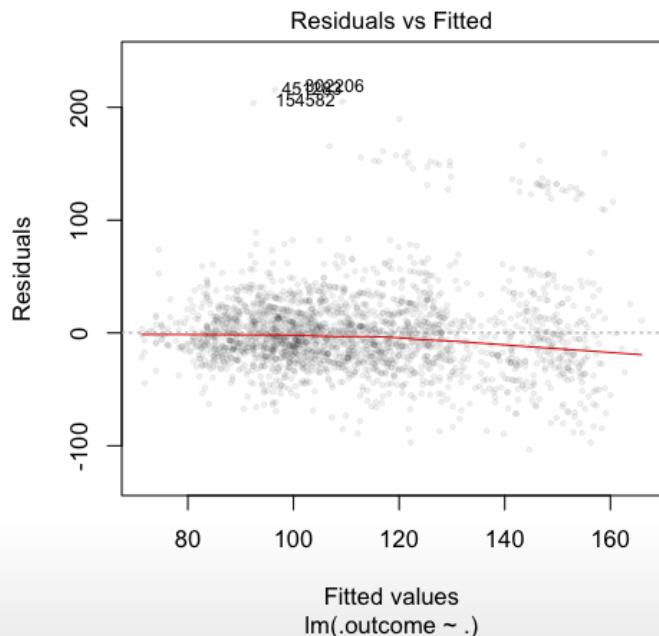
Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...

Resampling results

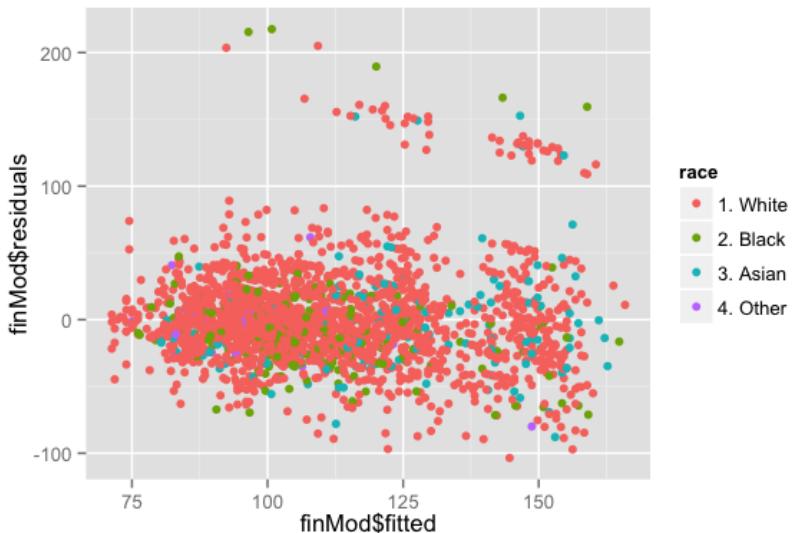
Diagnostics

```
plot(finMod, 1, pch=19, cex=0.5, col="#00000010")
```



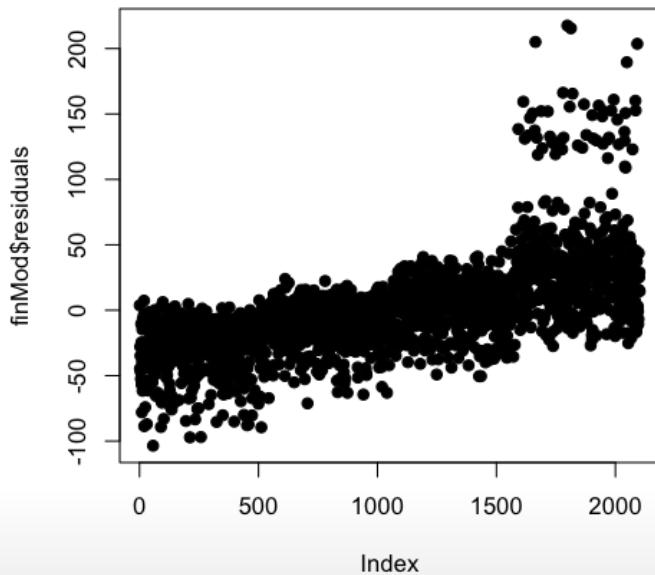
Color by variables not used in the model

```
qplot(finMod$fitted,finMod$residuals,colour=race,data=training)
```



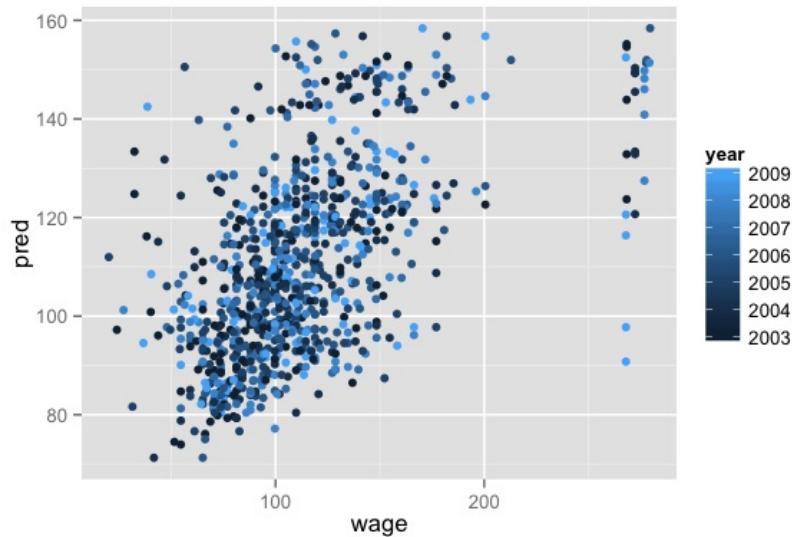
Plot by index

```
plot(finMod$residuals,pch=19)
```



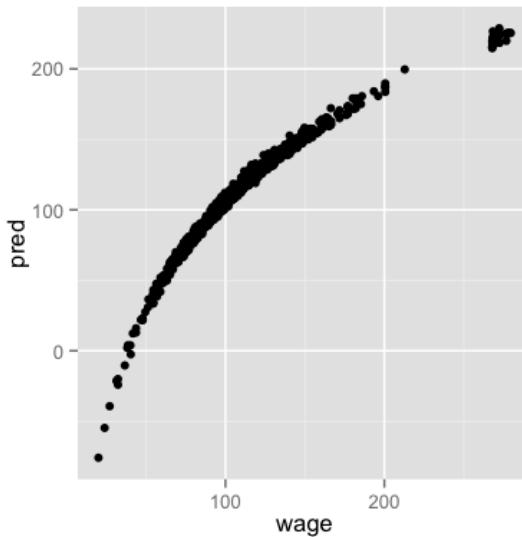
Predicted versus truth in test set

```
pred <- predict(modFit, testing)
qplot(wage,pred,colour=year,data=testing)
```



If you want to use all covariates

```
modFitAll<- train(wage ~ .,data=training,method="lm")  
pred <- predict(modFitAll, testing)  
qplot(wage,pred,data=testing)
```



Notes and further reading

- Often useful in combination with other models
- [Elements of statistical learning](#)
- [Modern applied statistics with S](#)
- [Introduction to statistical learning](#)



Predicting with trees

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Iteratively split variables into groups
- Evaluate "homogeneity" within each group
- Split again if necessary

Pros:

