Midterm Exam

Stats 506, Fall 2019

October 17, 2019

Instructions

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

You may bring as a reference a single, double sided, 8.5 by 11 inch page of notes. Please write your name and unique name on this sheet and submit it with your exam.

Your desk should be clear of all other materials except pens or pencils. No computer or cell phone use is permitted during the exam.

The exam has six questions organized into two parts. The four questions in part one are worth 110 points collectively and you should anser them all. There are two questions in part two worth 40 points each. You will *choose one* to answer. There are 150 total points available.

You have 80 minutes to complete the exam. Try not to spend too much time on any single question before beginning the others.

When finished, please bring your answer sheet and solution to part two to the front of the room. You may keep this exam.

Part One

Part one of the exam has four question, worth a total of 110 points. You should answer all four questions on the answer sheet provided.

Question 1 - R and dplyr basics [30 points]

In this question, you will be presented with short chunks of R code and asked to determine the value of the object `result` *in the global environment* after evaluating each. You should assume each chunk is run in a clean R session and that the chunks are independent. Write your answer in the space provided on the answer sheet. [5 points each]

a.

```r
x = 1:10
result = max( x %% 3L )
```

b.

```r
x = 2 * sum(1:100)
y = x > 5e3
result = typeof( c(x, y) )
```

c.

```r
result = 10
f = function(input) {
 result = sqrt(input)
 return( result )
}
input = f(result * result)
output = f(input)
```

d.

```r
start = 1
goal = 5^2
total = 0
while ( total <= goal ) {
 for ( i in 1:start ) {
    total = total + i
 }
 start = start + 1
}
result = total
```

e.

```r
x = 1:30
dim(x) = c(10, 3)
x = x %/% c(1, 30)
result = any( colSums(x) %% 2 == 0 )
```

f.

```r
library(tidyverse)
n = 100
df = tibble( a = rnorm(n) )
df = df %>%
  mutate( b = rbinom(n(), size = 1, prob = pnorm(a) ) ) %>%
  group_by(b) %>%
  summarize( n = n(), a = mean(a) )
result = unname( as.numeric( lapply(df, length) ) )
```

### Question 2 - Regular Expressions [20 points]

a.

This regexp matches syntactically valid names for R objects:

```
`^([A-Za-z]|[.][._A-Za-z])[._A-Za-z0-9]*$`
```

Consider the function below, then answer the question that follows.

```r
is_syn = function(x) {
  grepl('^([A-Za-z]|[.][._A-Za-z])[._A-Za-z0-9]*$', x)
}
```

For each of the following, write the result on your answer sheet in the space provided [1 pt each].

i. `is_syn('.2way')`
ii. `is_syn('.twoway')`
iii. `is_syn('2way')`
iv. `is_syn('._2way')`

b.

Chye is using an online tool to practice with regular expressions. An exercise in the current lesson asks him to write a regular expression which matches the first three words below, but does not match the fourth. Which of the following are valid solutions? On your answer sheet, circle “valid” if a solution is valid and “invalid” if not. For each invalid solution, make a minor correction to make it valid [3 pts each].

Task	Text
Match	cat.
Match	896.
Match	?=+.
Skip	abc1

i. `...\.`
ii. `(.){3}[.]`
iii. `[^.]$`
iv. `^[^1]`

c.

You have been asked to update a collection of related R scripts you wrote a few years ago. The scripts all use dplyr and tidyr and there are several thousand lines of code. Before making the requested updates, you decide it would be a good idea to replace instances of `spread` with `pivot_wider` and instances of `gather` with `pivot_longer`. You decide to start by using `grep` at the command line to create a checklist of files/lines to change. After your first attempt, you notice you are getting a lot of lines where you used the words `gather` or `spread` in the comments.

