Final Project

36-350, Fall 2011

You will work on projects in groups of 3. You will upload a text file to Blackboard with your ranking of the following projects, and the instructors will assign you to groups. You must give us your rankings by Wednesday, 9 November at 11:59 pm. (If you do not tell us your preferences, we will assign preferences to you.) There will be three components to the project: an oral presentation during the last week of class (approximately 15 minutes in duration), a written report describing the problem and what you did to solve it, and (documented) code.

1. *Drunken chessmaster* Imagine playing chess with only one piece (a knight, rook, queen, or bishop, ...) by always selecting your next move at random from all possible legal moves. This is something a drunken chessmaster might do. How many moves would it take for your piece to return to where it started? Does it matter where you start? If you play this way for a very long time and then stop which square will you most likely land on? You will write code to simulate this process, to estimate averages and distributions of answers to these questions and others like it.

2. *Stepwise model selection* In regression problems that often come up in data mining, there are very many covariates $X_1, \ldots, X_p$ that are potentially related with the response Y. An important problem is determining which subset of variables are the best predictors of Y. When p is large, using all p variables may be a bad idea; it is also computationally intractable to try every possible subset of the p-variables. Instead one may try fitting a sequence of models in a greedy manner by starting from 0 variables and then adding one variable at a time by choosing the "best" variable at each step according to some criterion. Alternatively, start with a full model using all p variables and then remove the "worst" variable at each step. We will provide you with data consisting of a vector of responses $\mathbf{Y}$ and a matrix of variables $\mathbf{X}$. You will write code to fit a sequence of linear regression models using ordinary least squares and some criterion from adding/removing variables to the model. Your code will also perform cross-validation to estimate the mean squared prediction error of each candidate model and to choose the best one.

3. *Word frequency* A classical claim in quantitative linguistics is **Zipf's law**, which says that the number of words from the dictionary which appear k or more times in a large text or collection of texts is proportional to

$k^{-\alpha}$: a few words appear a very large number of times, a vast number of words appear only once. We will provide you with a canonical text in electronic form; you will write code to estimate α from the text, plot the fitted distribution and the data, and assess uncertainty and goodness of fit.

4. *Document classification* We will provide you with news stories from the New York *Times* with known subject classification. You will write code to extract the text, turn the word frequencies into features, and automatically sort the text into categories, using regression-like methods. We will provide texts for two groups, one learning to discriminate articles about art from articles about music, the other learning to discriminate news stories from editorials.

5. *Markov chain language models* A simple statistical model of language is that words follow a Markov chain, with the next word being independent of earlier words given the latest word. You will write code to fit this model to a collection of documents, to simulate new text from it, and to compare it to a second-order Markov chain, where the next word is independent of earlier words given the latest *two* words.

6. *Markov chain genetic models* DNA consists of a series of distinct "base" molecules, conventionally written A, C, G, T. The sequence of bases specifies the genetic information used to grow organisms, such as yourselves. A simple statistical model of DNA is that bases follow a Markov chain, with the next base being independent of earlier bases given the latest base. You will write code to fit this model to the genome of a real organism (the social slime mold *Dictyostelium discoideum*), to simulate new DNA from it, and to compare it to k^{th}-order Markov chains, where the next base is independent of earlier bases given the latest k bases.