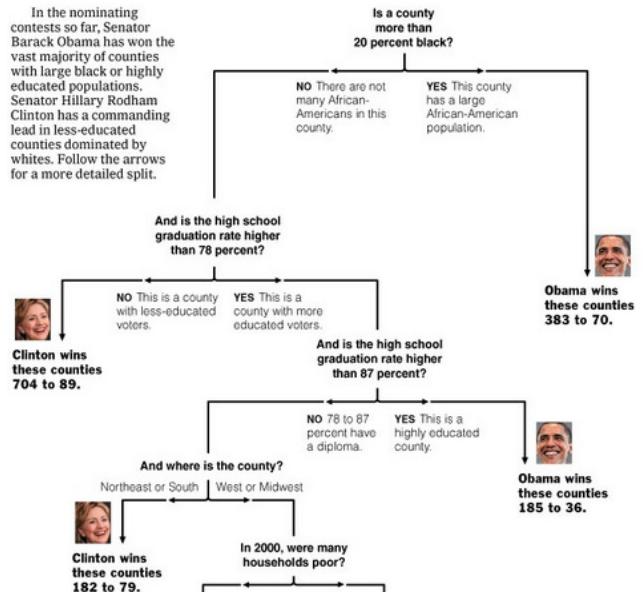
- Easy to interpret
- Better performance in nonlinear settings

Cons:

- Without pruning/cross-validation can lead to overfitting
- Harder to estimate uncertainty
- Results may be variable

Example Tree

Decision Tree: The Obama-Clinton Divide



<http://graphics8.nytimes.com/images/2008/04/16/us/0416-nat-subOBAMA.jpg>

Basic algorithm

1. Start with all variables in one group
2. Find the variable/split that best separates the outcomes
3. Divide the data into two groups ("leaves") on that split ("node")
4. Within each split, find the best variable/split that separates the outcomes
5. Continue until the groups are too small or sufficiently "pure"

Measures of impurity

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \text{ in Leaf } m} \mathbb{I}(y_i = k)$$

Misclassification Error:

$$1 - \hat{p}_{m_k(m)}; k(m) = \text{most common}$$

- 0 = perfect purity
- 0.5 = no purity

Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \times \hat{p}_{mk'} = \sum_{k=1}^K \hat{p}_{mk} (1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^K p_{mk}^2$$

- 0 = perfect purity
- 0.5 = no purity

Measures of impurity

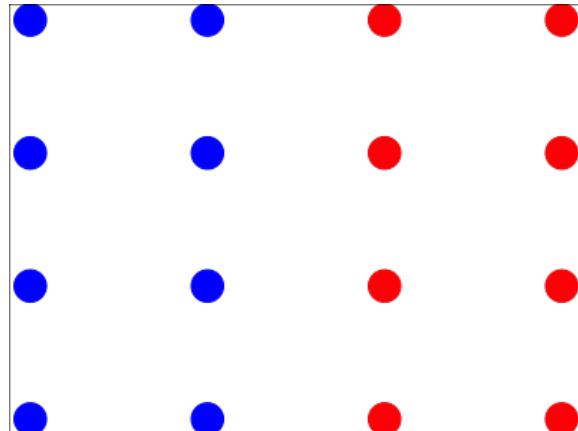
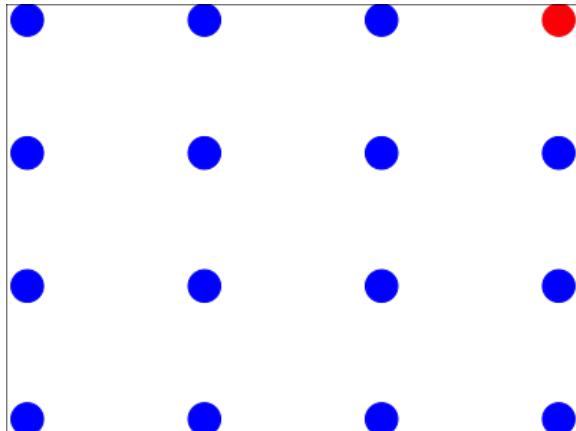
Deviance/information gain:

$$-\sum_{k=1}^K \hat{p}_{mk} \log_2 \hat{p}_{mk}$$

- 0 = perfect purity
- 1 = no purity

http://en.wikipedia.org/wiki/Decision_tree_learning

Measures of impurity



- **Misclassification:** $1/16 = 0.06$
- **Gini:** $1 - [(1/16)^2 + (15/16)^2] = 0.12$
- **Information:**
 $-[1/16 \times \log_2(1/16) + 15/16 \times \log_2(15/16)] = 0.34$

- **Misclassification:** $8/16 = 0.5$
- **Gini:** $1 - [(8/16)^2 + (8/16)^2] = 0.5$
- **Information:**
 $-[1/16 \times \log_2(1/16) + 15/16 \times \log_2(15/16)] = 1$

Example: Iris Data

```
data(iris); library(ggplot2)  
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"   "Petal.Length"  "Petal.Width"   "Species"
```

```
table(iris$Species)
```

setosa	versicolor	virginica
50	50	50

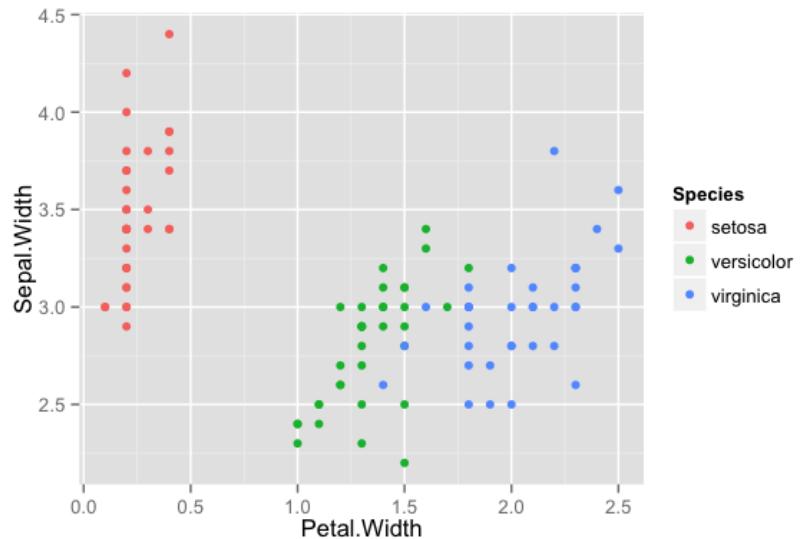
Create training and test sets

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

```
[1] 45 5
```

Iris petal widths/sepal width

```
qplot(Petal.Width,Sepal.Width,colour=Species,data=training)
```



Iris petal widths/sepal width

```
library(caret)
modFit <- train(Species ~ .,method="rpart",data=training)
print(modFit$finalModel)
```

n= 105

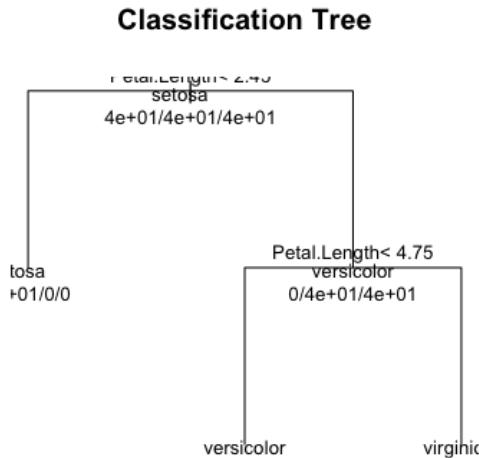
node), split, n, loss, yval, (yprob)

* denotes terminal node

- 1) root 105 70 setosa (0.3333 0.3333 0.3333)
- 2) Petal.Length< 2.45 35 0 setosa (1.0000 0.0000 0.0000) *
- 3) Petal.Length>=2.45 70 35 versicolor (0.0000 0.5000 0.5000)
 - 6) Petal.Length< 4.75 31 0 versicolor (0.0000 1.0000 0.0000) *
 - 7) Petal.Length>=4.75 39 4 virginica (0.0000 0.1026 0.8974) *

