Bio Stats II : Lab 6

Gavin Fay

03/02/2016

Lab schedule

1/27: Introduction to R and R Studio, working with data
2/03: Intro to plotting, manipulating data
2/10: Probability, linear modeling
2/17: Programming practices, conditional statements
2/24: Creating functions, debugging
3/02: Permutation analysis
3/09: Advanced plotting

Recommended reading

Fox, J. 2002. Bootstrapping Regression Models. `http://statweb.stanford.edu/~tibs/sta305files/FoxOnBootingRegInR.pdf`

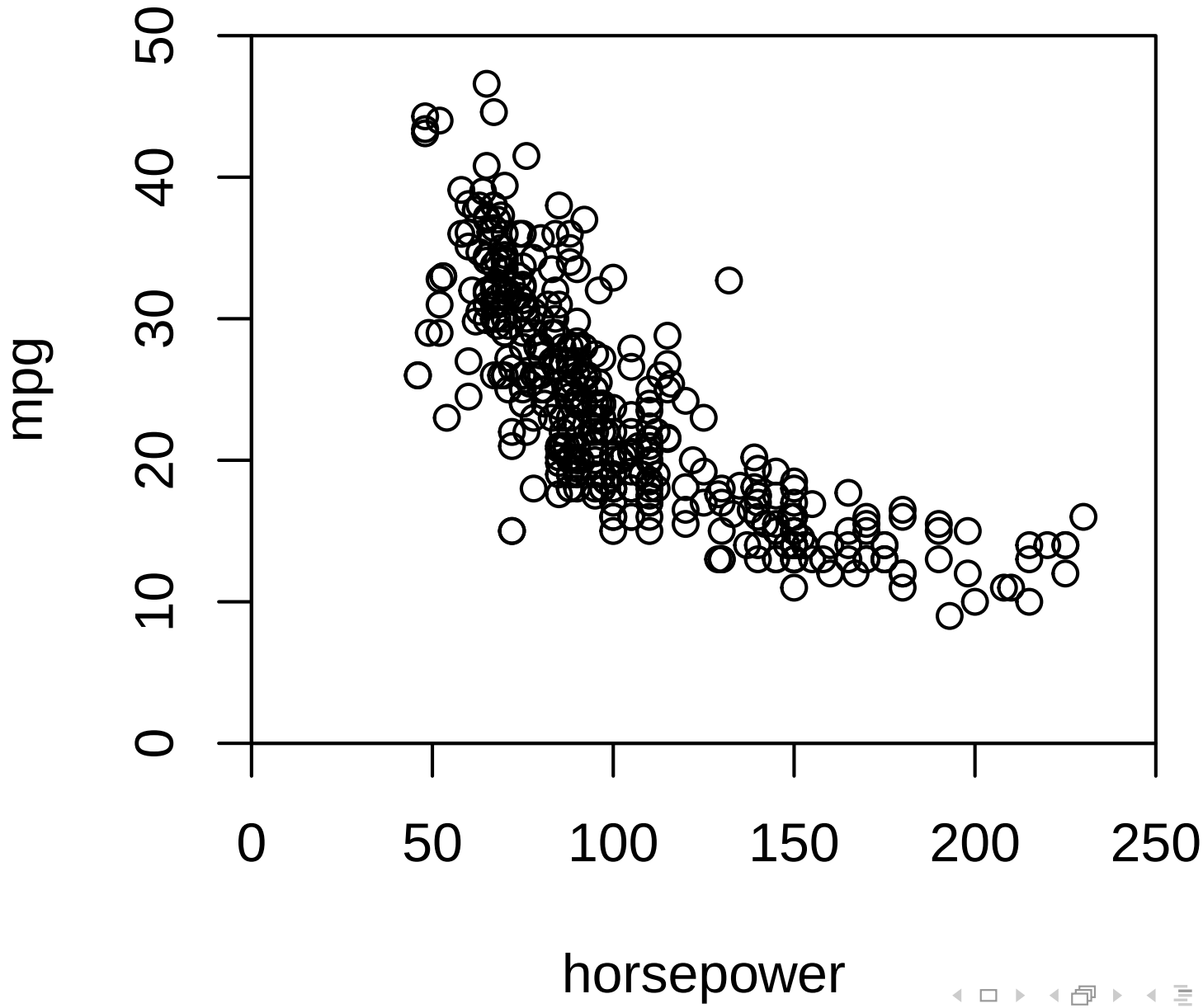
Efron, B. and Tibshirani, R. (1993) An Introduction to the Bootstrap. Chapman and Hall, New York, London.

