STAT S 670 Explolatory Data Analysis

Ganesh Nagarajan, gnagaraj@indiana.edu

November 20, 2015

1. Jackknife estimate
   Observation: Jackknifing is similar to Bootstrapping. Bootstrapping involves creating a population from the sample and sub-sampling from the population for the estimates. Jackkinfing is bootstrapping with replacement and everytime only one sample is removed.

The i_{th} estimate $\hat{\theta_{(i)}}$ of the statistic $\hat{\theta} = s(x)$ is $\hat{\theta_{(i)}} = s(x_i)$, $\forall i = 1, ...n$

$s(x_{(i)}) = \frac{1}{n-1} \sum_{j \neq i} x_j$
$= \frac{(n\bar{x}) - x_i}{n-1}$
$= \bar{x_i}$

Thus, $\bar{x_{(.)}} = \frac{1}{n} \sum_{i=1}^{n} \bar{x_i} = \bar{x}$

Thus the jack knife estimate, mean of the sample is the actual mean.

Standard error estimate for mean:
Therefore:
$\hat{se_{jack}}(\bar{x}) = \sqrt{(\sum_{i=1}^{n} \frac{x_i - \bar{x}^2}{(n-1)n})}$
$= \frac{\bar{\sigma}}{\sqrt{(n)}} = \frac{s}{\sqrt{(n)}}$

2.Standard error values,
From the known definitions,
$SDy_i = \frac{1}{\sqrt{n}} \sqrt{\sum_{i=1ton} (y_i - \bar{x})^(1/2)}$
Substituting and expanding,
$\frac{1}{(n-1)\sqrt{n}} \sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^(1/2)}$
$\frac{1}{n-1} SD(pv)$
so, SD(p.values)=(n-1)SD(y-1)
thus, $\frac{SD(p.values)}{\sqrt{n}} = \frac{(n-1)}{\sqrt{n}} SD(y-i) = SEC(PV)$
ie, the standard error of psuedo-values in jack knife is as same as the standard error value of "leave one out" method multiplies by (n-1)

3.LSAT vs GPA Dataset

```
x <- c(576,635,558,578,666,580,555,661,651,605,653,575,545,572,594)
y <- c(339,330,281,303,344,307,300,343,336,313,312,274,276,288,296)
data.df <- data.frame(x, y)

#Find Pearson correlation
corr.df <- cor(x, y, method = c("pearson"))
#------- 3 (a)
#Estimate theta user fischer's formula
theta <- 0.5*((1+corr.df)/(1-corr.df))
var.df<-1/(length(x)-3)
ci.l<-corr.df-1.96*var.df
ci.u<-corr.df+1.96*var.df

#-- Part B
```

