Bios301 Final Exam

2013-12-12

Please answer **any three** of the four questions below (if you submit answers to all four, only the first three will be graded). You may use any static reference materials you wish, but **no collaboration or consultation with any individual or service is permitted**. The instructor will answer any clarifying or technical questions you may have (email to chris.fonnesbeck@vanderbilt.edu or call/text to 615-955-0380). This exam is subject to the Vanderbilt University honor code.

The completed exam is due **Friday, Dec. 13 by Noon CST**. Your exam should be pushed to the private GitHub repository that you have been using to submit homework assignments. Please create a separate folder called `exam` to contain all exam materials. No content outside this directory will be considered when grading the exam. Content checked in after Noon on Dec. 13 will not be graded.

To obtain full marks, you must:

1. Submit working code to generate answers to all questions. Please test your code before submitting to ensure that it runs cleanly.
2. Completely annotate your code and your results in order to fully explain what you have done, and what the output means. It should be readable by a non-R user.
3. Submit a knitr file in either `.rnw` or `.rmd` format, along with its output (LaTeX or Markdown, respectively).

Question 1

```
# See pg 221 in book
```

25 points

a. Write a function `eval_poly` which will evaluate and return polynomials of the form:

$$P(x) = c_n x^{n\tilde{A}\acute{c}\hat{A}\acute{L}\hat{A}\check{S}1} + c_{n\tilde{A}\acute{c}\hat{A}\acute{L}\hat{A}\check{S}1} x^{n\tilde{A}\acute{c}\hat{A}\acute{L}\hat{A}\check{S}2} + \ldots + c_2 x + c_1$$

The arguments of the function should include x and the vector of polynomial coefficients $c = [c_1, \ldots, c_n]$.

b. For moderate to large values of n, evaluation of a polynomial at x can be done more efficiently using Horner's Rule:

1. Set $a_n \leftarrow c_n$.
2. For $i = n$ÃćÂĹÂŠ$1, \ldots, 1$ set $a_i = a_{i+1}x + c_i$.
3. Return a_1. (This is the computed value of $P(x)$).

Write an R function with arguments x and a vector of polynomial coefficients and which returns the value of the polynomial evaluated at x. Ensure that your function returns an appropriate vector of values when x is a vector.

c. Do some timings to compare the algorithms used in the previous two questions. In particular, try the following code:

```
system.time(eval_poly(x=seq(-10, 10, length=500000), c(1,-2,2,3,4,6,7)))
```

and compare it to a similar call to the function in part (b).

d. What happens to the comparison when the number of polynomial coefficients is smaller? Try the polynomial:

$$P(x) = 2x^2 + 17x$$ÃćÂĹÂŠ3

Question 2

25 points

Estimate the coverage probability of 95% bootstrap percentile intervals for the binomial probability `p` in the following model:

$$x \sim \text{Binomial}(n = 20, p = 0.3)$$

```
mean <- NULL
upper.bound <- NULL
lower.bound <- NULL
contains.mean <- NULL
# for loop creates multiple bootstraps of a binom(50,0.3) and the
# corresponding confidence intervals
for (i in 1:1000) {
```

```r
    my.sample <- rbinom(n = 50, size = 20, prob = 0.3)
    y <- sample(my.sample, replace = TRUE)
    mean[i] <- mean(y)
    upper.bound[i] <- mean[i] + 1.96 * sd(y)/sqrt(20)
    lower.bound[i] <- mean[i] - 1.96 * sd(y)/sqrt(20)
    # Want CI to contain true mean, 6:
    if (lower.bound[i] < 6 && upper.bound[i] > 6)
        contains.mean[i] <- 1 else contains.mean[i] <- 0
}
# creating a data frame of mean, upper and lower bounds, and the indicator
# variable contains.mean
Data <- data.frame(mean, lower.bound, upper.bound, contains.mean)
colnames(Data) <- c("Sample Mean", "Lower Bound", "Upper Bound", "Contains Mean")
mean_contains.mean <- mean(contains.mean)
head(Data)
```

```
##   Sample Mean Lower Bound Upper Bound Contains Mean
## 1        5.34       4.257       6.423             1
## 2        5.86       4.927       6.793             1
## 3        6.36       5.262       7.458             1
## 4        5.78       4.982       6.578             1
## 5        5.72       4.714       6.726             1
## 6        6.66       5.911       7.409             1
```

```r
cat(mean(contains.mean) * 100, "percent of CI's contained the mean")
```

```
## 97.5 percent of CI's contained the mean
```

Is this interval conservative, unbiased, or liberal?

```r
# Based on the results above, the rate of coverage is greater than 95% (or
# 100*(1-0.05)) and therefore, the interval is conservative. If the rate was
# less than 95%, it would be liberal.
```

Question 3

25 points

The central limit theorem claims that if X is a Poisson random variable with parameter ÃŐÂż, and

$$Z = \frac{X - \lambda}{\sqrt{\lambda}}$$

then Z is approximately standard normal (*i.e.* $N(0, 1)$), and the approximation improves as ÃŐÂż gets large.

