Gov 2001: Problem Set 1

Due Wednesday, February 3 at 6pm

Important Reminder:

The deadline for choosing a partner and a paper to replicate approved is **February 24, 2016**. Send a pdf of your proposed paper to Prof. Gary King and CC both TFs). We will approve the paper (or tell you to look for a new one). Include within your email a brief note (approx. 2-5 sentences) explaining why this paper is a good choice.

Instructions

You should submit your answers and R code to the problems below using the Quizzes section on Canvas.

Problem 1: Probability by Simulation

Simulation is an extremely powerful tool that we will utilize frequently throughout the semester. The following problem provides some practice calculating complex quantities using simulation.
Suppose you have been invited to the poker world championships. You will be playing five-card draw with a standard 52 card deck. You get five cards, then after considering your hand you have the option of discarding up to n cards ($n \in 1, ..., 5$) in order to draw n new cards from the deck. You want to determine the exact probabilities of getting certain hands in the game.

1.1

Use simulation to approximate the probability of getting three of a kind (three of the same number) when you are first dealt five cards (Note: for better accuracy, simulate at least 50,000 hands). Set the seed to 02138.

1.2

Use simulation to approximate the probability of getting a full house (three of a kind, and two of a different kind) when you are first dealt five cards. Set the seed to 8787.

1.3

Say you are dealt five cards, the Jack of hearts, the Jack of diamonds, the three of clubs, the two of diamonds, and the five of spades. You decide to turn in the last three cards and get three more cards. The cards you turn in do not go back into the deck. Given

what you have in your hand and what you put down, use simulation to approximate the probability of getting a full house from the second draw. (For simplicity, assume you have no opponents.) Set the seed to 02138.

Problem 2: More Probability by Simulation

One hundred people line up to board a plane with 100 seats. The first person in line is has lost her boarding pass, so she randomly chooses a seat. After that, each person entering the plane either sits in his or her assigned seat, if it is available, or, if not, chooses an unoccupied seat randomly.

2.1

Using simulation find the probability that when the 100th passenger finally enters the plane, she finds her seat unoccupied? Use at least 10000 iterations and set your starting seed to 02138.

2.2

Discuss the intuition behind your answer to 2.1. Why does it make sense given the structure of the problem?

Problem 3: Linear Regression and Bootstrapping

In this problem we will explore some of the pre-requisites for the course and introduce bootstrapping, a powerful non-parametric technique.

3.1

Using matrix algebra, write an R function that estimates the Ordinary Least Squares (OLS) coefficients, the σ^2 ancillary parameter, and the coefficient standard errors. Your function should take take two inputs: a vector of the dependent variable and a matrix of explanatory variables. Do not use any pre-programmed functions that perform OLS, such as `lm()`, except to check your results.

3.2

You are given the following linear model, which we will later use to generate data:

$$y_i = 4 + .5x_{i,1} - 4x_{i,2} + x_{i,3} + \epsilon_i$$

where $\epsilon_i \sim \text{Normal}(\mu = 0, \sigma^2 = 36)$.
Rephrase this model in terms of its systematic and stochastic components.

3.3

Generate one set of simulated outcomes according to the model outlined above using the covariates contained in the dataset `covs.csv` on the Canvas website.[1] Your simulated data will contain 1000 observations. Set the 'random' number generator in `R` using the `set.seed(02138)` command before drawing any random quantities. Use your function from section 3.1 to estimate the OLS coefficients and their standard errors. Report your results.

3.4

Bootstrapping is a data resampling procedure for statistical inference, which in one common form uses the empirical distribution of the observed data as a stand-in for the data's population distribution. Assuming that the empirical distribution is a reasonable approximation of the population distribution, we can resample the empirical distribution to generate hypothetical datasets. Model parameters can then be estimated for each of these datasets in order to simulate the sampling distribution of our estimators, and calculate standard errors for our estimates.

Here is the procedure: randomly draw 1000 datasets by resampling *with replacement* the observations from the data you generated in section 3.3[2]. Estimate the OLS coefficients for each bootstrapped dataset and store them in a matrix. Once you have the 1000 estimates for each coefficient in hand, estimate the standard errors by taking the standard deviation of the simulated sampling distribution for each coefficient. Again set the seed to 02138. Report the bootstrapped standard errors.