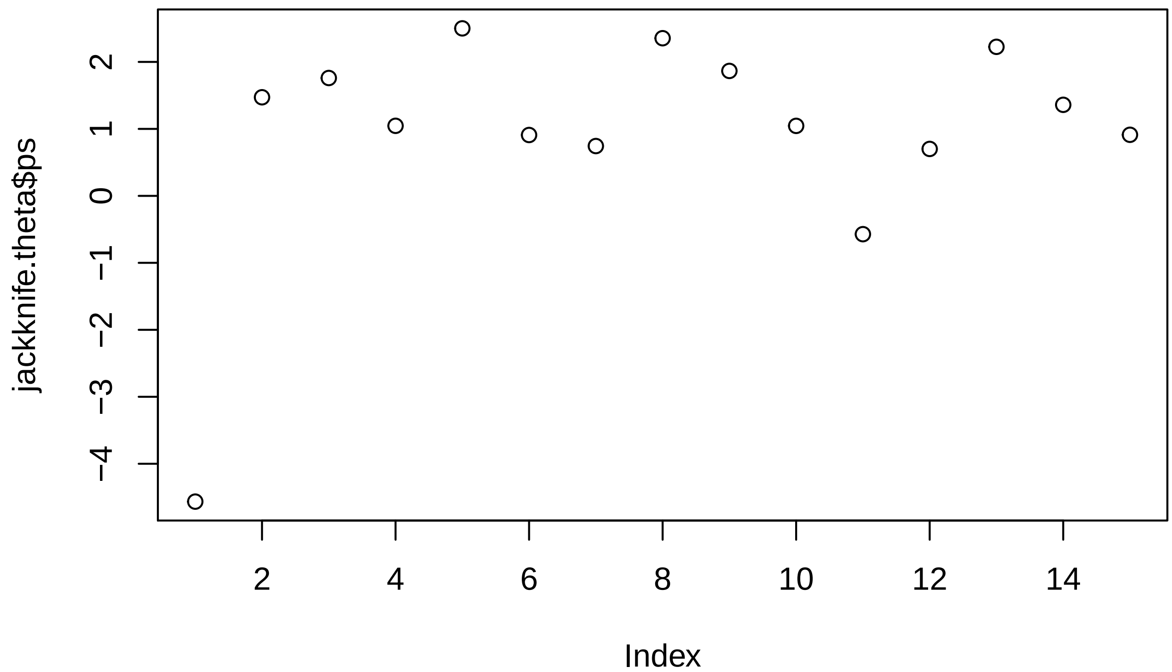
```r
jackknife.mean<-function(dat){
  n=nrow(dat)
  corr.df <- cor(dat$x, dat$y, method = c("pearson"))
  theta<-0.5*log((1+corr.df)/(1-corr.df))
  p.values <- numeric(n)
  est.theta <- numeric(n)
  for (i in 1:n){
    new.data<-dat[-i,]
    new.corr<-cor(new.data$x,new.data$y,method = c("pearson"))
    est.theta[i] <- 0.5*log((1+new.corr)/(1-new.corr))
    p.values[i] <- n*theta - (n-1)*est.theta[i]
  }
  bias = theta - mean(p.values)
  variance = var(p.values)
  se = sd(p.values)/sqrt(n)
  ci.lower = mean(p.values) - qt(0.975, df = (n-1))*se
  ci.upper = mean(p.values) + qt(0.975, df = (n-1))*se
  return(list(bias=bias, var = variance, ps = p.values,ci=c(ci.lower,ci.upper)))
}

#---Part(c)
jackknife.theta <- jackknife.mean(data.df)
print(jackknife.theta)
```

```
## $bias
## [1] 0.1191411
##
## $var
## [1] 2.912935
##
## $ps
##  [1] -4.5652958  1.4719784  1.7599688  1.0459610  2.5008589  0.9086096
##  [7]  0.7438389  2.3540039  1.8654888  1.0450339 -0.5719842  0.7006802
## [13]  2.2251007  1.3592711  0.9120457
##
## $ci
## [1] -0.02811946  1.86219412
```

```r
stem(jackknife.theta$ps)
```