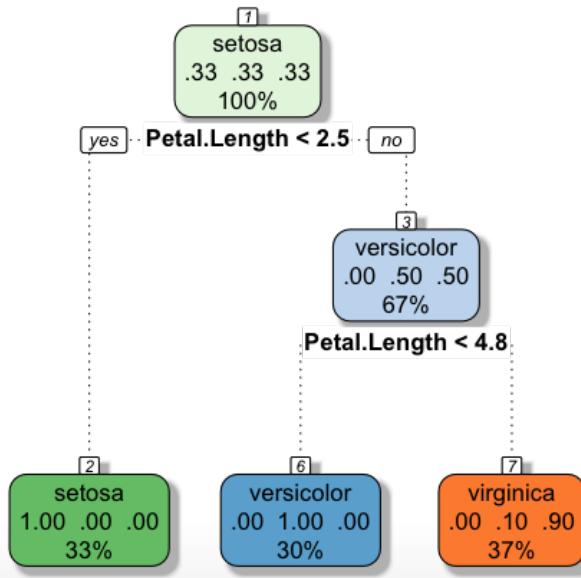
Plot tree

```
plot(modFit$finalModel, uniform=TRUE,  
     main="Classification Tree")  
text(modFit$finalModel, use.n=TRUE, all=TRUE, cex=.8)
```



Prettier plots

```
library(rattle)  
fancyRpartPlot(modFit$finalModel)
```



Predicting new values

```
predict(modFit,newdata=testing)
```

```
[1] setosa    setosa    setosa    setosa    setosa    setosa    setosa    setosa  
[9] setosa    setosa    setosa    setosa    setosa    setosa    setosa    versicolor  
[17] versicolor versicolor versicolor versicolor versicolor versicolor versicolor  
[25] virginica versicolor virginica versicolor versicolor virginica virginica  
[33] virginica versicolor virginica virginica virginica virginica virginica virginica  
[41] virginica virginica virginica virginica virginica  
Levels: setosa versicolor virginica
```

Notes and further resources

- Classification trees are non-linear models
 - They use interactions between variables
 - Data transformations may be less important (monotone transformations)
 - Trees can also be used for regression problems (continuous outcome)
- Note that there are multiple tree building options in R both in the caret package - [party](#), [rpart](#) and out of the caret package - [tree](#)
- [Introduction to statistical learning](#)
- [Elements of Statistical Learning](#)
- [Classification and regression trees](#)



Bagging

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Bootstrap aggregating (bagging)

Basic idea:

1. Resample cases and recalculate predictions
2. Average or majority vote

Notes:

- Similar bias
- Reduced variance
- More useful for non-linear functions

Ozone data

```
library(ElemStatLearn); data(ozone, package="ElemStatLearn")
ozone <- ozone[order(ozone$ozone), ]
head(ozone)
```

	ozone	radiation	temperature	wind
17	1	8	59	9.7
19	4	25	61	9.7
14	6	78	57	18.4
45	7	48	80	14.3
106	7	49	69	10.3
7	8	19	61	20.1

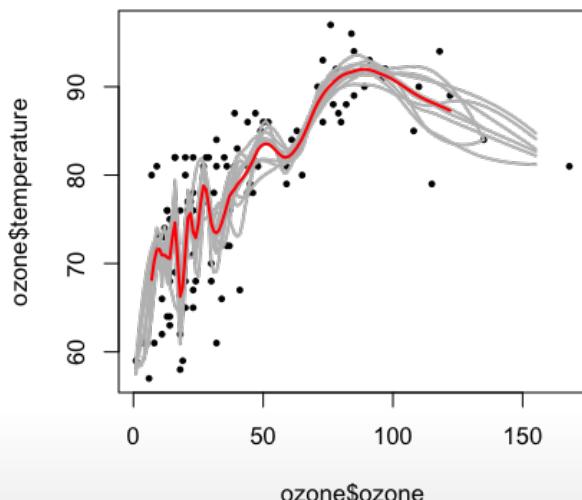
http://en.wikipedia.org/wiki/Bootstrap_aggregating

Bagged loess

```
ll <- matrix(NA,nrow=10,ncol=155)
for(i in 1:10){
  ss <- sample(1:dim(ozone)[1],replace=T)
  ozone0 <- ozone[ss,]; ozone0 <- ozone0[order(ozone0$ozone),]
  loess0 <- loess(temperature ~ ozone,data=ozone0,span=0.2)
  ll[i,] <- predict(loess0,newdata=data.frame(ozone=1:155))
}
```

Bagged loess

```
plot(ozone$ozone,ozone$temperature,pch=19,cex=0.5)
for(i in 1:10){lines(1:155,ll[i,],col="grey",lwd=2)}
lines(1:155,apply(ll,2,mean),col="red",lwd=2)
```



Bagging in caret

- Some models perform bagging for you, in `train` function consider `method` options
 - `bagEarth`
 - `treebag`
 - `bagFDA`
- Alternatively you can bag any model you choose using the `bag` function

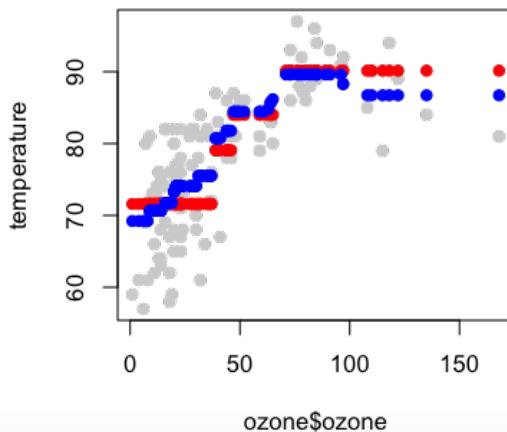
More bagging in caret

```
predictors = data.frame(ozone=ozone$ozone)
temperature = ozone$temperature
treebag <- bag(predictors, temperature, B = 10,
                 bagControl = bagControl(fit = ctreeBag$fit,
                                           predict = ctreeBag$pred,
                                           aggregate = ctreeBag$aggregate))
```

<http://www.inside-r.org/packages/cran/caret/docs/nbBag>

Example of custom bagging (continued)

```
plot(ozone$ozone,temperature,col='lightgrey',pch=19)
points(ozone$ozone,predict(treebag$fits[[1]]$fit,predictors),pch=19,col="red")
points(ozone$ozone,predict(treebag,predictors),pch=19,col="blue")
```



Parts of bagging

```
ctreeBag$fit
```

```
function (x, y, ...)  
{  
  library(party)  
  data <- as.data.frame(x)  
  data$y <- y  
  ctree(y ~ ., data = data)  
}  
<environment: namespace:caret>
```

Parts of bagging

```
ctreeBag$pred
```

```
function (object, x)
{
  obsLevels <- levels(object@data@get("response")[, 1])
  if (!is.null(obsLevels)) {
    rawProbs <- treeresponse(object, x)
    probMatrix <- matrix(unlist(rawProbs), ncol = length(obsLevels),
      byrow = TRUE)
    out <- data.frame(probMatrix)
    colnames(out) <- obsLevels
    rownames(out) <- NULL
  }
  else out <- unlist(treeresponse(object, x))
  out
}
```

<environment: namespace:caret>

Parts of bagging

```
ctreeBag$aggregate
```

```
function (x, type = "class")
{
  if (is.matrix(x[[1]]) | is.data.frame(x[[1]])) {
    pooled <- x[[1]] & NA
    classes <- colnames(pooled)
    for (i in 1:ncol(pooled)) {
      tmp <- lapply(x, function(y, col) y[, col], col = i)
      tmp <- do.call("rbind", tmp)
      pooled[, i] <- apply(tmp, 2, median)
    }
    if (type == "class") {
      out <- factor(classes[apply(pooled, 1, which.max)],
                     levels = classes)
    }
    else out <- as.data.frame(pooled)
  }
  else {
    x <- matrix(unlist(x), ncol = length(x))
```

Notes and further resources

Notes:

- Bagging is most useful for nonlinear models
- Often used with trees - an extension is random forests
- Several models use bagging in caret's *train* function

Further resources:

- [Bagging](#)
- [Bagging and boosting](#)
- [Elements of Statistical Learning](#)



