Final Project Guidelines

VERSION: 2018-03-11 · 23:53:51

1. Objectives

1. Apply a combination of the computational and inferential techniques addressed in this class to a statistical problem which is of interest to you. Some possible lines of inquiry are:
   - Propose and compare several statistical models for analyzing a type of dataset or problem, and evaluate/compare their performance on a real dataset.
   - Efficiently implement an algorithm to fit a statistical model of interest, possibly improving the implementation/efficiency of one seen in class. Evaluate/compare its performance on real and/or simulated data.
   - Perform a thorough and computationally intensive analysis a real dataset of interest and present your domain-specific (e.g., scientific, financial) conclusions/findings.
2. Create and thoroughly test a software library (i.e., **R** package) containing the tools you used above.
3. Clearly and effectively communicate your project's findings.

2. Format

- Groups of 1-2 students.
- STAT 840/CM 760 students must come up with their own projects[1]. STAT 440 students may come up with their own or select/modify one of the sample projects described in a separate document.
- The project itself consists of the following components (details below):
  1. An **8-10 page final report**.
  2. An **R package** containing your functions.
  3. A **1-2 page project proposal** due well before the final report, to see whether you are on the right track.

In addition, **all students** (not just one per team) are required to come discuss their project with me in person for 5-10 minutes at some point during class time or office hours between now and the end of the semester.

[1]But you should still go over the STAT 440 sample projects to get an idea of the expected difficulty level.

2.1. Final Report

To be written and formatted in the style of a scientific article. You must include an abstract and references (see below), but aside from that you are free to modify the following template:

0. Abstract: 100-150 words. An abstract must make the following points crystal clear without getting into technical details:

i. *Motivation:* Why is the problem relevant/important?
ii. *Challenge:* What about the problem is unresolved by existing approaches?
iii. *Contribution:* How do you address this problem? What did you discover?

1. Introduction: The introduction is like the abstract but with more detail. This is the place to cite the work of others on the problem. Your project must contain a minimum of 6 references to published work[2]. The introduction ends with a brief outline of the remaining sections in the article.

2. Methodology: This is where things get more technical in terms of notation, models, equations, algorithm descriptions, etc. Whatever is too technical however (e.g., lengthy derivations and low-level descriptions, additional figures and tables, computer code not contained in your package, etc.) should go to the Appendix.

3. Results: You must apply your methods to a real dataset, and/or to a simulated dataset with carefully chosen parameters (i.e., relevant to a specific application, comparing interpretable settings e.g., low/high, positive/negative correlation, etc). Depending on the nature of your project, this is the place to address some of the following questions:

- Which of the methods/models you tried works best?
- Under what condition(s) does each method/model work well/poorly?
- Do any of the models actually fit the data?
- What can you conclude about the dataset that you are studying?

4. Discussion: Briefly summarize your contributions, reminding the reader of both strengths and limitations of your methodology/analysis. Outline directions for future research.

5. Appendix: Include all computer code not in the **R** package (i.e., data analysis, visual tests), math, extra analyses you did, and whatever else you think is important but don't have space to include in the 10 pages. There is no limit on the length of the Appendix.

[2]For instance, Wikipedia does not count as published work, but arxiv is OK.

2.2. R Package

As we saw in Quiz 3, an **R** package is essentially just a folder of **R** functions. However, the process of creating an **R** package encourages you to think carefully about the design and structure of your code. Therefore, code which is made into a package is almost always better written and organized than code which is just lying around in one or more **R** scripts. Moreover, packages allow you to clearly document and rigorously test your functions – a critical component of the computational requirement for this project.

- All code for this project must be either written in **R**, or interfaced to **R** from another language. In other words, the function tests and analyses must be executable from within an **R** session. Note that **C++**, **Python**, and **Julia** programs can be called from **R** with the packages `Rcpp`, `reticulate`, and `JuliaCall` respectively[3].

- By all means you are allowed to use existing software libraries – see the LEARN tutorial "*Simple Steps for Writing an **R** Package*" for instructions on how to use functions from other **R** packages within your own. However, solely relying on other people's packages is unacceptable.

- **Most if not all** functions in your package *and* those you take from external libraries must be thoroughly tested. For example: unsimplified vs simplified loglikelihoods, histogram of random samples vs their PDF, MLE vs "projection plots", etc. For the last test we have been using the function `mle.check` throughout this class. This function (renamed to `optim_proj`) is now part of the **R** package `optimCheck` available on LEARN, which you are required to use instead of `mle.check` for this project.