[1] This data is in the comma-separated value format, so use the `read.csv()` function to load into `R`

[2] You may find it useful to sample row numbers using the `sample()` function with the `replace = TRUE` argument, and then use your vector of row numbers to subset or reorder the complete dataframe. Note that some rows will be sampled multiple times, so a hypothetical dataset may contain the same observation several times.

Problem 4: Even More Probability By Simulation

We'd like everybody to attempt this problem, but don't expect everybody to get the answer. It will be especially challenging for those who are new to programming in R. Don't worry if you can't get the right answer; we'll be extremely lenient on grading. We just want everybody to think through strategies you might use to code this, and then look at the answer key next week to see one way to properly do it.

The idea of Monte Carlo simulation was originally inspired by a board/card game puzzle. Stanislaw Ulam wanted to estimate the chances of winning a game of Solitaire.[3] After being unable to tackle the problem analytically, he imagined the idea of repeatedly playing a large number of games and simply counting the number of wins to obtain an approximation to the true answer.

We're going to do something similar to a more *recent* game to further illustrate the power of Monte Carlo simulations where analytical calculations are infeasible.[4] Zombie Dice is a press-your-luck dice game similar to Farkle or Yahtzee.[5] Players play as zombies and roll special six-sided dice with three symbols: "brains," "shotguns," and "footprints." The goal is to score as many brains as possible while avoiding rolling three shotguns during their turn.[6]

Each player starts their turn by drawing three dice randomly without replacement from a bag of 13 color-coded six-sided dice. 6 dice are green, 4 are yellow, and 3 are red. Table 1 gives the distribution of symbols on each die. Green dice are "easy" (more brains) while red dice are "hard" (more shotguns).

Die Type	# of Brains	# of Shotguns	# of Footprints
Green	3	1	2
Yellow	2	2	2
Red	1	3	2

Table 1: Distribution of symbols on Zombie Dice

The player then rolls these dice. Any dice that come up as"brains" are set aside as points (they don't go back in the bag). Any dice that come up as shotguns are also set aside (and don't go back into the bag).

If the player has at any point rolled a total of three or more shotguns on their turn (across all of their rolls), they are "out" and score zero points. If the player is still "alive," after rolling they may choose to either bank their points and score the number of brains that they have rolled in total or "press-their-luck" and roll again.

[3]See Roger Ekhart. (1987). "Stan Ulam, John von Neumann, and the Monte Carlo Method." *Los Alamos Science*. `http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9068`.

[4]Here, because of the recursive structure of the game and conditional stopping rules, analytical computation of expected winnings is *very* complex

[5]For the offical rules, see the Steve Jackson Games website `http://www.sjgames.com/dice/zombiedice/img/ZDRules_English.pdf`

[6]Note that this game *has* been studied been studied by actual academics – see Cook and Taylor "Zombie Dice: An Optimal Play Strategy" `http://arxiv.org/abs/1406.0351`

If the player chooses to roll again, they take any dice that came up as "footprints" and add dice randomly from the bag so that they have a total of three dice to roll (footprints and new dice combined). A player will always roll exactly 3 dice (so if they have three footprints, they draw no new dice from the bag). They roll these dice and repeat the above procedure until they are either "out" or choose to bank their points.[7]

4.1

Consider the following strategy when deciding whether to stop and score:

1. If there are less than 3 dice left in the bag, stop.
2. If you have rolled a total of two shotguns, stop.
3. Otherwise keep rolling

Use Monte Carlo simulation to calculate the expected number of points that would be scored by a player using this strategy. Set the random seed to 02138 at the beginning of the problem. Use a lot of iterations since precision is important for this answer – $50,000$ is good.

4.2

Now consider a more complex strategy:

1. If there are less than 3 dice left in the bag, stop.
2. If you have rolled a total of one shotgun, stop if you've scored 5 or more brains.
3. If you have rolled a total of two shotguns, stop if you've scored 1 or more brains.
4. Otherwise keep rolling

Use Monte Carlo simulation to calculate the expected number of points that would be scored by a player using this strategy (again, setting the seed at the beginning to 02138 and running $50,000$ iterations).

5

Please submit all your code for this assignment as a .R file. Your code should be clean, commented, and executable without error.

[7] There are rules for refreshing the bag of dice if they run out, but we will ignore these for simplicity – the stopping rules you'll look at will never require refreshing the dice