Random forests

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

Random forests

1. Bootstrap samples
2. At each split, bootstrap variables
3. Grow multiple trees and vote

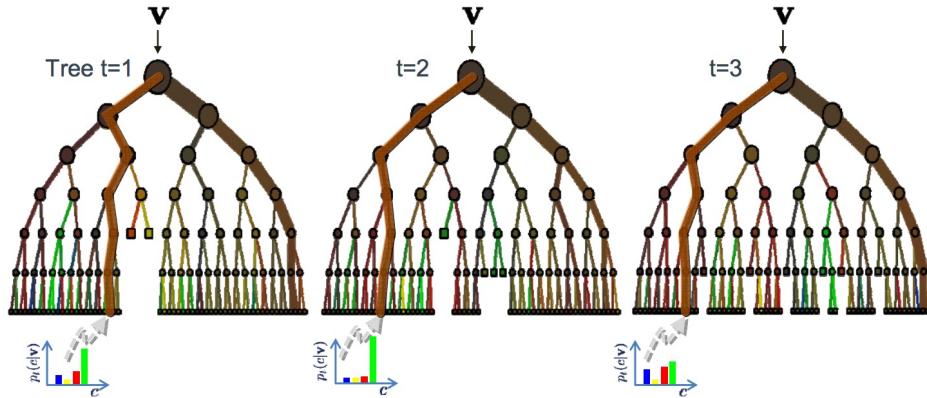
Pros:

1. Accuracy

Cons:

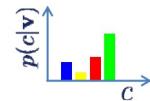
1. Speed
2. Interpretability
3. Overfitting

Random forests



The ensemble model

$$\text{Forest output probability } p(\mathbf{c}|\mathbf{v}) = \frac{1}{T} \sum_t p_t(\mathbf{c}|\mathbf{v})$$



<http://www.robots.ox.ac.uk/~az/lectures/ml/lect5.pdf>

Iris data

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

Random forests

```
library(caret)
modFit <- train(Species~ ., data=training, method="rf", prox=TRUE)
modFit
```

105 samples
4 predictors
3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing

Resampling: Bootstrap (25 reps)

Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...

Resampling results across tuning parameters:

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.9	0.9	0.03	0.04
3	0.9	0.9	0.03	0.05
4	0.9	0.9	0.03	0.05

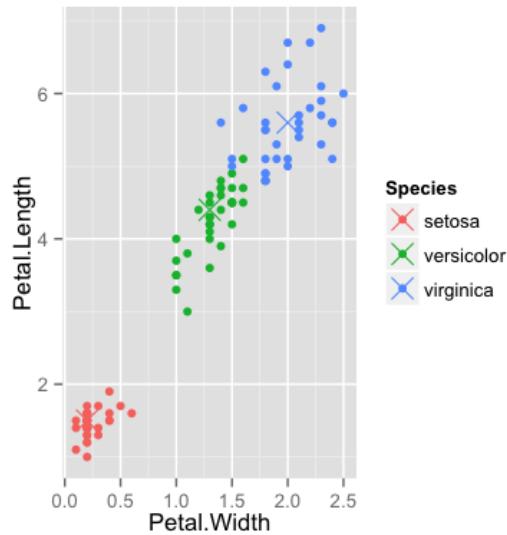
Getting a single tree

```
getTree(modFit$finalModel, k=2)
```

	left	daughter	right	daughter	split	var	split	point	status	prediction
1		2		3		4		0.70	1	0
2		0		0		0		0.00	-1	1
3		4		5		4		1.70	1	0
4		6		7		3		4.95	1	0
5		8		9		3		4.85	1	0
6		0		0		0		0.00	-1	2
7		10		11		4		1.55	1	0
8		12		13		1		5.95	1	0
9		0		0		0		0.00	-1	3
10		0		0		0		0.00	-1	3
11		0		0		0		0.00	-1	2
12		0		0		0		0.00	-1	2
13		0		0		0		0.00	-1	3

Class "centers"

```
irisP <- classCenter(training[,c(3,4)], training$Species, modFit$finalModel$prox)
irisP <- as.data.frame(irisP); irisP$Species <- rownames(irisP)
p <- qplot(Petal.Width, Petal.Length, col=Species,data=training)
p + geom_point(aes(x=Petal.Width,y=Petal.Length,col=Species),size=5,shape=4,data=irisP)
```



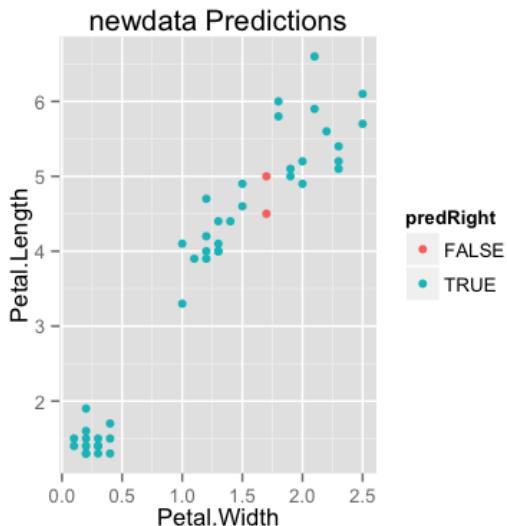
Predicting new values

```
pred <- predict(modFit,testing); testing$predRight <- pred==testing$Species  
table(pred,testing$Species)
```

pred	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	14	1
virginica	0	1	14

Predicting new values

```
qplot(Petal.Width,Petal.Length,colour=predRight,data=testing,main="newdata Predictions")
```



Notes and further resources

Notes:

- Random forests are usually one of the two top performing algorithms along with boosting in prediction contests.
- Random forests are difficult to interpret but often very accurate.
- Care should be taken to avoid overfitting (see [rfcv](#) function)

Further resources:

- [Random forests](#)
- [Random forest Wikipedia](#)
- [Elements of Statistical Learning](#)



Boosting

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Basic idea

1. Take lots of (possibly) weak predictors
2. Weight them and add them up
3. Get a stronger predictor

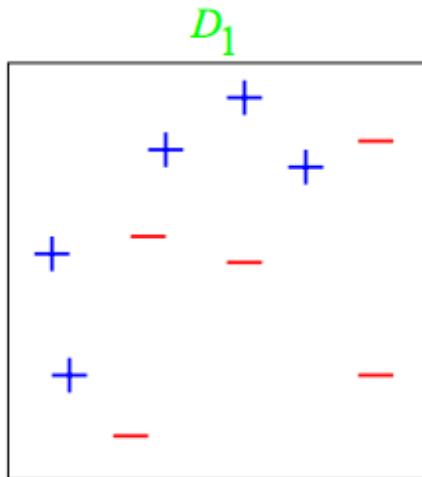
Basic idea behind boosting

1. Start with a set of classifiers h_1, \dots, h_k
 - Examples: All possible trees, all possible regression models, all possible cutoffs.
2. Create a classifier that combines classification functions: $f(x) = \text{sgn}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$.
 - Goal is to minimize error (on training set)
 - Iterative, select one h at each step
 - Calculate weights based on errors
 - Upweight missed classifications and select next h

[Adaboost on Wikipedia](#)

<http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>

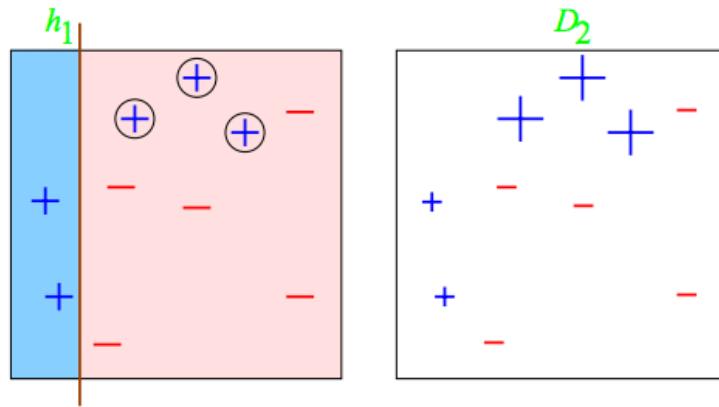
Simple example



<http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>

Round 1: adaboost

Round 1



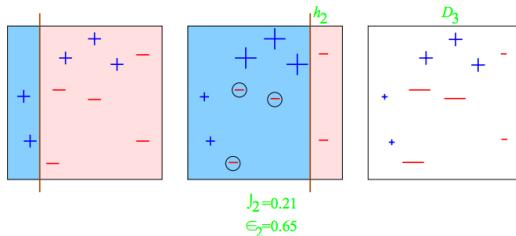
$$\epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

