Stat 610 Final Project Options

Writeup is due Tuesday, December 16, 11:59pm. Presentations will be in class December 9 and December 11. You may work in groups of up to 2 for this assignment.

October 24 is the deadline for letting me know what project you will be working on and who (if anyone) you will be working with.

No matter what project you choose, please set up a time to come and talk to me: I'll give you more details and tell you about problems you might want to look out for.

Assignment

You can choose from the following options:

- *k-means*: k-means is a commonly-used clustering method. You will write a function that takes a data matrix, a value for k (the number of clusters), and, optionally, a set of starting positions for the clusters. The function should implement the k-means algorithm for those values. In addition, you should have a function that does at least one of the following:
  - Creates a visualization of the algorithm's progress, with plots (or better, an animation) showing the cluster centers and cluster assignments as the algorithm progresses.
  - Aids users in their choice of k. The function should take a range of values of k, compute a measure of cluster quality for each of those values, and either plot or return a data structure containing the quality measure.
- *Graph layout*: One fairly classic problem in computer science has to do with visualizing graphs. There are a number of algorithms for placing the nodes of a graph in the plane so as to minimize the length of the edges between the nodes and and to minimize the number of crossings of edges between the nodes.

  You will implement one of the graph layout algorithms described here: `https://cs.brown.edu/people/rtamassi/gdhandbook/chapters/force-directed.pdf`. Your function should take a graph, encoded as an adjacency matrix (a $n \times n$ matrix, n the number of nodes in the graph with ijth element equal to 1 if there is an edge between nodes i and j and 0 otherwise), and should return a layout (an $n \times 2$ matrix giving the x and y position for each node in the graph).
- *Neighbor-joining trees*: One method for estimating phylogenetic relationships among a set of taxa is referred to as neighbor-joining clustering. You will write a function that takes a matrix describing the distances between a set of taxa, performs the neighbor-joining algorithm on that distance matrix (`https://en.wikipedia.org/wiki/Neighbor_joining`), and returns a tree object that can be visualized using either the `ape` or `igraph` R packages.
- *Hubs and authorities for network analysis*: Supreme Court opinions cite previous opinions, and these citations can be used to define a network. One way of characterizing nodes in a network is by assigning them "hub" and "authority" scores (`https://nlp.stanford.edu/`

IR-book/html/htmledition/hubs-and-authorities-1.html). You will write functions that compute "hub" and "authority" scores for nodes in a network, and apply them to a network of Supreme Court opinions.

This project is based on Fowler and Jeon, "The Authority of Supreme Court Precedent" *Social Networks*, (2008) (pdf available at https://www.sciencedirect.com/science/article/pii/S0378873307000378). Using the data at https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/XMBQL6, recreate Figure 6 or Figure 10 in the paper, or make an analogous figure containing cases you are interested in. The website contains pre-computed hub and authority scores for the opinions. You will of course re-compute these, but you can use them to check your work.

– *Markov models for language*: A naive but sometimes amusing model for language is a Markov model. In this model, language is assumed to be a sequence in which the next word is drawn from a distribution that depends only on the current word. The model can be elaborated slightly to one in which the next word depends on the current word along with the previous m words, for some fixed value of m.

  You will write functions that fit such models from text and that generate new text from the fitted model. You can use as input text either the books we used in homework 1 or text of your choice.

– *Approximate Bayesian computation for disease outbreaks*: Epidemiologists have developed models for disease spread, but these models often lead to intractable likelihoods. One strategy for fitting parameters in such models is approximate Bayesian computation (ABC), which is a simulation-based method for fitting models to data (we will discuss ABC in detail later in the course).

  You will write functions that perform ABC to fit parameters in a model of the spread of a virus. This will involve:

  – Drawing parameters from a prior distribution,

  – Drawing data according to a probability model given the parameters,

  – Computing a similarity measure between the simulated data and an observed dataset, and finally

  – Keeping or discarding the samples based on that similarity.

  The goal will be to recreate Figures 3a and 3c in Tony and Stumpf, "Simulation-based model selection for dynamical systems in systems and population biology", *Bioinformatics* (2010) (available at https://academic.oup.com/bioinformatics/article/26/1/104/182571), with the data provided in the supplement to that paper.

– *Model selection and post-selection inference*: In regression modeling, we sometimes want to include only a subset of our variables in the model and would like that decision to be data dependent. Two solutions to this problem are *forward stepwise selection* and *backward stepwise selection*, in which you fit a sequence of models, either adding or subtracting one variable at a time until some stopping criterion is reached. These procedures work for variable selection, but they invalidate the standard methods of inference in linear models.

You will:

- Write code implementing either forward stepwise selection or backward stepwise selection.
- Set up a simulation experiment, run your method on simulated data, obtain p-values or confidence intervals in the selected model, and report on whether the hypothesis tests and confidence intervals were valid.

– *Projection pursuit*: Projection pursuit (https://en.wikipedia.org/wiki/Projection_pursuit) is a method for finding "interesting" representations of a multivariate data set, where interesting can be defined by the user but is usually defined as maximally different from a normal distribution.

You will implement projection pursuit and run your algorithm on real and simulated data.

If you would like, you may also perform a replication or a partial replication of a published paper or work on a project related to your research. The caveat is that it must involve a reasonable computing component: if the work would be primarily data cleaning and using packages other people have developed, it is not a good candidate for a project. If you think you have a good idea, email me with what you would like to do, and a short description about why replication would make a good project. No guarantees that I will agree with you, but you have the option.

Submission parameters

- Github repository containing your code. You are required to have at least 3 different commits. If you are working in a group, each group member should have primary responsibility for part of the codebase, and the commits should show the code that they contributed. Part of each individuals' grade will be based on the quality and quantity of the code he or she is responsible for.
- Github repository should also containing a file with tests for your code. If we run

  ```
  devtools::test_dir('.')
  ```

  the tests should run and your code should pass the tests.
- A pdf writeup describing what you did and what results you got.
- Each group will present their work in the last week of class. The presentation will take the form of a code review: groups will describe what they did and present evidence that their code works correctly, and other members of the class will ask questions about design choices.