Final Project

Due: June 22, 2018 @ 10:30am

Overview

In lieu of a written final exam, you and your classmates will construct a an annotated list of R functions in a group repository to help you review the material you learned throughout the semester and provide you with a set of notes that you can use for reference outside of the course. In addition to the notes, individually you will also write a short comparative discussion of related simulations, which provides you with a glimpse into another important part of the computational sciences.

The class will be provided with a starter repository on Github that must be used for putting together the R notes, following the instructions provided below. Everyone's submission must commited and merged into the `master` branch of the group repository by the start of the final interviews on **June 22, 2018** at 10:30am. The comparative discussion, which will be submitted separately, is also due by **June 22, 2018** at 10:30am.

Students will also be scheduled to meet with the course instructor for a 10 minute final interview on **June 22, 2018** during the scheduled final exam period from 10:30am to 1:15pm. If you have an unavoidable conflict with this time slot, you must inform the instructor and schedule another time during the final exams period that you will do the interview.

Grading

The assessment of your contribution to the R notes/comparative discussion and your performance during the final interview will be combined into a single grade, with the R notes/submission part being worth 70% and the final interview worth 30%. As per the syllabus, the combined grade is worth 25% of your overall course grade.

Final project guidelines

Consider the following guidelines as you get started with putting together your contribution to the final project, **but please note that this list is not exhaustive**. It is important that you fulfill the basic requirements for the R notes and the comparative discussion of related simulations, but organization, thoroughness, and creativity are encouraged. Grades will reflect the overall quality of the final project.

Annotated list of R functions

Purpose — A significant portion of the course is dedicated to learning how to use R and the `tidyverse` ecosystem to wrangle and explore data, and then analyze it using statistics. Remembering the different functions, how they are used, and what problems they solve requires consistent practice and review. This is why it's important to have a set of notes that explain how a function works in your own words, which can be referenced long after you've completed this course.

For your final project — You will be assigned a set of functions that we used during the course that you are responsible for annotating. Your annotations will need to be merged into a single, uniform document with the rest of your classmates that will ideally serve as a useful reference after the course is complete. To get everyone started, you will be provided with a group starter repository and a notes template containing a pre-filled example for the `ggplot2` syntax and how to use `geom_histogram()`. Use these examples as inspiration for how to put together notes on the other functions in R.

Annotating the R functions

Unless told otherwise, when writing notes for a function, please include the following:

- The function's name
- The important inputs
  - If the input was used in class or for a homework assignment, then it's important
  - For `ggplot2` functions, include a separate subsection for important aesthetic mappings (see example in your template file)
- A summary — 1 or 2 sentences — of what the function does
- An example showing how to use the function
  - **Your example cannot use the same dataset as the examples found in the help documentation or from class.** For example, when explaining the `dplyr` function you should not use the `presidential` dataset, but you *can* use a dataset from one of the homework assignments.

List of R functions

Your notes should discuss the following packages and functions:

- The `ggplot2` package
  - `geom_histogram()`, `geom_boxplot()`, `geom_point()`, `geom_bar()`, `geom_col()`, `stat_ecdf()`, `geom_smooth()`, `facet_wrap()`, `facet_grid()`
  - Basic instructions on how to change the axes labels and give the plot a title
- The `readr` package
  - `read_csv()`, `read_rds()`, `write_csv()`, `write_rds()`
- The `dplyr` package
  - `select()`, `slice()`, `rename()`, `recode()`, `arrange()`, `filter()`, `mutate()`, `group_by()`, `summarize()`, `count()`, `cume_dist()`, `combine()`, `pull()`
- The `tibble` package
  - Instructions on manually creating a data frame using `data_frame()`
- The `tidyr` package
  - `gather()`, `spread()`, `separate()`, `unite()`
- The `rvest` package
  - Instead of notes for each function, describe the procedure for using the SelectorGadget extension with `read_html()`, `html_nodes()`, and `html_text()` to perform basic webscraping tasks
- Statistics functions
  - Summary statistics functions: `mean()`, `median()`, `sd()`, `IQR()`, `min()`, and `max()`
  - Linear modeling: `lm()`
  - Instructions on how to create a Q-Q plot using `geom_qq()` from `ggplot2` and then plot the **ideal reference line** to check for deviations from normality
- The `infer` package
  - Describe the inputs for the four functions: `specify()`, `hypothesize()`, `generate()`, and `calculate()`, see the class 16 slides for guidance
  - Instructions **with an example** on how you use `infer` to conduct a hypothesis test: how you simulate the null distribution, and how you calculate the p-value