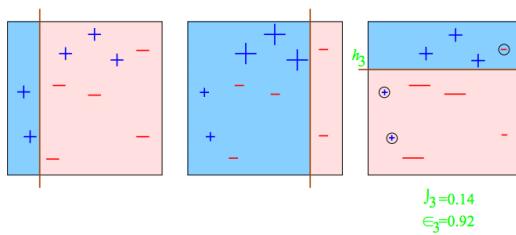
<http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>

Round 2 & 3

Round 2



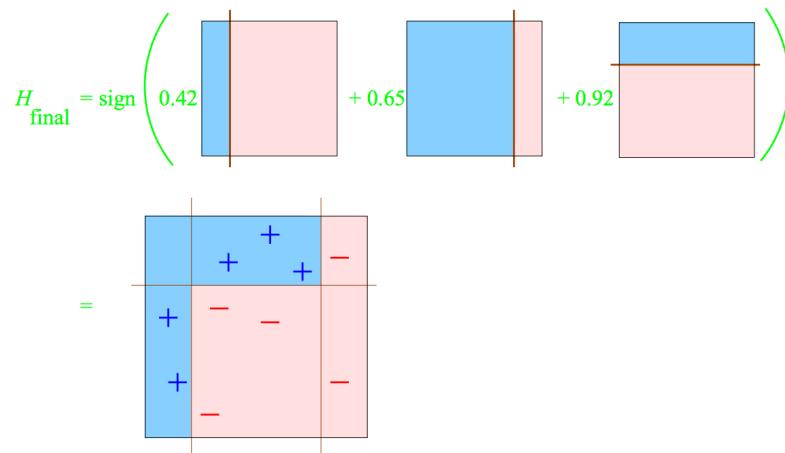
Round 3



<http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>

Completed classifier

Final Hypothesis



<http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>

Boosting in R

- Boosting can be used with any subset of classifiers
- One large subclass is [gradient boosting](#)
- R has multiple boosting libraries. Differences include the choice of basic classification functions and combination rules.
 - [gbm](#) - boosting with trees.
 - [mboost](#) - model based boosting
 - [ada](#) - statistical boosting based on [additive logistic regression](#)
 - [gamBoost](#) for boosting generalized additive models
- Most of these are available in the caret package

Wage example

```
library(ISLR); data(Wage); library(ggplot2); library(caret);
Wage <- subset(Wage, select=-c(logwage))
inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]
```

Fit the model

```
modFit <- train(wage ~ ., method="gbm", data=training, verbose=FALSE)  
print(modFit)
```

2102 samples

10 predictors

No pre-processing

Resampling: Bootstrap (25 reps)

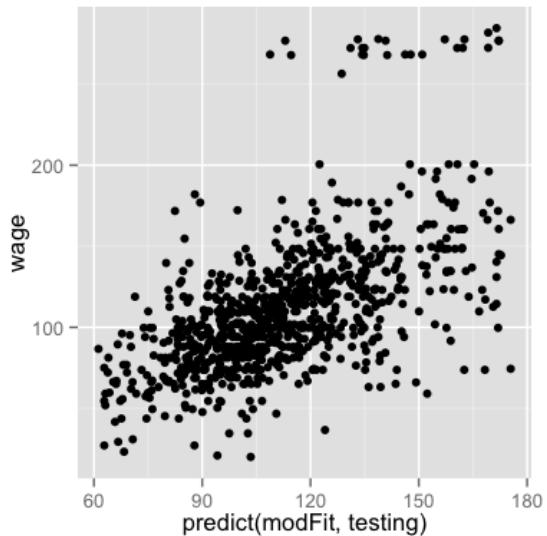
Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...

Resampling results across tuning parameters:

interaction.depth	n.trees	RMSE	Rsquared	RMSE	SD	Rsquared	SD
1	50	30	0.3	1	0.02		
1	100	30	0.3	1	0.02		
1	200	30	0.3	1	0.02		
2	50	30	0.3	1	0.02		
2	100	30	0.3	1	0.02		
2	200	30	0.3	1	0.02		

Plot the results

```
qplot(predict(modFit,testing),wage,data=testing)
```



Notes and further reading

- A couple of nice tutorials for boosting
 - Freund and Shapire - <http://www.cc.gatech.edu/~thad/6601-gradAI-fall2013/boosting.pdf>
 - Ron Meir- <http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf>
- Boosting, random forests, and model ensembling are the most common tools that win Kaggle and other prediction contests.
 - http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf
 - <https://kaggle2.blob.core.windows.net/wiki-files/327/09ccf652-8c1c-4a3d-b979-ce2369c985e4/Willem%20Mestrom%20-%20Milestone%201%20Description%20V2%202.pdf>



Model based prediction

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Basic idea

1. Assume the data follow a probabilistic model
2. Use Bayes' theorem to identify optimal classifiers

Pros:

- Can take advantage of structure of the data
- May be computationally convenient
- Are reasonably accurate on real problems

Cons:

- Make additional assumptions about the data
- When the model is incorrect you may get reduced accuracy

Model based approach

1. Our goal is to build parametric model for conditional distribution $P(Y = k|X = x)$
2. A typical approach is to apply [Bayes theorem](#):

$$Pr(Y = k|X = x) = \frac{Pr(X = x|Y = k)Pr(Y = k)}{\sum_{\ell=1}^K Pr(X = x|Y = \ell)Pr(Y = \ell)}$$

$$Pr(Y = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^K f_{\ell}(x)\pi_{\ell}}$$

3. Typically prior probabilities π_k are set in advance.

4. A common choice for $f_k(x) = \frac{1}{\sigma_k \sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{\sigma_k^2}}$, a Gaussian distribution

5. Estimate the parameters (μ_k, σ_k^2) from the data.
6. Classify to the class with the highest value of $P(Y = k|X = x)$

Classifying using the model

A range of models use this approach

- Linear discriminant analysis assumes $f_k(x)$ is multivariate Gaussian with same covariances
- Quadratic discriminant analysis assumes $f_k(x)$ is multivariate Gaussian with different covariances
- Model based prediction assumes more complicated versions for the covariance matrix
- Naive Bayes assumes independence between features for model building

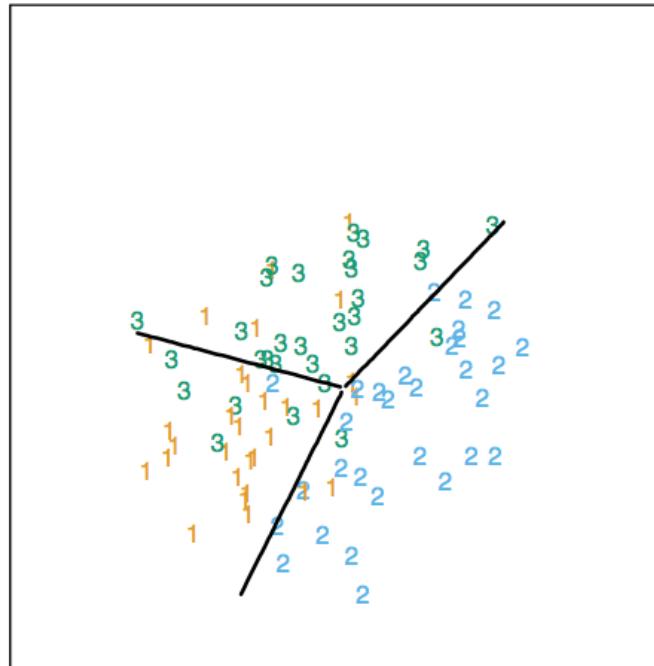
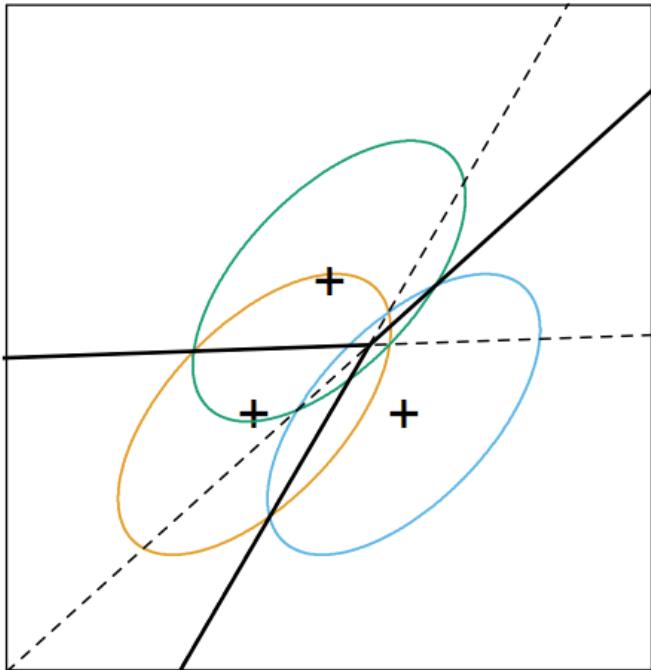
<http://statweb.stanford.edu/~tibs/ElemStatLearn/>