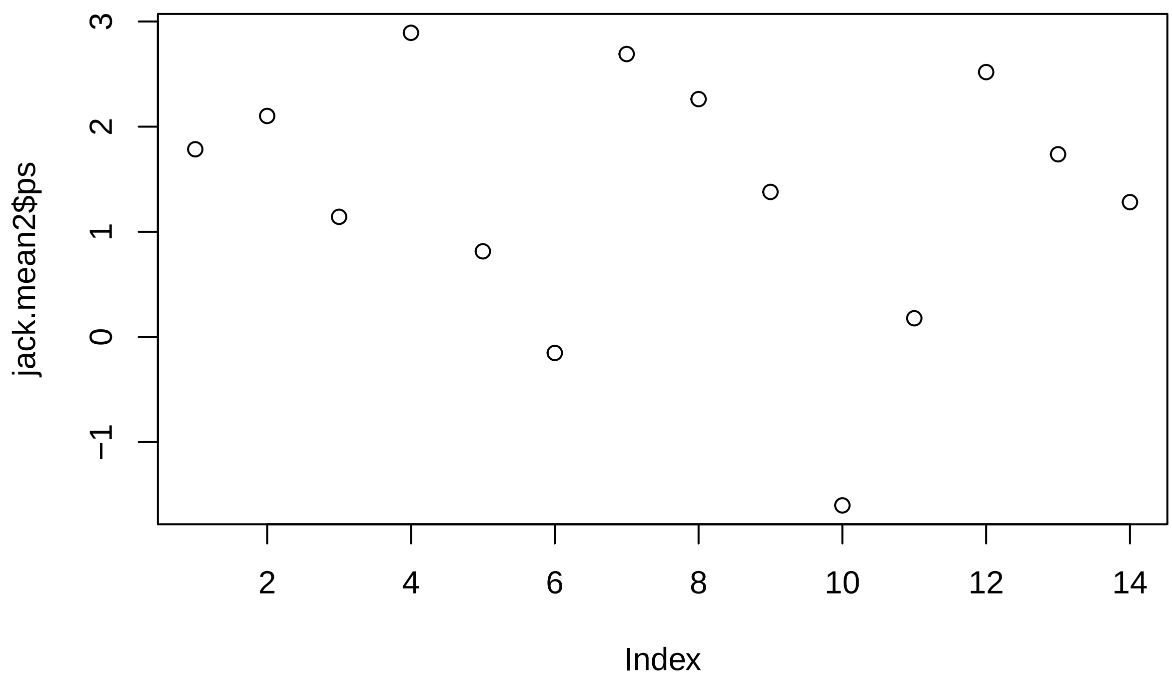
```
##
##   The decimal point is at the |
##
##   -4 | 6
##   -2 |
##   -0 | 6
##    0 | 7799004589
##    2 | 245
```

```r
plot(jackknife.theta$ps)
```



It can be seen that the first point is the outlier.

```
jack.df2 <- data.df[-1,]
jack.mean2 <- jackknife.mean(jack.df2)
plot(jack.mean2$ps)
```




```r
#----part(d)
bootstrap.est <- function (dat,n.sam)
{
  n <- length(dat$x)
  index <- 1:n
  rho <- cor(dat$x, dat$y, method = c("pearson"))
  theta <- 0.5*log((1 + rho)/(1 - rho))
  stat <- numeric(n.sam)
  oob.err <- numeric(n.sam)
  for (i in 1:n.sam)
  {
    sampleindex<- sample(index,n,replace<-TRUE)
    stat[i] <- cor(dat$x[sampleindex], dat$y[sampleindex],method=c("pearson"))
    oob.i <- setdiff(index,unique(sampleindex))
    oob.dat <- dat[oob.i,]
    oob.err[i] <- sum((oob.dat-stat[i])^2)/length(oob.i)
  }
  bias <- theta - mean(stat)
  variance <- var(stat)
  se <- sqrt(variance)
  avg.oob.err <- mean(oob.err)
  output <- list(bias=bias,var=variance,se=se,avg.oob.err=avg.oob.err)
}
boot.est<-bootstrap.est(data.df,10)
print(boot.est)
```

```
## $bias
## [1] 0.2485747
##
## $var
## [1] 0.05361252
##
## $se
## [1] 0.2315438
##
## $avg.oob.err
## [1] 457819.8
```

