Bootstrapping for prediction

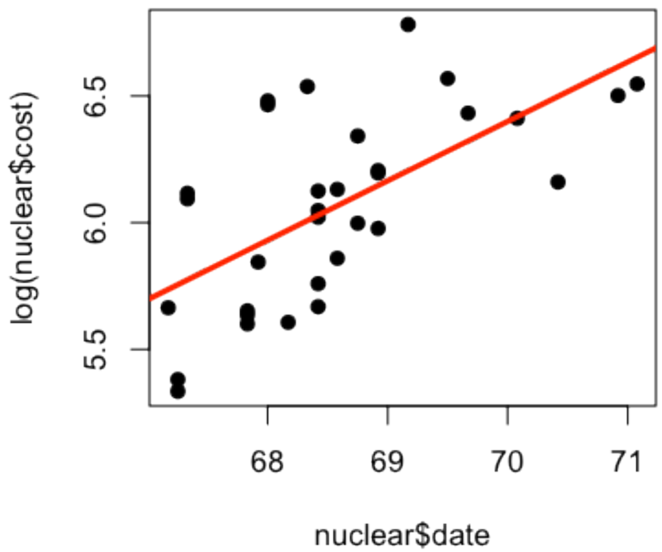
Jeffrey Leek, Assistant Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Key ideas

- Bootstrapping can be used for
  - Cross-validation type error rates
  - Prediction errors in regression models
  - Improving prediction

Bootstrapping prediction errors

```
library(boot); data(nuclear)
nuke.lm <- lm(log(cost) ~ date,data=nuclear)
plot(nuclear$date,log(nuclear$cost),pch=19)
abline(nuke.lm,col="red",lwd=3)
```



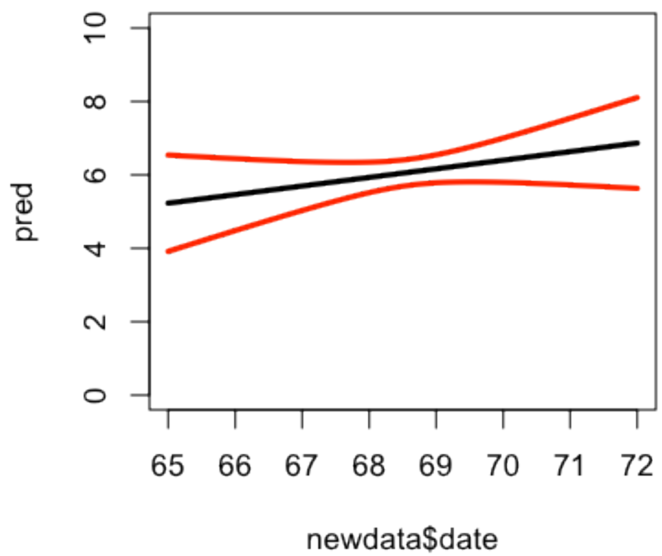
Bootstrapping prediction errors

```
newdata <- data.frame(date = seq(65,72,length=100))
nuclear <- cbind(nuclear,resid=rstudent(nuke.lm),fit=fitted(nuke.lm))
nuke.fun <- function(data,inds,newdata){
  lm.b <- lm(fit + resid[inds] ~ date,data=data)
  pred.b <- predict(lm.b,newdata)
  return(pred.b)
}
nuke.boot <- boot(nuclear,nuke.fun,R=1000,newdata=newdata)
head(nuke.boot$t)
```

```
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13] [,14] [,15]
[1,] 4.565 4.597 4.629 4.661 4.693 4.725 4.757 4.789 4.821 4.853 4.885 4.917 4.950 4.982 5.014
[2,] 6.453 6.450 6.447 6.444 6.441 6.438 6.435 6.432 6.429 6.426 6.423 6.420 6.417 6.414 6.411
[3,] 5.168 5.183 5.198 5.213 5.228 5.243 5.258 5.273 5.288 5.303 5.318 5.333 5.348 5.363 5.378
[4,] 5.401 5.413 5.425 5.437 5.449 5.461 5.473 5.485 5.497 5.509 5.521 5.533 5.545 5.557 5.569
[5,] 4.013 4.047 4.081 4.115 4.149 4.183 4.217 4.251 4.285 4.319 4.353 4.387 4.421 4.454 4.488
[6,] 6.263 6.261 6.259 6.258 6.256 6.254 6.252 6.250 6.248 6.246 6.245 6.243 6.241 6.239 6.237
     [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30]
[1,] 5.046 5.078 5.110 5.142 5.174 5.206 5.238 5.270 5.303 5.335 5.367 5.399 5.431 5.463 5.495
[2,] 6.408 6.405 6.402 6.399 6.396 6.393 6.390 6.387 6.384 6.381 6.377 6.374 6.371 6.368 6.365
```