        - Test must be between two variables (one response variable and one explanatory variable), one explanatory variable must be categorical, the response can be either categorical or numerical
        - Example should show how we compute the *p*-value for a one-sided test and for a two-sided test
    - Instructions **with an example** on how you use `infer` to find a confidence interval using bootstrapping, including how you would find different size intervals, for example a 90% interval or a 95% interval
        - Confidence interval must be computed for a **difference** of two values, such as *means* or *proportions*
        - Consult the class 17 slides for guidance on how to correctly compute the upper and lower bounds!
- The `modelr` package
    - **An example** of showing how to plot a simple linear model with one explanatory variable on top of the data in a scatter plot using `data_grid()` (or `seq_range()`), `add_predictions()`, and `geom_line()` from `ggplot2`
    - Instructions **with an example** on how to use `add_predictions()` to create *observed versus predicted values* plots
    - Instructions **with an example** on how to use `add_residuals()`, `add_predictions()`, and `geom_ref_line()` to create *residual versus predicted values* plots
    - Instructions **with an example** on how you interpret the *observed versus predicted* and *residual versus predicted* plots to determine whether the conditions for using a linear model are met: **linearity**, **nearly normal residuals**, and **constant variability**

Remember, these notes are ultimately for you, so think about what you would find helpful when you refer back to these.

Merging into the master branch

Students will complete their annotations in separate branches of the final project repository, writing their annotations under the appropriate header. When your annotations are complete, you will need to have your work merged into the `master` branch of the repository. Before you can do this, you will need to have your work edited and reviewed by one of your classmates. This will be completing using Github's **Pull Request** functionality. The instructor will set up the **Pull Requests** and assign the peer reviewers. In order to indicate that you are ready for a review, you should post the following in the discussion box of the **Pull Request**:

> My annotations are ready for review @username

You will need to replace `@username` with your reviewer's actual Github username. You should also send your reviewer a direct message on Slack or email him or her to begin the review. After you receive your feedback, you will use the suggestions to update and correct your submission. After your edits are complete and commited to your branch, message the instructor on Slack to check to see that you've satisfied the reviewer's comments. If you have, the instructor will approve the merge and add your contribution to the master version of the document.

- **Due date for annotations draft:** 12:00pm noon on Thursday, June 21st. This is to allow enough time for the reviews to be completed and for you to commit any final edits to your branch.
- **Peer reviews must be completed by 6:00pm on Thursday, June 21st and any final edits should be commited no later than 9:00am on Friday, June 22nd.**

Comparative discussion of simulations

Purpose — The field of computational and data sciences extends beyond the topics focused on in this course. From the CDS-101 perspective, we've used the terms **model** and **simulation** as shorthand for *data-driven* models and simulations. Yet, there is an alternative approach to model and simulation building that works in the opposite direction and is an indispensable tool in the natural sciences, engineering, and computational social sciences. This class of models and simulations *generate* predictions and data without using an underlying dataset as input. To distinguish these from their *data-driven* counterparts, we will refer to them as follows:

- A *microscopic* or *mechanism-driven* model or simulation is based on the known laws of nature. An example is deriving equations of motion for the planets in our solar system using Newton's law of universal gravitation.

After building this kind of model or simulation, the researcher will scan the model's parameter space and look for trends in the predictions and outputs, which are then compared against experimental data (if available). If the model or simulation generates predictions or outputs that accord with the experimental data, then the proposed mechanism can be regarded as a plausible explanation for observed trends. However, if the predictions or outputs fail to agree with the experimental data, then the model or simulation is falsified and the proposed mechanism is rejected.

For your final project — You will visit the following two webpages containing short summaries of two simulations that share a common lineage:

- Ising Model (https://jkglasbrenner.github.io/ising-model-js/)
- Schelling's Model of Segregation (https://jkglasbrenner.github.io/schelling-model-js/)

Each page also contains the simulation itself implemented in Javascript, which runs inside the web browser. The simulations are visual and interactive, allowing you to change a small set of parameters using a simple dashboard. After becoming familiar with the different simulations and developing a basic intuition for how each one behaves, you will then compare and contrast them in a short write-up. Your comparative discussion must be at least 2 paragraphs in length (a minimum of one paragraph per simulation) and include the following:

- At least three ways in which the simulations are similar to one another
- For each simulation, at least one way it is *different* from the other one
- Pick one of the simulations and suggest a feature or rule that you could add to it that would change its outputs and predictions. You only need to do this using plain language, you are not expected to write any code for this. Be sure to hypothesize what you think the changes will do and what they might mean. For example, how do you anticipate that your proposed change will simulate different physical mechanisms or human behavior?