Statistics 506 Midterm - Solutions

Name, uniquename:

October 24, 2017

Instructions

Answer each of the questions that follow in the space provided. If you need additonal space, you may use a piece of plain white paper with the problem number and your name clearly labelled at the top.

No notes or resources are allowed during the exam. A help page for regular expressions will be displayed.

The exam has eight questions worth 20-30 points each and totalling 200 points. You have 90 minutes to complete the exam. Try not to spend more than 10 minutes on any single question before beginning the others.

When finished, please bring your exam to the front of the room.

Questions

1. For each of the regular expressions in the first row of the table below, indicate which strings it matches among the rows names by writing 'T' (for "TRUE") when there is a match. Leave the remaining cells blank. [25 points]

string ↓ regex →	`t{2,}`	`([a-z])\1`	`^A.*a$`	`[aeiou]$`	`(^[rt])\1`
Alita			T	T	
Correy		T			
Desirre		T		T	
Farris		T			
Jarrard		T			
Mariette	T	T		T	
Omarr		T			
Parris		T			
Qinntette	T	T			
Tacarra		T		T	

---

2. For each snippet of R code below, give the value of `z` after the code is run. You may assume each code starts with an empty global environment and that only base packages are loaded. [30 pts total; 5 pts each]

a.

```
x = 1:10
dim(x) = c(5, 2)
z = apply(x, 2, sum)
z
```

```
## [1] 15 40
```

b.

```
  x = 8
  f = function(y){
    y %% 2
  }
  z = f(x)
  z
```

```
## [1] 0
```

c.

```
  x = 1:10
  y = lapply(x, function(n) {1:n}^2)
  z = y[[4]]
  z
```

```
## [1]  1  4  9 16
```

d.

```
  x = rep(1:3, each=3)
  y = rep(1:3, 3)
  dim(x) = dim(y) = c(3, 3)
  z = t(x) %*% y
  z = z[,3]
  z
```

```
## [1]  6 12 18
```

e.

```
  z = 10
  f = function(z){
    z = 2*median(z)
    z
  }
  y = f(z)
  z
```

```
## [1] 10
```

f.

```
 z = 1
 while(z %/% 4 <= 5){
   if(z %% 2 == 0){
     z = z + 1
   } else {
     z = 2*z
   }
 }
z
```

```
## [1] 30
```

---

3. [25 pts] For normally distributed data $X_i \sim_{iid} N(0, \sigma^2)$ there are two commonly used estimates of the variance σ^2:

   - the unbiased sample variance: $\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$

- and the maximum likelihood estimate: $\tilde{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$.

In this question you will write simulation code for comparing the performance of these two estimators.

a. Write a function `f` that takes a numeric vector `x` and returns a length two vector containing the variance estimates $\hat{\sigma}^2$ and $\tilde{\sigma}^2$. [5 pts]

```
f = function(x) {
  n = length(x)
  sum({x-mean(x)}^2)/c('p_hat'=n-1, 'p_tilde'=n)
}
```

b. Write a second function, `g`, that accepts an `n` by `m` array and returns an `m` by `2` array with estimates of $\hat{\sigma}^2$ and $\tilde{\sigma}^2$ for each column of the input array. Your code should use vectorization and avoid unnecessary loops. [5 pts]

```
g = function(x){
  n = dim(x)[1]
  ss = rowSums({t(x) - colMeans(x)}^2)
  cbind('p_hat' = ss/{n-1}, 'p_tilde'=ss/n)
}
```

c. The mean-squared error (MSE) of the estimator $\hat{\sigma}^2$ as an estimate of σ^2 is the expectation $E[(\hat{\sigma}^2 - \sigma^2)^2]$. Likewise, the MSE of $\tilde{\sigma}^2$ is $E[(\tilde{\sigma}^2 - \sigma^2)^2]$. The MSE can be decomposed into terms for the bias and variance as follows:

- bias $= E[\hat{\sigma}^2] - \sigma^2$
- variance $= E[(\hat{\sigma}^2 - E[\hat{\sigma}^2])]$
- mse $=$ bias^2 + variance