4. Linear Regression boot strap estimator

```r
source("rrline.r")

boot <- function(data, R)
{
  n <- length(data$x)
  index <- 1:n
  #Create a matrix with R rows
  theta <- matrix(0,R,2)
  oob.err <- numeric(R)
  boot.in <- numeric(R)
  fit <- run.rrline(data$x,data$y)
  theta.hat <- c(fit$a,fit$b)
  for(i in 1:R)
  {
    sample.index = sample(index, n, replace = TRUE)
    inbag <- data[sample.index,]
    oob.ind <- setdiff(index, unique(sample.index))
    oob.data <- data[oob.ind,]
    bootstrapped.y <- run.rrline(inbag$x, inbag$y)
    theta[i,1] <- bootstrapped.y$a
    theta[i,2] <- bootstrapped.y$b
    #print(theta)
    #print(oob.data$y)
    bootstrap.out <- oob.data$x*bootstrapped.y$b + bootstrapped.y$a
    #print(bootstrap.out)
    #print(oob.data$y)
    oob.err[i] <- sum((oob.data$y - bootstrap.out)^2)/length(oob.data)
  }
  mean.theta = apply(theta,2,mean)
  bias = theta.hat - mean.theta
  var = diag(cov(theta))
  se = sqrt(var)
  #print(oob.err)
  avgooberr = mean(oob.err)
  return(out = list(boot.a = mean.theta[[1]], boot.b = mean.theta[[2]], bias = bias, var = var, se=se, a
}
attach(faithful)
data <- data.frame(waiting,eruptions)
names(data) = c("x", "y")
boot.est = boot(data, 10)
```

```
##            a        b     |res|
## 1 -2.50234  0.08448 112.8501
## 2  0.50420 -0.00679 109.0280
## 3 -0.00958  0.00012 109.0477
## 4  0.00033  0.00000 109.0470
## 5 -0.00001  0.00000 109.0470
##   -2.00740  0.07780 109.0470
##            a        b     |res|
## 1 -2.01945  0.07794 106.8154
## 2  0.11499 -0.00142 106.5849
## 3  0.00735 -0.00009 106.5743
## 4  0.00047 -0.00001 106.5736
## 5  0.00003  0.00000 106.5736
##   -1.89661  0.07641 106.5736
##            a        b     |res|
## 1 -2.06512  0.07762 110.9929
## 2  0.05943 -0.00070 110.9460
## 3  0.00186 -0.00002 110.9446
## 4  0.00006  0.00000 110.9445
## 5  0.00000  0.00000 110.9445
##   -2.00377  0.07690 110.9445
##            a        b     |res|
## 1 -2.13997  0.07955 102.8902
## 2  0.50957 -0.00763 102.6311
## 3 -0.04621  0.00066 102.4002
## 4  0.00000  0.00000 102.4002
## 5  0.00000  0.00000 102.4002
##   -1.67661  0.07258 102.4002
##            a        b     |res|
## 1 -2.59966  0.08562 119.1951
## 2  0.68907 -0.01000 115.8751
## 3 -0.07057  0.00116 115.7681
## 4  0.01628 -0.00024 115.7659
## 5 -0.00168  0.00003 115.7661
##   -1.96656  0.07656 115.7661
##            a        b     |res|
## 1 -2.33448  0.08334 130.3850
## 2  0.04006 -0.00059 129.9840
## 3 -0.00771  0.00011 130.0612
## 4  0.00133 -0.00002 130.0479
## 5 -0.00023  0.00000 130.0502
##   -2.30103  0.08285 130.0502
##            a       b     |res|
## 1 -2.65614 0.08629 110.8304
## 2 -0.06530 0.00081 111.1787
## 3 -0.00466 0.00006 111.2036
## 4 -0.00033 0.00000 111.2053
## 5 -0.00002 0.00000 111.2054
##   -2.72646 0.08715 111.2054
##            a        b     |res|
## 1 -2.43128  0.08448 117.2417
## 2 -0.03436  0.00048 117.5954
## 3  0.00237 -0.00003 117.5708
## 4 -0.00016  0.00000 117.5725
```

```
## 5  0.00001  0.00000 117.5724
##   -2.46342  0.08494 117.5724
##          a        b     |res|
## 1 -1.93981  0.07687 110.5435
## 2  0.30078 -0.00425 109.5922
## 3  0.01843 -0.00029 109.6067
## 4 -0.00288  0.00004 109.6048
## 5  0.00036  0.00000 109.6051
##   -1.62311  0.07237 109.6051
##          a        b     |res|
## 1 -2.26535  0.08223 105.3204
## 2  0.31219 -0.00508 102.7439
## 3 -0.02352  0.00033 102.8237
## 4  0.00351 -0.00004 102.8131
## 5 -0.00047  0.00001 102.8145
##   -1.97363  0.07744 102.8145
##          a        b     |res|
## 1 -2.79375  0.08750 115.6700
## 2  0.03594 -0.00045 115.4075
## 3  0.00128 -0.00002 115.3981
## 4  0.00005  0.00000 115.3978
## 5  0.00000  0.00000 115.3978
##   -2.75648  0.08704 115.3978
```

```r
print(boot.est)
```

```
## $boot.a
## [1] -2.138769
##
## $boot.b
## [1] 0.07942551
##
## $bias
## [1]  0.131368971 -0.001625505
##
## $var
## [1] 1.635508e-01 3.154919e-05
##
## $se
## [1] 0.404414087 0.005616867
##
## $avgooberr
## [1] 12.16987
```