James et al. Lab 5

Zieffler, A. S., Harring, J. R., & Long, J. D. (2011). Comparing Groups: Randomization and Bootstrap Methods Using R. Hoboken, NJ: Wiley

Auto dataset

in 'ISLR' package



Jackknifing

Recall: Have some data x_1, x_2, , . . . , x_n and we are computing a statistic.

Systematically leave out one observation at a time and recompute the statistic (e.g. mean).
mean$(x_2, \ldots, x_n)$; mean$(x_1, x_3, \ldots, x_n)$; mean$(x_1, x_2, x_4, \ldots, x_n)$

Variance estimation

$$\text{Var}_{(\text{jacknife})} = \frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_i - \hat{\theta}_{(.)})^2$$

Bias correction

$$\widehat{Bias}_\theta = n\hat{\theta}_{(.)} - (n-1)\bar{\theta}_{(jk)}$$

Custom Jackknife function in R

Jackknife estimates of slope from a linear model.

```
> slope <- coef(lm(mpg~horsepower,data=Auto))[2]
> lm.fn <- function(index,data)
+   return(coef(lm(mpg~horsepower,data=data,
+                  subset=index))[2])
> jacknife_index <- matrix(NA,nrow=nrow(Auto)-1,ncol=nrow(Auto))
> for (i in 1:nrow(Auto))
+   jacknife_index[,i] <- seq(1:nrow(Auto))[-i]
> jacknife_samples <- apply(jacknife_index,2,lm.fn,data=Auto)
> n <- nrow(Auto)
> bias_correct <- n*slope - (n-1)*mean(jacknife_samples)
> bias_correct
horsepower
-0.1574706
> slope
horsepower
-0.1578447
> var_jk <- ((n-1)/n)*sum((jacknife_samples-slope)^2)
> var_jk
[1] 5.602113e-05
> summary(lm(mpg~horsepower,data=Auto))$coefficients[2,2]^2
[1] 4.154448e-05
> quantile(jacknife_samples,c(0.025,0.975))
      2.5%      97.5%
```

Using `jackknife()` (in `bootstrap`)

Usage: `jackknife(x, theta, ...)`

`x` is vector containing the data.

`theta` is the function to be jackknifed.

e.g. mean of mpg

```
> library(bootstrap)
> jack <- jackknife(Auto$mpg,mean)
> jack$jack.bias
[1] 0
> jack$jack.se
[1] 0.3942124
```

slope of mpg~horsepower

```
> jack <- jackknife(1:nrow(Auto),lm.fn,data=Auto)
> jack$jack.bias
   horsepower
-0.0003741311
> jack$jack.se
[1] 0.007484702
> jack$jack.se^2
[1] 5.602077e-05
> quantile(jack$jack.values,c(0.025,0.975))
```

Lab exercise 1

Using the data in LaengelmavesiI2.csv, find the estimate for the coefficient of variation (CV) of lengths of pike.

Use jackknifing to bias correct this estimate, and estimate the sampling error for the CV.

Plot a histogram of the jackknifed estimates for the CV, and add vertical lines corresponding to the upper and lower limits of a central 95% confidence interval.