Why linear discriminant analysis?

$$\begin{aligned} & \log \frac{\Pr(Y = k | X = x)}{\Pr(Y = j | X = x)} \\ &= \log \frac{f_k(x)}{f_j(x)} + \log \frac{\pi_k}{\pi_j} \\ &= \log \frac{\pi_k}{\pi_j} - \frac{1}{2} (\mu_k + \mu_j)^T \Sigma^{-1} (\mu_k + \mu_j) \\ &\quad + x^T \Sigma^{-1} (\mu_k - \mu_j) \end{aligned}$$

<http://statweb.stanford.edu/~tibs/ElemStatLearn/>

Decision boundaries



Discriminant function

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\mu_k)$$

- Decide on class based on $\hat{Y}(x) = \operatorname{argmax}_k \delta_k(x)$
- We usually estimate parameters with maximum likelihood

Naive Bayes

Suppose we have many predictors, we would want to model: $P(Y = k|X_1, \dots, X_m)$

We could use Bayes Theorem to get:

$$P(Y = k|X_1, \dots, X_m) = \frac{\pi_k P(X_1, \dots, X_m|Y = k)}{\sum_{\ell=1}^K P(X_1, \dots, X_m|Y = k)\pi_\ell}$$
$$\propto \pi_k P(X_1, \dots, X_m|Y = k)$$

This can be written:

$$\begin{aligned} P(X_1, \dots, X_m, Y = k) &= \pi_k P(X_1|Y = k)P(X_2, \dots, X_m|X_1, Y = k) \\ &= \pi_k P(X_1|Y = k)P(X_2|X_1, Y = k)P(X_3, \dots, X_m|X_1, X_2, Y = k) \\ &= \pi_k P(X_1|Y = k)P(X_2|X_1, Y = k) \dots P(X_m|X_1 \dots, X_{m-1}, Y = k) \end{aligned}$$

We could make an assumption to write this:

$$\approx \pi_k P(X_1|Y = k)P(X_2|Y = k) \dots P(X_m|Y = k)$$

Example: Iris Data

```
data(iris); library(ggplot2)  
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"   "Petal.Length" "Petal.Width"   "Species"
```

```
table(iris$Species)
```

setosa	versicolor	virginica
50	50	50

Create training and test sets

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

```
[1] 45 5
```

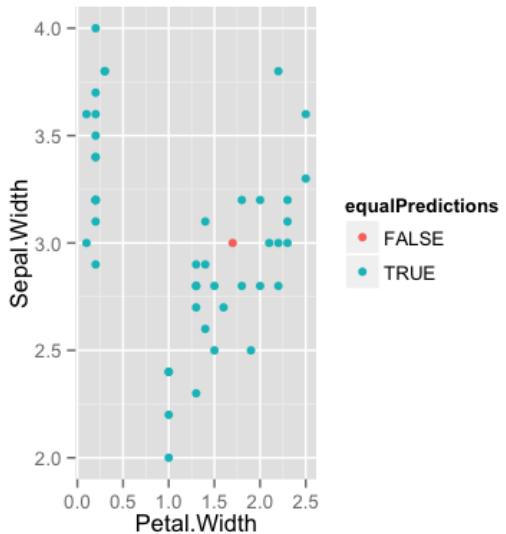
Build predictions

```
modlda = train(Species ~ ., data=training, method="lda")
modnb = train(Species ~ ., data=training, method="nb")
plda = predict(modlda, testing); pnb = predict(modnb, testing)
table(plda, pnb)
```

		pnb		
		setosa	versicolor	virginica
plda				
	setosa	15	0	0
	versicolor	0	13	1
	virginica	0	0	16

Comparison of results

```
equalPredictions = (plda==pnb)  
qplot(Petal.Width,Sepal.Width,colour=equalPredictions,data=testing)
```



Notes and further reading

- [Introduction to statistical learning](#)
- [Elements of Statistical Learning](#)
- [Model based clustering](#)
- [Linear Discriminant Analysis](#)
- [Quadratic Discriminant Analysis](#)



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:

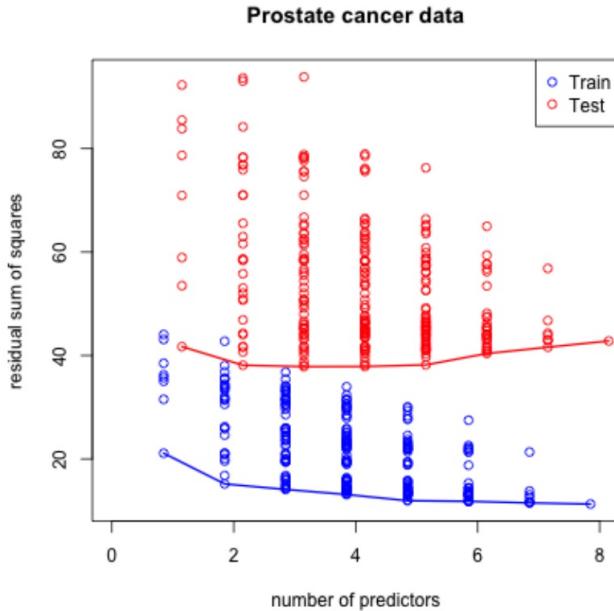
- You will get a good estimate of Y
- The estimate (of Y) will be biased
- We may reduce variance in the estimate