5.k-fold validation

```r
source("rrline.r")

cval.prog <- function (data,n.folds)
{
  n <- length(data$x)
  floorlen <- floor(n/n.folds)
  folds <- rep(1:n.folds,floorlen)
```

```
  k <- n - length(folds)

  if(k>0){
    #Create 1:k till k times
    folds = c(folds,1:k)
  }

  foldindicies = sample(folds,n,replace=FALSE)
  fit = run.rrline(data$x, data$y)
  theta = c(fit$a, fit$b)
  stat = numeric(n.folds)
  cverr = numeric(n.folds)
  cv.mat = matrix(0, n.folds, 2)
  for (i in 1:n.folds)
  {
    b = foldindicies == i
    cv = run.rrline(data[b,]$x, data[b,]$y)
    cv.mat[i,1] = cv$a
    cv.mat[i,2] = cv$b
    oobdata = data[!b,]
    pred = oobdata$x*cv$b + cv$a
    cverr[i] = sum((oobdata$y - pred)^2)/nrow(data[!b,])
  }
  mean.theta = apply(cv.mat, 2, mean)
  bias = theta - mean.theta
  var = diag(cov(cv.mat))
  se = sqrt(var)
  avgcverr = mean(cverr)
  return(out = list(boot.a = mean.theta[[1]], boot.b = mean.theta[[2]], bias = bias, var = var, se=se, a
}

attach(faithful)
```

```
## The following objects are masked from faithful (pos = 3):
##
##     eruptions, waiting
```

```
data.input = data.frame(waiting,eruptions)
names(data.input) = c("x", "y")
cval.res = cval.prog(data.input, 3)
```

```
##          a        b    |res|
## 1 -2.50234  0.08448 112.8501
## 2  0.50420 -0.00679 109.0280
## 3 -0.00958  0.00012 109.0477
## 4  0.00033  0.00000 109.0470
## 5 -0.00001  0.00000 109.0470
##   -2.00740  0.07780 109.0470
##          a       b    |res|
## 1 -2.50732 0.08449 32.89986
## 2 -0.00408 0.00005 32.90666
## 3 -0.00014 0.00000 32.90690
## 4 -0.00001 0.00000 32.90691
```

```
## 5  0.00000 0.00000 32.90691
##   -2.51155 0.08455 32.90691
##           a        b    |res|
## 1 -1.42924  0.06867 37.07355
## 2  0.19265 -0.00264 37.25148
## 3 -0.02006  0.00043 37.20709
## 4  0.00120 -0.00003 37.20974
## 5 -0.00007  0.00000 37.20958
##   -1.25553  0.06644 37.20958
##           a        b    |res|
## 1 -2.85560  0.09052 39.75831
## 2  0.13748 -0.00197 39.43783
## 3 -0.00414  0.00007 39.44889
## 4  0.00014  0.00000 39.44850
## 5  0.00000  0.00000 39.44852
##   -2.72212  0.08862 39.44852
```

```r
print(cval.res)
```

```
## $boot.a
## [1] -2.163062
##
## $boot.b
## [1] 0.07986628
##
## $bias
## [1]  0.155662098 -0.002066272
##
## $var
## [1] 0.6288030713 0.0001394087
##
## $se
## [1] 0.79297104 0.01180715
##
## $avgcverr
## [1] 0.2806631
```

```r
#It can be note that the program runs RR run program runs three times, which is as expected.
```