Complete the code skeleton below to compare $\hat{\sigma}^2$ $\tilde{\sigma}^2$ in terms of the MSE, bias, and variance using simulation for various values of n and σ^2. Your code should not add any additional `for` or `apply` loops and should make use of the function `g` you wrote in part b. [15 pts]

```
mcrep=1e3
sigsq_values = seq(.2, 2, 20)
n_values = seq(10, 30, 5)

for(sigsq in sigsq_values){
  for(n in n_values){

     ## insert your code here
     # generate data
     x = rnorm(mcrep*n, 0, sqrt(sigsq))
     dim(x) = c(n, mcrep)
     ests = g(x)

     # compare performance
     avg = colMeans(ests)
     bias = avg - sigsq
     varn = rowSums({t(ests) - avg}^2)
     mse  = bias^2 + varn

     # output results
     rbind(bias, varn, mse)
  }
}
```

4. Assume we have a data frame `Loblolly` loaded in R with three columns `height`, `Seed`, and `age`, recording, respectively, the height in feet of 14 loblolly trees (seeds 1-14) at 3, 5, 10, 15, 20, and 25 years. For each of the pipes below: (i) determine the dimensions of the resulting data frame and, (ii) concisely explain what each pipe is computing, and (iii) provide a descriptive replacement for column names denoted by `XX0` or `XX1`. You should assume that the `dplyr` package has been loaded. Your explanations should each be a single sentence. [30 pts total; 10 pts each]

a.

```
out =
  Loblolly %>%
  group_by(age) %>%
  summarize(XX0 = sum(height) / n(),
            XX1 = sqrt(sum({height - XX0}^2) / {n()-1})
  )
dim(out)
```

```
## [1] 6 3
```

```
## XX0 = mean_height, XX1=sd_height
```

b.

```
out =
  Loblolly %>%
  group_by(Seed) %>%
  mutate(height = height / height[1]) %>%
  filter(age==25) %>%
  ungroup %>%
  summarize(XX0 = median(height))
dim(out)
```

```
## [1] 1 1
```

```
## XX0 = median_relative_growth
```

c.

```
out =
  Loblolly %>%
  filter(age %% 5 == 0) %>%
  group_by(Seed) %>%
  mutate(
    XX0 = c(0,diff(height)) / c(1, diff(age))
  ) %>%
  ungroup %>%
  group_by(age) %>%
  summarize(XX1 = mean(XX0))
dim(out)
```

```
## [1] 5 2
```

```
## XX1 = avg_growth_rates
```

---

5. Consider the Loblolly data from the previous problem. Convert the long-form Loblolly data from question 4 into a data frame with one row per tree (i.e. value of `Seed`) and columns for the heights in each year. Rename these columns `yr3` ... `yr25`. Standardize each column of heights using z-scores (you can do this before or after reshaping).

a. Write an R expression using `dplyr` and `tidyr` to accomplish the task above. Assume the Loblolly data are already present in a data frame named `Loblolly` [10 pts].

```
Loblolly %>%
  group_by(age) %>%
  mutate(height={height-mean(height)}/sd(height)) %>%
  rename(yr=age) %>%
  spread(yr, height, sep='')
```

b. Write Stata commands to accomplish the task. Assume the data set has already been loaded into Stata in the long format described in question 4. [10 pts]

```
rename height yr
reshape wide yr, i(Seed) j(age)

foreach var of varlist yr3-yr25 {
  quietly summarize `var'
  replace `var' = (`var' - r(mean)) / r(sd)
}
```

---

6. Consider an R data frame `population` with three columns - 'country', 'year', and 'population' - giving the populations of various countries from 1995 to 2013 as reported by the World Health Organization. Write R code to accomplish each of the tasks below. Be as concise as possible; each of the tasks can be accomplished in a single expression using pipes. [25 pts]

a. Count the number of times each country appears in the data and store this in a data frame `n_years`. [5 pts]

```
n_years = population %>% group_by(country) %>% summarize(n=n())
```

b. Use `n_years` from part a to create a reduced data set that includes `pop_complete` only countries with all 19 years from 1995 to 2013 observed. [5 pts]

```
pop_complete =
  population %>%
  filter(country %in% {n_years %>% filter(n==19)}$country
  )
```

c. Using the reduced data set from part (b), for each country compute the relative population growth from 1995 to 2000, from 2000 to 2005, and from 2005 to 2010. Store the results in an object `rel_growth`. [10 pts]

```
rel_growth =
  pop_complete %>%
  filter(year %in% c(1995, 2000, 2005, 2010)) %>%
  group_by(country) %>%
  arrange(year)   %>%
  rename(p = population) %>%
  summarize(r2000 = p[2]/p[1],
            r2005 = p[3]/p[2],
            r2010 = p[4]/p[3])
```

d. Use the results of part (c) to find all countries whose population declined during all three periods. [5 pts]

```
rel_growth %>% filter(r2000<1 & r2005<1 & r2010<1)
```