Task: Write a call to `grep` that will find all lines in .R or .Rmd files in the local directory that use the *functions* `gather` or `spread`. [4 pts]

### Question 3 - Linux Shell Skills [30 points]

a. Match each of the shell commands on the right to its best description on the left. Each description is used at most once. Write your answers in the space provided on the answer sheet. [2pts each]

```
i. echo                 A. writes the contents of a file to stdout
ii. #!/bin/bash         B. changes the working directory
iii. <, >               C. prints a string to stdout
iv. cat                 D. extracts the first n lines from a file or stdin
v. ls                   E. lists the files in a specified directory
vi. cd                  F. the "pipe" which redirects stdout to stdin
vii. head -n            G. a "shebang" specifying the shell the script is written for
viii. tail -n           H. redirects stdout to a specificed file
ix. |                   I. extracts the last n lines from a file or stdin
```

b. Consider the shell script below. Within the script there are 6 missing “commands” or similar, indicated with angled brackets, e.g. .
Replace each missing “command” with a “command” from part a, adding options and/or parameters as appropriate based on context from the script. Some commands are missing in more than one place. [2 pts each]

Background. A popular format for working with data too large to store in memory is the *bucketed*, *columnar* format. A columnar data storage format is one in which each column or variable of a rectangular data array is stored in its own file on disk. A bucketed or chunked data storage format is one in which the rows are split into buckets (aka chunks or groups) with rows from the same bucket being stored in different files on disk. A rectangular array can be stored in a *bucketed*, *columnar* format by organizing buckets into directories with each file within a directory representing a column of the rectangular array.

```
<command1>
## This script checks if the buckets (folders) in a
## directory all contain exactly the same variables (files)
## with a given extension.
##
## usage:
##  ./bash check_buckets.sh path ext
##
## Updated: October 16, 2019
## Author: James Henderson

# path to bucketed data and extension for variables
path=$1
ext=$2

# change directory to parent directory of buckets (i.e. path)
<command2> $path
```

(*The script continues on the next page.*)

```
# Create a list of buckets (directories)
buckets=$(ls -d */)

# Extract names of first and last bucket
b1=$(echo $buckets | tr ' ' '\n' | <command3> )
b_last=$(echo $buckets | tr ' ' '\n' | <command4>)

# Create a list of all files with extension $ext in $b1
cd $b1
files1=$(<command5>)
## handle the case when files1 is empty.
if [ -z "$files1" ]; then
 echo "No variables with extension $ext found folder $b1."
 exit
fi

# Create a list of all files with extension $ext in $b_last
cd ../$b_last
files_last=$(<command5>)
if [ -z "$files_last" ]; then
 echo "No variables with extension $ext found in folder $b_last."
 exit
fi
cd ..

# Loop over buckets and check if variables match
# the first (or last) bucket.
for bucket in $buckets; do
  # Check vars in first bucket against vars in last bucket.
  if [ $bucket == $b1 ]; then
    # Loop over files in final bucket
    for file in $files_last; do
      if [ ! -f $b1/$file ]; then
          echo "File $file not found in bucket $b1." <command6> /dev/stderr
          exit
      fi
    done
    echo "All vars bucket $b_last found in $b1."
    continue
  fi

  # Check vars in current bucket against first bucket
  for file in $files1; do
      # Check if variable file exists
      if [ ! -f $bucket/$file ]; then
      echo "File $file not found in bucket $bucket." <command6>  /dev/stderr
      exit
      fi
  done
  echo "All vars in bucket $b1 found in bucket $bucket."
done
```

Question 4 - dplyr and tidyr [30 points]

In this question you will write or interpret short dplyr pipes that explore or analyze the `Orange` data, which contains 35 rows and 3 columns recording the growth of orange trees. The dataset has three columns:

- `Tree`: an ordered factor, identifying individual trees,
- `age`: a numeric vector giving the number of days since the tree was planted,
- `circumference`: a numeric vector recording the circumference of the trunk in mm.