- Most numerical tests (e.g., difference between simplified and unsimplified loglikelihoods) can be formally included as unit tests within your package itself using the **R** package `testthat`. See here for detailed instructions on how to do this. As we saw in Quiz 3, even projection plots (as returned by the function formerly known as `mle.check`) can be converted into numerical unit tests. See the `optimCheck` subfolder `tests/testthat` for several examples of how to automate such tests with the `testthat` package.

- As has been done for all **R** code posted online, all exported package functions must be documented using `roxygen2`. See the package-writing tutorial on LEARN for further instructions.

[3]Under exceptional circumstances (i.e., difficulty and scope of the project), you may create a package (or its equivalent) in a language other than **R**, but this requires **explicit written permission** from the instructor.

2.3. Proposal

• The proposal should motivate the problem and identify knowledge gaps as in the Abstract and Introduction, but replace contributions with what you have done so far and (especially) what you plan to do.

• The purpose of the proposal is to get you thinking about the project and receive feedback from us well before the due date, so you can budget your remaining time and resources accordingly.

• The proposal will be evaluated on concreteness, clarity, and feasibility. In order to gauge this last aspect you will need to try a couple of things out.

3. Deadlines

- The final report and **R** package are due on Friday, April 20 at 11:59 PM.
- The proposal is due Thursday, March 29 at 11:59 PM. This will count as Quiz 5.

  ⋆ **To incentivize you to get started on the projects early, if the proposal grade is higher than the final report grade, it could count for up to 20% of the project grade.**

  ⋆ **Conversely, simply paraphrasing the sample project descriptions posted online will result in a proposal grade of zero.**
- **Lateness:** 5% penalty per day. No final projects will be accepted after Tuesday, April 24. Late proposals will not be graded.

3.1. Online Submission

Both the report and the project description must be submitted in PDF format. Everything (proposal, report, **R** package) must be submitted on LEARN as follows:

1. Login to LEARN and join a Group: at the top of the screen, click

   `Connect > Groups > View Available Groups`

   Agree on a Group number (say X) between 1-100 with your other team members, and click **`Join Group`** beside **`Project > Group`** X.

   Note: If you do not join a LEARN group, you will not receive a Project grade.
2. Submit file(s): at the top of the screen, click **`Assessments > Dropbox`**, then **`Group`** X**`: Proposal/Report`**, then **`Add a File`**. **Note:** The names of all collaborators must appear on all files.
3. Please label your PDF files as **`group`**X**`-proposal.pdf`** and **`group`**X**`-report.pdf`**, where X is the group number. The **R** package should be submitted as a `tar.gz` file (created with `devtools::build`) and can be named whatever you want. Please *do not* submit additional files for **R** code. All code not in the package should be in the Appendix.

4. Grading

This is an open-ended project which you should think of as an opportunity for statistical thinking and exploring areas of interest, rather than a set of questions for which you must provide the correct, unique answer. Aside from basic presentation (legibility, labeling of figure axes, keeping only 2-3 decimals out of 8, etc.), here are some elements that I will be looking for in terms of evaluation:

⋆ **Presentation:** Clarity and organization of ideas. Make sure to emphasize your main results. Use of proper formatting (e.g., sections headings, numbered figures/tables with informative captions, plot axes labeled and sized correctly, etc.) will greatly improve the legibility of your report. While you are not required to use `rmarkdown`, lower quality formatting than the default **R** Markdown output is insufficient. Think carefully about how to display results (e.g., figures vs. tables, relevant information needed to make your point). Whenever possible, provide uncertainties for point estimates. If not possible, explain why.

⋆ **Computing:** Magnitude of the computing challenge. Efficiency of coding. You are allowed to use other people's packages, but only doing so is insufficient. Extensive code testing and commenting is mandatory. Include all code **R** code not in the package (i.e., data preprocessing and analysis, graphical tests) in the Appendix, clearly organized and with section headings.

⋆ **Reasoning:** Essentially, I'm looking for your ability to justify the subjective decisions you'll need to make and to think critically about a model or dataset. For example, how did you pick the dataset or parameters for a simulation study? Why did you include or exclude specific elements from your analysis? How did you modify the sample projects based on resource constraints? What did you learn about the data, model, or methodology? What are the pros and cons of said model/methodology?

4.1. Additional Comments

- *Whatever you do, do it well.* You will be rewarded more for clarity and thoroughness on something less technically challenging, than for something very difficult but poorly explained.

- Do feel free to modify the sample projects as you see fit (with appropriate justification).

- If applicable, highlight the computational challenges you had to overcome.

- I have several other potential projects not detailed here. If you are interested, don't hesitate to ask.