Bootstrapping

Recall:

- Have some data x_1, x_2, , . . . , x_n and we are computing a statistic.
- Randomly draw n values from the data *with replacement* (same value can be drawn multiple times).
- Calculate statistic from new random pseudo-data.
- Repeat a large number of times to obtain the distribution: $t_1, t_2, \ldots, t_n$.
- Resulting distribution is the bootstrap sampling distribution.
- Compute standard deviation and 95% confidence intervals from this. Finished.

Bootstrapping

Resampling methods in `R` make heavy use of `sample()`

Recall that `sample()` can be used to obtain samples from a given vector.

```
> # determine order of 6 players
> sample(1:6,replace=FALSE)
[1] 6 4 1 5 2 3
> # roll six dice
> sample(1:6,replace=TRUE)
[1] 1 6 6 3 1 5
> # pick six lottery numbers
> sample(1:49,6,replace=FALSE)
[1] 40 41 22 20  1 14
```

Performing generic case bootstrapping in R

Bootstrap estimates of slope from a linear model.

```
> slope <- coef(lm(mpg~horsepower,data=Auto))[2]
> #lm.fun returns an estimate of the slope
> #given subset of data provided by 'index'
> lm.fn <- function(index,data)
+   return(coef(lm(mpg~horsepower,data=data,
+                  subset=index))[2])
> #set up somewhere to store 1000 bootstrap samples
> boot_index <- matrix(NA,nrow=nrow(Auto),ncol=1000)
> # create the jackknife index vectors
> for (i in 1:1000)
+   boot_index[,i] <- sample(1:nrow(Auto),
+                            replace=TRUE)
> # Fit the model n times to the bootstrapped datasets
> boot <- apply(boot_index,2,lm.fn,data=Auto)
> slope
horsepower
-0.1578447
> mean(boot)
[1] -0.1585091
> summary(lm(mpg~horsepower,data=Auto))$coefficients[2,2]^2
[1] 4.154448e-05
> var(boot)
[1] 5.407456e-05
```

Case bootstrapping using 'bootstrap'

In `bootstrap` package.

Usage:
`bootstrap(x, theta, ...)`

`x` is vector containing the data.

`theta` is the function to be jackknifed.

e.g. mean of mpg

```
> boot <- bootstrap(Auto$mpg,1000,mean)
> mean(Auto$mpg)
[1] 23.44592
> sd(Auto$mpg)
[1] 7.805007
> mean(boot$thetastar)
[1] 23.43381
> sd(boot$thetastar)
[1] 0.3926011
```

Case bootstrapping using `boot()`

In package `boot()`

`boot()` is very powerful, can be used to do (almost) all kinds of bootstraps. However, typically requires you to still write functions.

Good description of `boot()` function in Fox (2002).

```
>   boot(data,  #data, can be vector or dataframe
+        statistic, # function of interest
+        R,   #number of bootstraps
+        sim = "ordinary",  #type of simulation
+        stype = c("i", "f", "w"),  #flag for 2nd arg
+        strata = rep(1,n),  #bootstrap within strata
+        weights = NULL,  #for importance sampling
+        ran.gen = function(d, p) d, #for parametric boots
+        mle = NULL,
+        ...,
+        parallel = c("no", "multicore", "snow"),
+        ncpus = getOption("boot.ncpus", 1L),
+        cl = NULL)
```

Implementing BCA bootstrapping using `boot()`

```
> lm.fn2 <- function(data,index)
+   return(coef(lm(mpg~horsepower,data=data,
+                  subset=index))[2])
> boot_slope <- boot(Auto,statistic=lm.fn2,R=1000)
> boot.ci(boot_slope, conf=0.95, type= "all")
Warning in boot.ci(boot_slope, conf = 0.95, type = "all"): bootstrap
variances needed for studentized intervals
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot_slope, conf = 0.95, type = "all")

Intervals :
Level      Normal              Basic
95%   (-0.1723, -0.1433 )   (-0.1722, -0.1429 )

Level     Percentile            BCa
95%   (-0.1728, -0.1435 )   (-0.1723, -0.1433 )
Calculations and Intervals on Original Scale
```