Prostate cancer

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

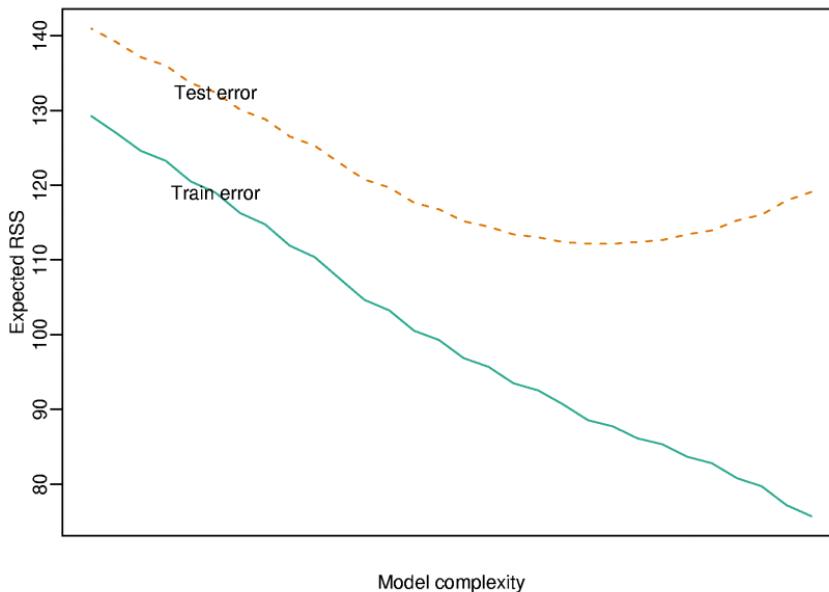
```
'data.frame': 97 obs. of 10 variables:
 $ lcavol : num -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num 2.77 3.32 2.69 3.28 3.43 ...
 $ age     : int 50 58 74 58 62 50 64 58 47 63 ...
 $ lbph    : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi     : int 0 0 0 0 0 0 0 0 0 0 ...
 $ lcp     : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: int 6 6 7 6 6 6 6 6 6 ...
 $ 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.cbcn.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}_λ is the estimate from the training data and look at a new data point $X = x^*$

$$\begin{aligned} E\left[\{Y - \hat{f}_\lambda(x^*)\}^2\right] &= \sigma^2 + \{E[\hat{f}_\lambda(x^*)] - f(x^*)\}^2 + \text{var}[\hat{f}_\lambda(x_0)] \\ &= \text{Irreducible error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

<http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/> <http://www.cbcn.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.cbcn.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 choosing λ figure out which $p - \lambda$ coefficients to make nonzero

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

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

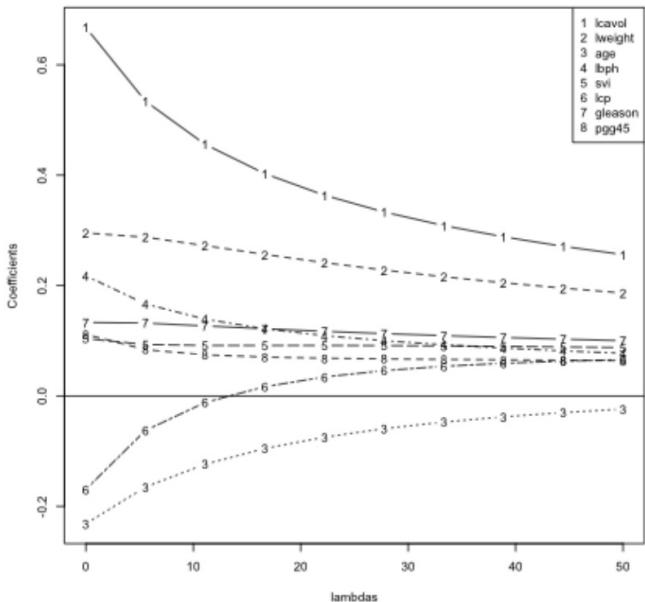
equivalent to solving

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

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.cbcn.umd.edu/~hcorrada/PracticalML/>

Ridge coefficient paths



<http://www.biostat.jhsph.edu/~ririzarr/Teaching/649/> <http://www.cbcn.umd.edu/~hcorrada/PracticalML/>

Tuning parameter λ

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

Lasso

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

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.cbcn.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

The screenshot shows a web browser displaying the Netflix Prize Leaderboard at www.netflixprize.com//leaderboard. The page has a red header with the Netflix logo and a yellow banner that says "COMPLETED". Below the banner, there's a navigation bar with links for Home, Rules, Leaderboard, and Update. The main section is titled "Leaderboard" in blue. It displays a table of top teams with their best test scores, improvements, and submit times. The table includes a header row and 10 data rows. A note at the bottom indicates the winning RMSE of 0.8567.

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

<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)
```

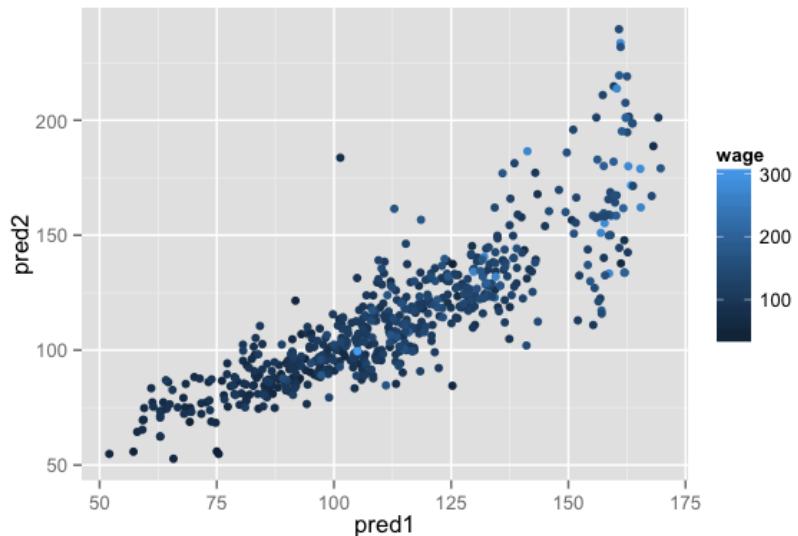
```
[1] 898 11
```

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 [ensemble learning](#)

Recall - scalability matters



Innovation

by Mike Masnick

Fri, Apr 13th 2012
12:07am

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:

5

<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

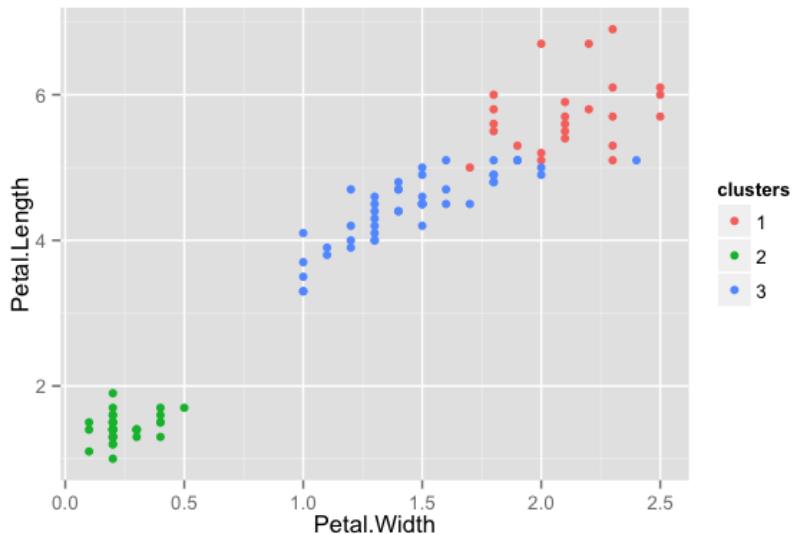
Iris example ignoring species labels

```
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

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)

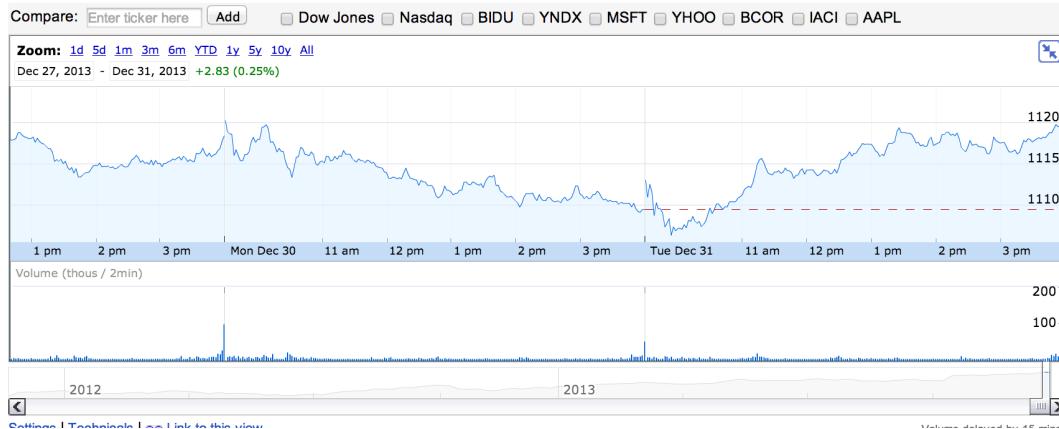
You have no recent quotes

1,120.71
+11.25 (1.01%)

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
Dec 31 - Close Mkt cap 374.42B Inst. own 72%
NASDAQ real-time data - Disclaimer P/E 32.19
Currency in USD

8+1
9.4k

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

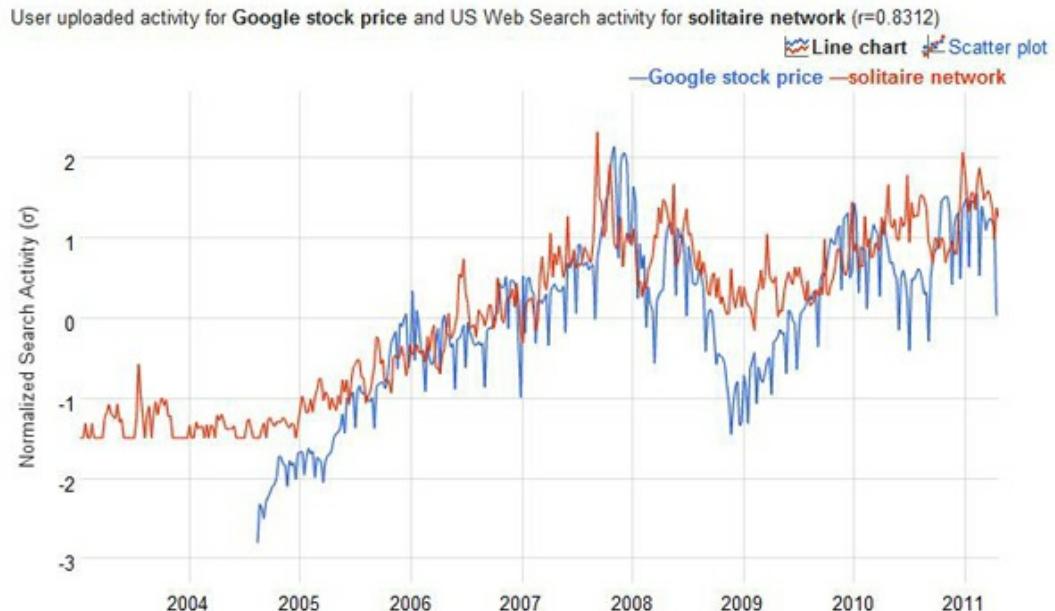


<https://www.google.com/finance>

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!)

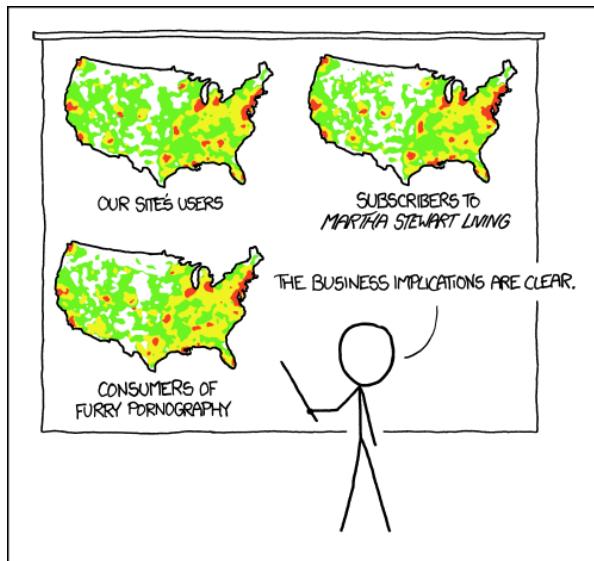
Beware spurious correlations!



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

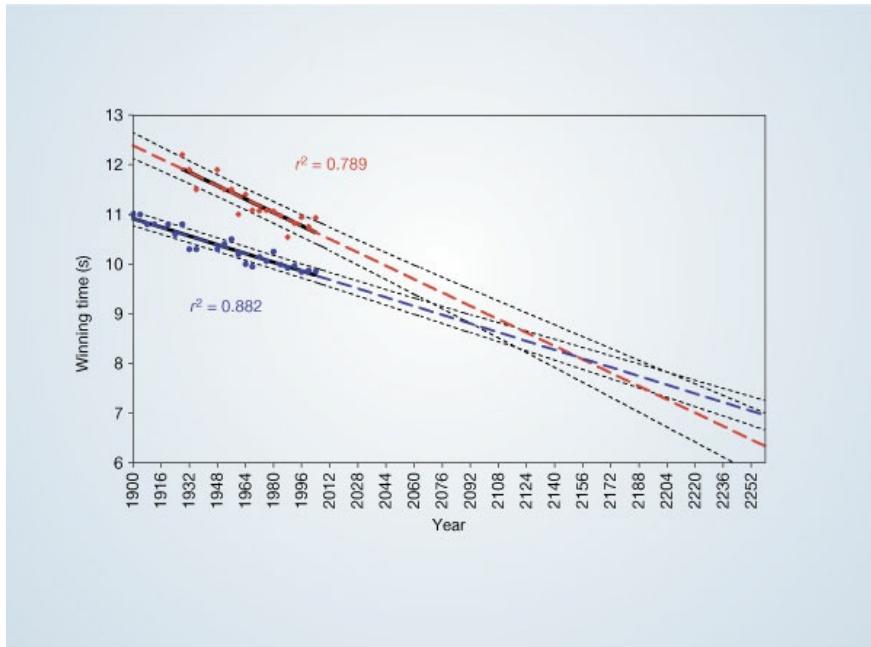
<http://www.newscientist.com/blogs/onepercent/2011/05/google-correlate-passes-our-we.html>

Also common in geographic analyses



<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")
```

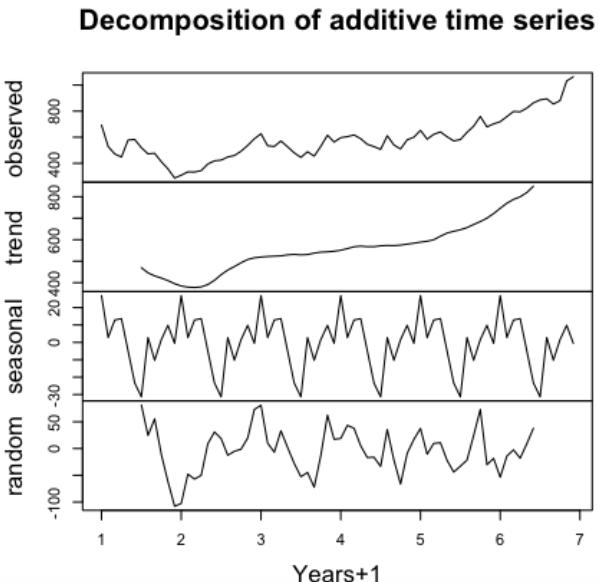
Example time series decomposition

- **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

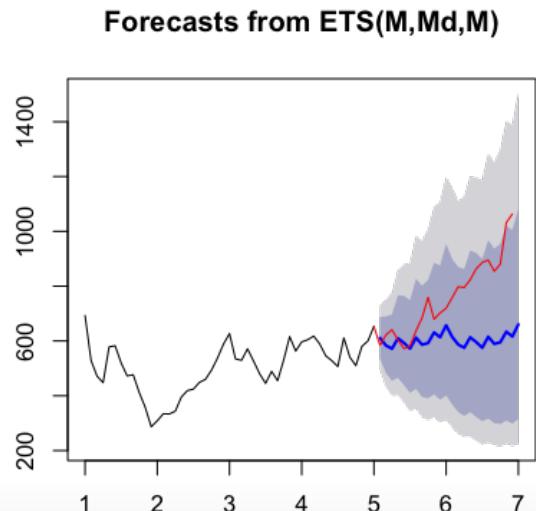
$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_{t-1}$$

	Seasonal Component		
Trend Component	N	A	M
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.