a. Write a dplyr pipe to determine the number of observations per tree. [5 pts]

b. Write a dplyr pipe to change the units of age to "years" and circumference to "cm". [5 pts]

c. Write a dplyr pipe to add a column assigning a z-score to each tree, centered around the mean for all trees at a given age. [7 pts]

d. Write a dplyr pipe to calculate the average rate of growth (mm/day) between between age 0 and the final measurement across all trees. [7 pts]

e. Describe the result of the following pipe. Your *description* should touch on both the rows and columns and also describe the substantive question addressed by the result. [6 pts]

```
Orange %>%
  group_by(Tree) %>%
  mutate(
new_growth = c(circumference[1], diff(circumference)),
elapsed = c(age[1], diff(age))
  ) %>%
  group_by(age) %>%
  summarize(
avg_rate = mean(new_growth / elapsed),
se = sd( new_growth / elapsed ) / sqrt(n())
  )
```

Part Two

This is part two of the exam. In this part you should choose *one* of the two questions, either question 5 or question 6, to answer. Write your answer on a blank sheet of paper. Include your name at the top and clearly label which question you have selected to answer. Do your best to follow good coding style and include comments.

Question 5 - S3 Methods in R [40 points]

In this question, you will define a new S3 generic `about` for summarizing basic information about an R object in a compact fashion.

a. Define a new S3 generic `about`. Use `...` in the function template. [5 pts]

b. Define a default method for the `about` generic. Your method should call the `str` generic with the option `give.attr = FALSE`. [5 pts]

c. Define an `about` method for objects of class `data.frame` that prints the following information to the console:

d. A string, "'data.frame': NN obs of PP variables:" with NN and PP replaced with appropriate numbers. This should be its own line. [5 pts]

ii. A string " PP numeric variables: [V1, V2, V3, . . .]" giving the numbers of numeric columns (PP), and listing the names (V1, V2, . . .) as a comma separated list. [10 pts]

iii. A string " PP factor variables: [V1, V2, V3, . . .]" giving the numbers of factor and/or character columns (PP), and listing the names (V1, V2, . . .) as a comma separated list. [10 points]

iv. Define an `about` method for objects of class `tbl` (a tibble, inheriting from the data.table class) that calls the `data.frame` method above and then also prints, when applicable, the number and names of any "list" columns. [5 pts]

Question 6 - Vectorization [40 points]

Below you will find an R script which implements a Monte Carlo study to estimate the number of games won by each of three teams in a hypothetical baseball league. Refer to the header and comments for additional context.

There are three "tasks", denoted `<task X>`, asking you either to rewrite and improve existing portions of the script or complete it by filling in missing code chunks.

Use vectoriztation wherever possible in your answers.

```r
# The Riddler's Fall Classic, (Stats 506 Midterm, Question 6)
# https://fivethirtyeight.com/features/which-baseball-team-will-win-the-riddler-fall-classic/
#
# Background:
#   Three hypothetical baseball teams with each batter having one of four
#   outcomes:
#      BB - a walk granting one base and advancing those on base 1 position,
#      DB - a double with the batter getting to 2nd base and any players on
#           base scoring,
#      HR - a home run, the batter and anyone on base scores,
#       K - a strike out, the batter is out.
#
#   Team 1 (MW): P(BB = .4), P(K = .6)
#   Team 2 (DD): P(DB = .2), P(K = .8)
#   Team 3 (TT): P(HR = .1), P(K = .9)
#
# Notes: The number of batters reaching base by each team in an inning follows a
#  Negative Binomial distribution. The number of runs scored is a funciton of
#  this number.

# Original question: If these teams play in a baseball season (162 games, or 81
#  games between each pair of teams), which team wins the most games on average?

# libraries: -----------------------------------------------------------------
library(tidyverse)

# Functions to simulate a inning or nine: ------------------------------------
sim_inning = function(n = 1, team = c('mw', 'dd', 'tt')){
  # simulates the number of runs scored by "team" in a single inning
  # inputs:
  #   team - one of 'mw', 'dd', or 'tt'
  #   n - an integer for the number of innings to simulate
  # outputs: a length n numeric vector with a simulated number of runs scored
  #          for each of n innings.
  team = match.arg(team)

  if ( team == 'mw' ) {
   # number of batters that walk before there are three outs
   # the fourth and each subsequent walk scores a run
   return( pmax( rnbinom(n, 3, .6) - 3, 0) )
  }
```