Residual bootstrap

```
> slope <- coef(lm(mpg~horsepower,data=Auto))[2]
> lm1 <- lm(mpg~horsepower,data=Auto)
> #resid.fun returns an estimate of the slope
> #given a resample of residuals from a model.
> resid.fn <- function(model,data) {
+   data$mpg <- fitted(model) + sample(resid(model),
+                                      replace=TRUE)
+   return(coef(lm(mpg~horsepower,data=data))[2])
+ }
> #set up somewhere to store 1000 bootstrap samples
> resid.boot <- rep(NA,1000)
> # create the jackknife index vectors
> for (i in 1:1000)
+   resid.boot[i] <- resid.fn(lm1,Auto)
> mean(resid.boot)
[1] -0.1577105
> var(resid.boot)
[1] 3.929659e-05
> quantile(resid.boot,c(0.025,0.975))
      2.5%      97.5%
-0.1705829 -0.1455244
```

Lab exercise 2

`hake.csv` contains abundance data for silver hake from the 2015 spring bottom trawl survey.

Write a function that produces bootstrapped estimates for the mean abundance per tow based on case resampling (5000 samples).
- use `sample()` rather than `bootstrap` or `boot`.

Compare the standard deviation of the bootstrapped estimates of the mean to the standard error of the mean from the original sample.

(*If you already did a bootstrap for the variance, compare the standard deviation of the bootstrapped variance estimates to the standard error of the variance*).

Calculate the estimtate of the bias in the estimate of the mean (*or variance*) using the bootstrap.

Compute an approximate 95% confidence interval for the mean based on the bootstrap, assuming normality. Compare this to the interval based on percentiles of the bootstrap sampling distribution.

Plot how the bootstrap confidence interval for the mean changes with the number of bootstrap samples (100, 500, 1000, 2000, 5000, 10000)

Validation Approach

Recall:
Split data into training set and a validation (or hold-out) set.

Fit model to training set, fitted model used to predict the responses for the observations in the validation set.

Resulting validation set error rate (e.g. MSE for quantitative response) is an estimate of the test error rate.

```
> set.seed(1)
> train <- sample(392,196)
> lm.fit <- lm(mpg ~ horsepower, data=Auto, subset=train)
> mean((Auto$mpg - predict(lm.fit, Auto))[-train]^2)
[1] 26.14142
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data=Auto, subset=train)
> mean((Auto$mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 19.82259
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data=Auto, subset=train)
> mean((Auto$mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 19.78252
```

Lab exercise 3

Consider the polynomial regression of wage on age (lecture 7). e.g.

```
> wage_lm.1 <- lm(wage~poly(age,4),data=Wage)
```

Define a unique random number seed.
Use the validation approach to estimate the test error rate for polynomials of order 2, 3, 4, 5, and 6.

Conduct the validation 20 times for each polynomial regression.
Plot the distribution (use boxplots) for the validation test error rate as a function of the degree of polynomial.

Based on the results, what order polynomial would you use?

Cross-Validation of GLM, LOOCV

Make use of `cv.glm` in `boot`.

```
> library(boot)
> glm.fit <- glm(mpg~horsepower,data=Auto)
> coef(glm.fit)
(Intercept)  horsepower
 39.9358610  -0.1578447
> cv.err <- cv.glm(Auto,glm.fit)
> cv.err$delta
[1] 24.23151 24.23114
> cv.error <- rep(NA,5)
> for (i in 1:5){
+   glm.fit <- glm(mpg~poly(horsepower,i),data=Auto)
+   cv.error[i] <- cv.glm(Auto,glm.fit)$delta[1]
+ }
> cv.error
[1] 24.23151 19.24821 19.33498 19.42443 19.03321
```

k-Fold Cross-Validation

Use `cv.glm()` with argument `K=k` to perform k-fold cross-validation.

```
> set.seed(17)
> cv.error.10 <- matrix(0,nrow=10,ncol=10)
> for (isim in 1:10){
+   for (ideg in 1:10) {
+     glm.fit <- glm(mpg~poly(horsepower,ideg),data=Auto)
+     cv.error.10[isim,ideg] <- cv.glm(Auto,
+                                      glm.fit,K=10)$delta[1]
+   }
+ }
> #cv.error.10
```

