Final Exam

Stats 506, Fall 2019

Monday, December 16, 4-6 pm

Instructions

This is the final exam for Stats 506, Fall 2019. You have two hours to complete the exam. You may use a 3x5 inch hand written index card for notes, but no other notes or electronic devices.

This exam has 8 questions and you should answer all questions. Please write your answers to each question on blank sheets of paper, with your name clearly printed in the top right corner. Use a fresh page for each set of questions as listed below:

1. Questions 1 and 2
2. Question 3
3. Question 4
4. Question 5
5. Questions 6 and 7
6. Question 8

Please note the questions are worth varied number of points, ranging from 10-35 as indicated below:

1. Memory in R, 25 points
2. SAS, 20 points
3. SAS proc sql, 30 points
4. Stata / dplyr, 30 points
5. data.table, 30 points
6. data.table, 10 points
7. parallel computing, 20 points
8. cross validation and parallel computing, 35 points

Question 5 asks you to repeat either question 3 or question 4 using data.table syntax. Please choose just one and keep this in mind while working on these questions.

In addition to this exam, there are four sripts on which the questions are based:

1. "Final Exam, Question 1b"
2. "Final Exam, Quesiton 2"
3. "Final Exam, Question 4"
4. "Final Exam, Question 4 - R"

Please make sure you have one of each before beginning the exam.

Good luck!

Question 1 [25 points]

A [15 points]

For each of the R objects below, pick the most appropriate choice for how much memory it occupies, as reported by `pryr::object_size()`. Recall that `pryr::object_size()` only counts values with multiple names bound to it once. Also recall that each integer requires 4B to store, each double or pointer 8B, and short strings in the global string library 56B.

The character vector `stringr::fruit` has length 80 and, if it helps, `pryr::object_size(stringr::fruit)` returns `5.55 kB`.

Here are the choices. Each choice should be used exactly once.

(a) 326 kB (b) 80 kB (c) 8 MB (d) 680 kB (e) 68.9 kB

Pick the best choice from the options above.

i. `matrix( 1:2e2, nrow = 2e2, ncol = 1e4)`

ii. `rep(stringr::fruit, 100)`

iii. `factor( rep(stringr::fruit, 1000) )`

iv. `runif(1e4)`

v. (see below)

```
sapply(1:1e4,
 function(x){
   y = x
   names(y) = paste0(letters[ {x - 1} %% 26 + 1 ], x %/% 26)
   y
 })
```

B [10 points]

Consider the code provide in the script "Final Exam, Question 1b" adapted from the solution to problem set 3. Estimate, how much working memory (RAM) is required, rounding up to the nearest GB. Assume that R itself and any other objects in the global environment occupy approximately 150 MB. Explain your answer. *Hint: Add up objects created within the function environment that have a memory foot prints on the order of 0.5GB or larger, stopping after the two largest objects are removed.*

Question 2 [20 points]

Consider the SAS script "Final Exam, Question 2" distributed with the exam.

A [8 points]

Complete the following tasks.

i. `<task1>`: Fill in the header with an appropriate description and any other missing information.

ii. `<task2>`: Write a short descriptive phrase to fill in the blank for `<task2>` in the script. Use no more than 2-5 words.

1. `<task3>`: Provide a descriptive name for the macro and associated tables labeled `<task3>`. Use `snake_case` if needed.

B [12 points]

Answer the following questions.

i. How many sas7bdat files will be created in the folder `./` by running this script? What are their names? How many columns do they have?

ii. How many tables will this script create in the work library? What are their names? (You do not need to say how many columns these have.) Note: sas macros work by code substitution and are not encapsulated in environments like functions in R or other languages.

Question 3 [30 points]

In this question, you will rewrite parts of the the SAS script "Final Exam, Question 2" from the previous question using `proc sql`.

A [10 points]

Rewrite the tasks located under the comment "Find rows corresponding to the maximum category for each storm" using `proc sql`. (Stop at the next comment.) Your solution should use a single sql statement without creating any intermediate tables.

B [20 points]

Rewrite the body of the macro (between `%macro ...;` and `%mend;`) using `proc sql`. The best solution will avoid creating any tables in the `work` library; however, a literal translation is also acceptable.

In your solution, use the descriptive name you selected in `<task3>` of question 2 part A, rather than the generic `task3`.

Question 4 [30 points]

In this question you will demonstrate your understanding of Stata code by translating select portions of a script into R, using dplyr syntax. For the original script, refer to "Stats 506 Final, Question 4". A partial translation has also been provided as "Stats 506 Final, Question 4 - R".

However, within that R code are three missing chunks labeled "A", "B", and "C". Write dplyr code to complete these chunks, using the Stata script as a reference for what should be accomplished. Each chunk is worth 10 points.