(*The script continues on the next page.*)

```
  if ( team == 'dd' ) {
    # number of batters that double before there are three outs
    # the second and subsequent doubles score runs
    return( pmax( rnbinom(n, 3, .8) - 1, 0) )
  }

  if ( team == 'tt' ) {
    # the number of batters that hit home runs before there are three outs
    # each homerun scores a run
    return( rnbinom(n, 3, .9) )
  }

} # ends sim_inning function body

sim_nine = function( n = 1, team = c('mw', 'dd', 'tt') ) {
  # simulates the number of runs score in the first nine innings
  # for one or more games of "riddler league" baseball
  # inputs:
  #    n - the number of games to simulate, an integer
  #    team - one of 'mw', 'dd', or 'tt'
  # outputs: a length n numeric vector, representing a simulated number of runs
  #          in the first 9 innings of n games
  team = match.arg(team)

  #Simulate runs score in 9 inning games.
  #<Task 1> Improve this block of code using vectorization. [20 points]
  runs = matrix(NA, 9, n)
  for ( game in 1:n ){
    for ( inning in 1:9 ){
      runs[inning, game] = sim_inning(n = 1, team = team)
    }
  }

  apply(runs, 2, sum)

}

extras = function(team1, team2) {
  # This function simulates the outcome for games tied after nine innings.
  # Inputs:
  #   team1, team2 - character strings giving the teams to simulate the game
  #    outcome for, conditional on the game being tied after 9 innings.
  #   each in "mw", "dd", or "tt"
  # Outputs: the name of the team that wins, either team1 or team2.
```

(*The script continues on the next page.*)

```
  while ( TRUE ) {
    top = sim_inning( n=1, team=team1)
    bottom = sim_inning( n=1, team =team2)
    if ( top > bottom ) {
      return(team1)
    } else if ( top < bottom ) {
      return(team2)
    }
  }
} #ends extras function body

# Generate a Monte Carlo sample of games: -------------------------------------
mcrep = 1e4
mchalf = mcrep / 2
games_mw = sim_nine(mcrep, team = 'mw')
games_dd = sim_nine(mcrep, team = 'dd')
games_tt = sim_nine(mcrep, team = 'tt')

# Estimate the average number of runs scored per 9 innings by each team: ------
#! <Task 2> 10 points

# Compute the approximate winning probabilities: ------------------------------
# MW vs DD
mw_vs_dd = ifelse( games_mw[1:mchalf] > games_dd[1:mchalf], 'mw', 'dd')
mw_vs_dd_ties = which( games_mw[1:mchalf] == games_dd[1:mchalf] )
mw_vs_dd[ mw_vs_dd_ties ] =
  sapply( mw_vs_dd_ties, function(i) extras('mw', 'dd') )

# MW vs TT
mw_vs_tt = ifelse( games_mw[{mchalf + 1}:mcrep] > games_tt[1:mchalf],
                   'mw', 'tt')
mw_vs_tt_ties = which( games_mw[{mchalf + 1}:mcrep] == games_tt[1:mchalf] )
mw_vs_tt[ mw_vs_tt_ties ] =
  sapply( mw_vs_tt_ties, function(i) extras('mw', 'tt') )

# DD vs TT
dd_vs_tt = ifelse( games_dd[{mchalf + 1}:mcrep] > games_tt[{mchalf + 1}:mcrep],
                   'dd', 'tt')
dd_vs_tt_ties =
  which( games_dd[{mchalf + 1}:mcrep] == games_tt[{mchalf + 1}:mcrep] )
dd_vs_tt[ dd_vs_tt_ties ] =
  sapply( dd_vs_tt_ties, function(i) extras('dd', 'tt') )

# Expected winning totals in a 162 game season for each team: -----------------
## Recall, each pair of teams plays 81 games. Games are independent.
# <Task 3> [10 points]
```