Setting K=n replicates LOOCV.

k-fold cross-validation using custom function

Consider the logistic regression model of *Solea* presence/absence as a function of salinity.

```
> Solea <- read.table("Solea.txt", header = T)
> solea_glm.1 <- glm(Solea_solea ~ salinity,
+                    data = Solea, family = binomial)
> sole_glm_fn <- function(train,validate) {
+  glm.1 <- glm(Solea_solea ~ salinity,data = train, family = binomial)
+  pred <- round(predict(glm.1,newdata=validate,type='response'))
+  return(length(which((pred == validate$Solea_solea)==FALSE))
+          /nrow(validate))
+ }
> k=nrow(Solea)
> groups <- sample(rep(1:k,length=nrow(Solea)),replace=FALSE)
> cv.err <- rep(0,k)
> for (i in 1:k)
+   {
+    pick <- which(groups==i)
+    cv.err[i] <- sole_glm_fn(Solea[-pick,],Solea[pick,])
+   }
> cv.err.k <- mean(cv.err)
> cv.err.k
[1] 0.2461538
> loo_cv <- cv.glm(Solea,solea_glm.1)$delta[1]
```

Lab exercise 4, Cross-validation of classification

Linear discriminant analysis, species classification of Laengelmavesi based on length, weight and height.

```
> library(MASS)
> Laengel <- na.omit(read.csv(file="Laengelmavesi2.csv",
+                     header=TRUE))
> lda1 <- lda(species~.,data=Laengel)
> pred_lda1 <- predict(lda1, data=Laengel)$class
> vote <- pred_lda1==Laengel$species
> length(which(vote==FALSE))/nrow(Laengel)
[1] 0.2229299
```

How does the classification rate vary among species?
Write a function to perform cross-validation for this analysis. Compute the LOOCV, and 10 estimates for the 5-fold CV.
Use LOOCV to determine the predictive ability of analyses based on only one of either weight, length, or height. Compare these CV results to those of the full model. For species identification purposes, which of these measurements would you prioritize collecting?

HINTS

Write down the steps you need to take to perform the calculations.
Make use of existing code from earlier in the lab to help.

Use as.formula() to create a formula object that can be used in a call to a model.

Lab exercise 5, Modeling catch rate of silver hake

BONUS

Write code that uses `boot()` to conduct residual bootstrapping for a negative binomial GLM of the 2015 silver hake catch rate data.
(e.g. `nb.1 <- glm.nb(hake~1,data=hakedata)`, use 5000 bootstraps)
Obtain bootstrap estimates for the variance of the mean catch rate and bias-corrected 95% confidence interval.

BONUS BONUS

Use the estimates of the parameters of the negative binomial in the above model to perform a parametric bootstrap.
Compare the results with those from the nonparametric and residual bootstrapping.

Permutation test

Is the class of students in BioStats 2 lab gender biased toward females?

There are 48 SMAST grad students, 23 are female.
(per directory on SMAST website, we know this is incorrect)
9 students in BioStats lab, 6 are female.

```
> p.gender <- function(niter, all.SMAST,class.size, obs.females) {
+    num.fem <- vector(length=niter)
+    for (i in 1:niter) {
+      X <- sample(x=all.SMAST, size=class.size,replace=FALSE)
+      num.fem[i] <- sum(X=="F")
+    }
+    p.value <- sum(num.fem >= obs.females)/niter
+    return(p.value)
+ }
> SMAST_grads <- c(rep("M", 25), rep("F", 23))
> p.gender(niter=10000, all.SMAST=SMAST_grads, class.size=9,
+          obs.females=6)
[1] 0.1939
> #'exact' probability
> sum(dhyper(6:9,23,25,9))
[1] 0.1900894
```