a. Write code to simulate a large number of Z values for values of ÃŐÂż in the set
$1, 3, 5, \ldots, 99$. Execute the code and describe how the distribution of Z changes as ÃŐÂż increases.

```
lambda <- seq(1, 29, by = 2)
x <- matrix(0, 100, length(lambda))
z <- matrix(0, 100, length(lambda))
# Creating the x matrix:
for (i in 1:length(lambda)) {
    x[, i] = rpois(100, lambda[i])
}
# Creating the z matrix:
for (i in 1:length(lambda)) {
    z[, i] = (x[, i] - lambda[i])/sqrt(lambda[i])
}
mean(z)

## [1] -0.03931

sd(z)

## [1] 0.9966

# As lambda increases, the distribution of z becomes centered about zero
# with a std deviation of 1.
```

b. Write a function in ggplot2 that generates a series of QQ-plots, which plots the *calculated* values of Z against their *theoretical* values under a true standard normal, along with a diagonal that represents perfect correspondence between the actual and theoretical values. Use this code to generate QQ-plots for the results in part (a). An individual plot should similar to this:

You will probably need to plot a few multi-plot panels so that your graphs are readable.

Hint: to get theoretical quantiles corresponding to Z, use the following code:

```
qnorm(ppoints(length(z)))[order(order(z))]
```

```
# The given function qnorm will be x axis actual z vals are y axis I am
# getting multiple errors and am running out of time; however, I would use
# something similar to the code below
library(ggplot2)
g <- ggplot(data, aes(x = line, y = x[, 1])) + geom_point() + xlab("theoretical quantiles")
    ylab("actual quantiles") + ggtitle(qqplot) + geom_abline(line, color = "red")

## Error:  ggplot2 doesn't know how to deal with data of class
function

g

## Error:  object 'g' not found
```

c. Based on your plots, about how large must ÃŐÂż be before the approximation becomes reasonable? Here "reasonable" means that it is appropriate to make normality assumptions with the sample.

Question 4

25 points

In order to explore how home run hitting in baseball has changed throughout history, let's create a data visualization using the `baseball-databank` database.

Use the data in the `Batting` table to create a plot of home run rate by team across years. We will use home run rate to account for differing numbers of at-bats (AB) among players and teams:

$$HRRate = HR/AB$$

So, this quantity expresses the probability of hitting a home run in any given at bat.

Your solution must include the following:

a. Connect to the `baseball-databank` database using `RSQLite` in order to query from the `Batting` table.

```
setwd("~/Bios301/baseball-databank")
library(RSQLite)

## Loading required package:  DBI

drv <- dbDriver("SQLite")
con <- dbConnect(drv, dbname = "baseball-archive-2011.sqlite")
# Listing tables:
dbListTables(con)

##  [1] "AllstarFull"         "Appearances"         "AwardsManagers"
##  [4] "AwardsPlayers"       "AwardsShareManagers" "AwardsSharePlayers"
##  [7] "Batting"             "BattingPost"         "Fielding"
## [10] "FieldingOF"          "FieldingPost"        "HallOfFame"
## [13] "Managers"            "ManagersHalf"        "Master"
## [16] "Pitching"            "PitchingPost"        "Salaries"
## [19] "Schools"             "SchoolsPlayers"      "SeriesPost"
## [22] "Teams"               "TeamsFranchises"     "TeamsHalf"

# Listing fields in Batting table
dbListFields(con, "Batting")

##  [1] "playerID"  "yearID"    "stint"     "teamID"    "lgID"
##  [6] "G"         "G_batting" "AB"        "R"         "H"
## [11] "2B"        "3B"        "HR"        "RBI"       "SB"
## [16] "CS"        "BB"        "SO"        "IBB"       "HBP"
## [21] "SH"        "SF"        "GIDP"      "G_old"
```

b. Calculate **team** home run rate (not individual players) using `ddply` (*remember to deal with missing values!*).

```
library(plyr)
# Saving data in a data frame battingdata
battingdata <- dbGetQuery(con, "Select teamID, lgID, HR, AB from Batting;")
HRRate <- ddply(battingdata, .(teamID), summarise, HRRate = mean(HR, na.rm = TRUE)/mean
    na.rm = TRUE))
head(HRRate)

##   teamID   HRRate
## 1    ALT 0.002225
## 2    ANA 0.029573
## 3    ARI 0.031285
## 4    ATL 0.028054
## 5    BAL 0.027930
## 6    BFN 0.004404
```

c. Import the `leagues.csv` table from the `databases` folder and merge it with your dataset to get the proper league names corresponding to `lgID` (league ID – though there are only 2 today, there have been 7 leagues in the history of baseball).

```
data <- read.csv("../Datasets/leagues.csv")
# this is not working for some reason....
batting.merged <- merge(data, battingdata, by = "lgID", all = TRUE)
```

d. Create a single scatter plot using `ggplot2` of HR rate plotted against year, and color code the points according to league.

```
library(ggplot2)
new.data <- merge(HRRate, data, by = "lgID")

## Error:  'by' must specify a uniquely valid column

qplot(yearID, HRRate, color = lgID, data = new.data)

## Error:  object 'new.data' not found
```