---

7. In this problem you will write a function for generating bootstrap samples of a numeric vector `x`. You will return the samples as an object of class `bootstrap` and write S3 methods for this class. [25 pts total]

   a. Define a `bootstrap` function that accepts: a numeric vector `x`, a function `f`, and an integer `n` indicating the number of bootstrap samples to draw. Set a default value of `n=1e3`. Your function should return an object of class `bootstrap` with two elements: 1) `obs` with the value of `f(x)`, 2) `boot_stats` a vector of length `n` containing `f` the values returned by `f` on each bootstrap sample. Your `bootstrap` function should also check that `f(x)` is a length one numeric vector and produce an error if not. [15 pts]

```
bootstrap = function(x, f, nboot=1e3){
  obs = f(x)
  if(!is.numeric(obs) || length(obs)!=1){
    stop('f(x) is not a length one numeric vector!')
  }

  out = list(obs=obs,
             boot_stats =
               sapply(1:nboot, function(i) f(sample(x,length(x),replace=TRUE))),
             call=match.call() ## This was not required, but is good practice.
  )
  class(out) = 'bootstrap'
  out
}
```

   b. Write an S3 `print` method for objects of class bootstrap that displays on a single line the observed statistic and number of bootstrap samples. Be sure to label the values displayed. [5 pts]

```
print.bootstrap = function(obj){
  cat(capture.output(obj$call),'\n')
  cat(sprintf('f(x) = %4.3f; nboot = %g\n', obj$obs, length(obj$boot_stats)))
}
```

   c. Write an S3 `summary` method for objects of class bootstrap that displays the observed statistic, a confidence interval constructed using the percentile method, and the associated confidence level on a single line. The confidence level should be configurable using an argument `level` and have a default of .95. [5 pts]

```
summary.bootstrap = function(obj, level=.95){
  cat(capture.output(obj$call),'\n')

  # get quantiles for associated confidence level
  p = {1-level}/2
  p = c(p, 1-p)
  q = quantile(obj$boot_stats, p)

  cat(sprintf('f(x) = %.3f (%.3f, %.3f); level = %4.2f; nboot = %g\n',
              obj$obs, q[1], q[2], level, length(obj$boot_stats)
              )
      )
}
```

8. Below is a short uncommented Stata program. Write a single sentence concisely describing what the program appears to be doing. Then, translate the Stata program into R. [20 pts]

```
import delimited mtcars.csv
keep mpg cyl wt
```

```
  generate  mpg_wt = mpg*wt

  collapse (count) n=mpg (sum) mpg_wt (mean) mpg wt \\\
    (sd) mpg_sd=mpg  wt_sd=wt, by cyl

  generate r_hat = (mpg_wt - n*mpg*wt) / ((n-1)*mpg_sd*wt_sd)

  keep cyl r_hat
```

Solution: The code is computing the Pearson correlation between `wt` and `mpg` within eachy `cyl` group.

```r
# a concise translation
mtcars %>%
  select(mpg, wt, cyl, am) %>%
  group_by(cyl) %>%
  summarize(r_hat = cor(mpg, wt))

# or a more direct translation
mtcars %>%
  select(mpg, wt, cyl, am) %>%
  group_by(cyl) %>%
  summarize(n = n(),
            mpg_wt = sum(mpg*wt),
            mpg_m = mean(mpg), wt_m = mean(wt),
            mpg_sd = sd(mpg), wt_sd = sd(wt)
  ) %>%
  transmute(cyl=cyl,
            r_hat = {mpg_wt - n*mpg_m*wt_m} / {{n-1}*mpg_sd*wt_sd})
```