Bootstrapping prediction errors

```
pred <- predict(nuke.lm,newdata)
predSds <- apply(nuke.boot$t,2,sd)
plot(newdata$date,pred,col="black",type="l",lwd=3,ylim=c(0,10))
lines(newdata$date,pred + 1.96*predSds,col="red",lwd=3)
lines(newdata$date,pred - 1.96*predSds,col="red",lwd=3)
```



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

Bagged loess

```
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
```

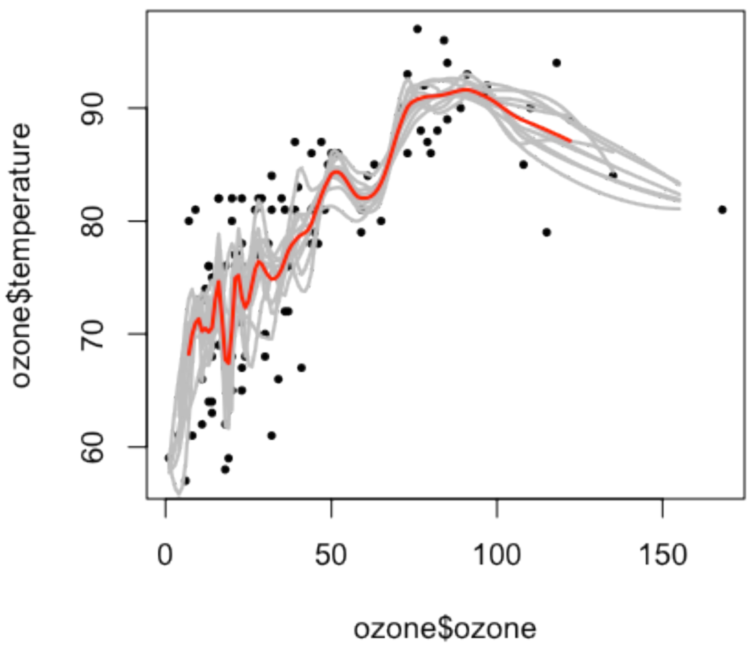
http://en.wikipedia.org/wiki/Bootstrap_aggregating

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)
```



Bagged trees

Basic idea:

1. Resample data
2. Recalculate tree
3. Average/mode) of predictors

Notes:

1. More stable
2. May not be as good as random forests

Iris data

```
data(iris)
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Bagging a tree

```
library(ipred)
bagTree <- bagging(Species ~.,data=iris,coob=TRUE)
print(bagTree)
```

```

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = Species ~ ., data = iris, coob = TRUE)

Out-of-bag estimate of misclassification error:  0.06
```

Looking at bagged tree one

```
bagTree$mtrees[[1]]$btree
```

```
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 150 98 virginica (0.33333 0.32000 0.34667)
   2) Petal.Length< 2.5 50  0 setosa (1.00000 0.00000 0.00000) *
   3) Petal.Length>=2.5 100 48 virginica (0.00000 0.48000 0.52000)
     6) Petal.Width< 1.75 52  5 versicolor (0.00000 0.90385 0.09615)
      12) Petal.Length< 4.9 46  2 versicolor (0.00000 0.95652 0.04348)
        24) Petal.Width< 1.65 44  0 versicolor (0.00000 1.00000 0.00000) *
        25) Petal.Width>=1.65 2  0 virginica (0.00000 0.00000 1.00000) *
      13) Petal.Length>=4.9 6  3 versicolor (0.00000 0.50000 0.50000)
        26) Sepal.Width>=2.65 3  0 versicolor (0.00000 1.00000 0.00000) *
        27) Sepal.Width< 2.65 3  0 virginica (0.00000 0.00000 1.00000) *
     7) Petal.Width>=1.75 48  1 virginica (0.00000 0.02083 0.97917)
      14) Petal.Length< 4.85 3  1 virginica (0.00000 0.33333 0.66667)
        28) Sepal.Length< 5.95 1  0 versicolor (0.00000 1.00000 0.00000) *
```