Question 5 [30 points]

Repeat either question 3 or question 4 in R using `data.table`. Be sure to do all parts. At the top of your answer, clearly indicate which of the two questions you have chosen to answer.

Question 6 [10 points]

This question relies on Medicare payment data available in the "Physician and Other Supplier public use files". Each row of this data details the frequency with which individual medical providers bill Medicare for a particular service and the average payment amount for those services.

The key variables to be used are:

- HCPCS_CODE - a unique identifier for each billable medical service
- HCPCS_DESCRIPTION - a description for each HCPCS code
- LINE_SRVC_CNT - the number of times each provider billed a particular code
- AVERAGE_MEDICARE_PAYMENT_AMT - the average amount a provider was paid for each billed service
- NPI - a unique identifier for each provider

Assume the data has been read into R using `data.table::fread()` as a data.table object `med_data`.

Consider the following code using data.table syntax.

```
med_data[ , `:=`(totpay = LINE_SRVC_CNT * AVERAGE_MEDICARE_PAYMENT_AMT)]
avg_pay =
 med_data[ , .( n = sum(LINE_SRVC_CNT), totpay = sum(TOTPAY),
                desc = HCPCS_CODE_DESCRIPTION[1]
              ), .(HCPCS_CODE)]
avg_pay[ , `:=`(avg = totpay / n)]
top10 = avg_pay[ n > 1e5 ][order(-avg)][1:10]
```

a. In one sentence, describe the data.table `top10`.

b. Write `dplyr` code to produce `top10` from `med_data`. Your solution should be a single chain of piped commands.

Question 7 [20 points]

Consider the following function which waits some amount of time depending on the value of its input `x`.

```
wait = function(x){
 # wait 1/x seconds unless x is integer-valued
 # when x is integer valued, wait x seconds if even or
 #  6 * x seconds if odd

 if ( x == floor(x) ) {
   y = ifelse( {x %% 2} == 0, x, 6 * x )
   Sys.sleep(y)
 } else {
   Sys.sleep( 1 / x )
 }
}
```

For each of the following, determine whether it would typically complete sooner with the option `mc.preschedule = TRUE` or `mc.preschedule = FALSE`. Then, approximate the typical run time for the option you chose. Briefly explain your answer.

```
## a
mclapply(1:6, wait, mc.cores = 2, mc.preschedule = ??)`

## b
mclapply( runif(1000, min = 5e1, max = 1.5e2), wait,
  mc.cores = 2, mc.preschedule = ??)
```

For reference, consider the following:

```
system.time({
  parallel::mclapply(1:1000, function(x) { Sys.time() },
                     mc.cores = 2, mc.preschedule = TRUE)
})
```

```
##    user  system elapsed
##   0.012   0.010   0.027
```

```
system.time({
  parallel::mclapply(1:1000, function(x) { Sys.time() },
                     mc.cores = 2, mc.preschedule = FALSE)
})
```

```
##    user  system elapsed
##   2.498   5.856   3.639
```

Question 8 [35 points]

In this question, you will write code to compare two machine learning regression models. Each model has a tuning parameter to be selected using cross validation.

You have 100,000 data points organized into a `data.table` object `dt`. This data table has a continuous response `y` and 200 features `x1:x200`.

Model 1 is based on kernel regression and has a bandwidth parameter `h`. Assume that you have already written a function `fit_mode1` which takes a data.table and a value of `h` and returns the fitted model object: `fit_model1(dt, h)`.

Model 2 utilizes ridge regression and has a smoothing parameter `lambda`. As with model 1, assume you have written a function `fit_model2(dt, lambda)` which accepts an input data set and a value of `lambda` and returns the fitted model object.

Both model objects belong to classes with associated `predict` methods, `predict(fitted_model, new_data)` that return a prediction for `y` based on the fitted model and new data.

You need to use 20-fold cross validation to choose the best `h` and `lambda` from a set of previously determined values `h_ops` and `lambda_ops`. Each model takes around 1s to fit.

a. [5 pts] Add a column to `dt` by reference, which randomly divides it into training and testing sets using an 80/20 split.

b. [5 pts] Divide the training set into 20 folds for cross validation and then reorder the data so that the rows for each fold are contiguous in memory.

c. [20 pts] Use `mclapply`, `futures`, or `foreach` and `doParallel` to pick `h` for model 1 and `lambda` for model 2 that minimize the cross-validated mean squared error (MSE). Assume you have 8 cores available for these computations.

d. [5 pts] Using the values of `h` and `lambda` selected in the previous part, write code to compare the MSE on the test data, making sure to refit the models using the full training data.