Looking at bagged tree two

```
bagTree$mtrees[[2]]$btree
```

```
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 150 98 versicolor (0.33333 0.34667 0.32000)
   2) Petal.Length< 2.6 50  0 setosa (1.00000 0.00000 0.00000) *
   3) Petal.Length>=2.6 100 48 versicolor (0.00000 0.52000 0.48000)
     6) Petal.Length< 4.85 51  3 versicolor (0.00000 0.94118 0.05882)
      12) Petal.Width< 1.65 45  0 versicolor (0.00000 1.00000 0.00000) *
      13) Petal.Width>=1.65 6  3 versicolor (0.00000 0.50000 0.50000)
        26) Sepal.Width>=3.1 3  0 versicolor (0.00000 1.00000 0.00000) *
        27) Sepal.Width< 3.1 3  0 virginica (0.00000 0.00000 1.00000) *
     7) Petal.Length>=4.85 49  4 virginica (0.00000 0.08163 0.91837)
      14) Petal.Width< 1.75 8  4 versicolor (0.00000 0.50000 0.50000)
        28) Petal.Length< 4.95 2  0 versicolor (0.00000 1.00000 0.00000) *
        29) Petal.Length>=4.95 6  2 virginica (0.00000 0.33333 0.66667)
          58) Petal.Width>=1.55 2  0 versicolor (0.00000 1.00000 0.00000) *
```

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

```
library(randomForest)
forestIris <- randomForest(Species~ Petal.Width + Petal.Length,data=iris,prox=TRUE)
forestIris
```

```

Call:
 randomForest(formula = Species ~ Petal.Width + Petal.Length,      data = iris, prox = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 3.33%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          2        48        0.04
```

Getting a single tree

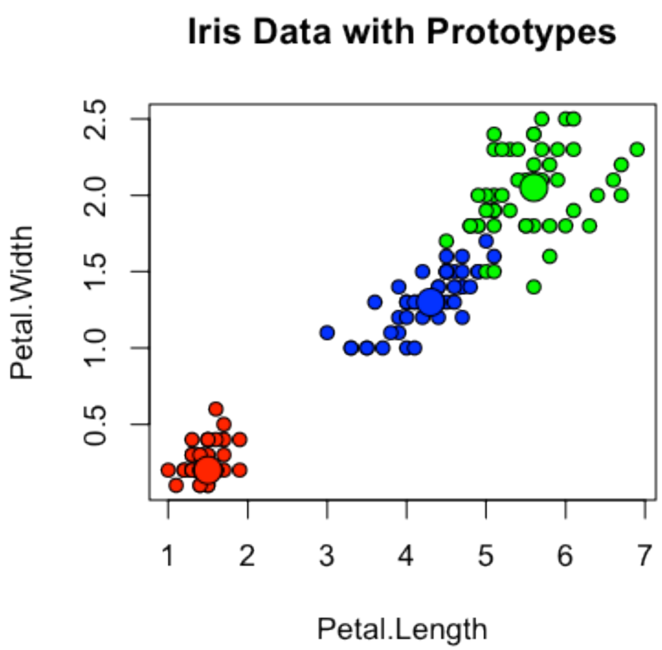
```
getTree(forestIris,k=2)
```

```
   left daughter right daughter split var split point status prediction
1              2              3         2        2.45      1          0
2              0              0         0        0.00     -1          1
3              4              5         1        1.70      1          0
4              6              7         1        1.55      1          0
5              0              0         0        0.00     -1          3
6              8              9         1        1.35      1          0
7              0              0         0        0.00     -1          2
8              0              0         0        0.00     -1          2
9             10             11         1        1.45      1          0
10            12             13         2        5.20      1          0
11             0              0         0        0.00     -1          2
12             0              0         0        0.00     -1          2
13             0              0         0        0.00     -1          3
```

17/22

Class "centers"

```
iris.p <- classCenter(iris[,c(3,4)], iris$Species, forestIris$prox)
plot(iris[,3], iris[,4], pch=21, xlab=names(iris)[3], ylab=names(iris)[4],
bg=c("red", "blue", "green")[as.numeric(factor(iris$Species))],
main="Iris Data with Prototypes")
points(iris.p[,1], iris.p[,2], pch=21, cex=2, bg=c("red", "blue", "green"))
```



18/22

Combining random forests

```
forestIris1 <- randomForest(Species~Petal.Width + Petal.Length,data=iris,prox=TRUE,ntree=50)
forestIris2 <- randomForest(Species~Petal.Width + Petal.Length,data=iris,prox=TRUE,ntree=50)
forestIris3 <- randomForest(Species~Petal.Width + Petal.Length,data=iris,prox=TRUE,nrtee=50)
combine(forestIris1,forestIris2,forestIris3)
```

```

Call:
 randomForest(formula = Species ~ Petal.Width + Petal.Length,      data = iris, prox = TRUE, ntree
               Type of random forest: classification
                     Number of trees: 600
No. of variables tried at each split: 1
```

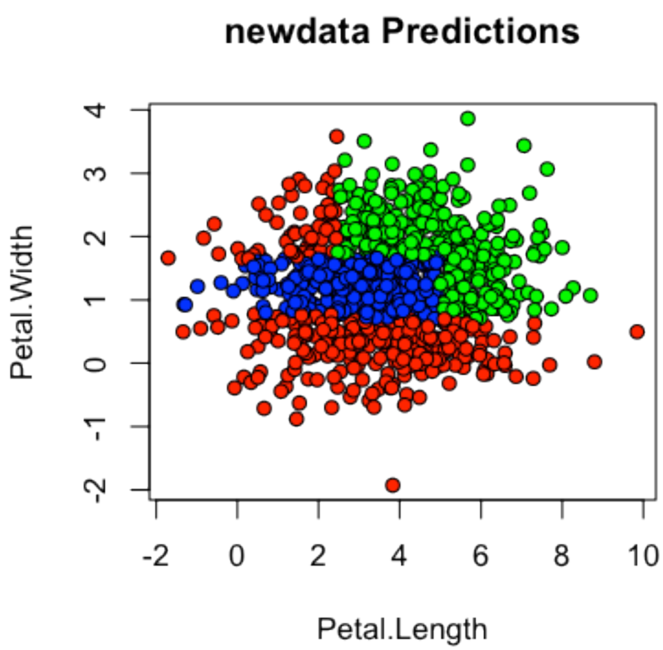
Predicting new values

```
newdata <- data.frame(Sepal.Length<- rnorm(1000,mean(iris$Sepal.Length),
                                           sd(iris$Sepal.Length)),
                      Sepal.Width <- rnorm(1000,mean(iris$Sepal.Width),
                                           sd(iris$Sepal.Width)),
                      Petal.Width <- rnorm(1000,mean(iris$Petal.Width),
                                           sd(iris$Petal.Width)),
                      Petal.Length <- rnorm(1000,mean(iris$Petal.Length),
                                            sd(iris$Petal.Length)))

pred <- predict(forestIris,newdata)
```

Predicting new values

```
plot(newdata[,4], newdata[,3], pch=21, xlab="Petal.Length",ylab="Petal.Width",
bg=c("red", "blue", "green")[as.numeric(pred)],main="newdata Predictions")
```



Notes and further resources

Notes:

- Bootstrapping is useful for nonlinear models
- Care should be taken to avoid overfitting (see rfcv funtion)
- Out of bag estimates are efficient estimates of test error

Further resources:

- Random forests
- Random forest Wikipedia
- Bagging
- Bagging and boosting