Problem set 1: *due Feb. 15 at 11:59 PM*

Instructions

This problem set has 3 questions. Please submit your solutions through GitHub Classroom. In this repository you will find three `.jl` files, one for each question. Please report your answers by editing the `answers.md` file (Markdown format). In some questions, I will ask you to explain and discuss your results; please include any textual answers in this file.

To solve this problem set, you will need to install packages `Random`, `ForwardDiff`, `QuantEcon`, and `Distributions`.

Grading

For this first problem set, I will grade your submission only based on correctness of your code and textual answers. I will provide feedback on the readability and organization of your code and, if applicable, any suggestions on how to improve. Starting from the next problem set, your grade will also take into account readability and organization.

Late submissions will incur a penalty of 20% for each day it is late. I will grade your latest commit you pushed to the repository. If you unintentionally push new content after the deadline, please let me know as soon as possible.

Question 1: Variance estimation and numerical precision

For a given random variable Y, we can estimate its population mean $\mathbb{E}[Y]$ by applying the *analogy principle*, this is, we calculate the sample analog of that population measure:

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Similarly, for the population variance $\mathbb{V}[Y]$, we have the sample analog estimator

$$s^2 = \frac{1}{N}\sum_{i=1}^{N} (y_i - \bar{y})^2 \qquad (1)$$

We can manipulate this expression as follows:

$$\begin{aligned} s^2 &= \frac{1}{N}\sum_{i=1}^{N} \left(y_i^2 - 2y_i\bar{y} - \bar{y}^2\right) \\ &= \frac{1}{N}\left(\sum_{i=1}^{N} y_i^2 - 2N\bar{y}^2 - N\bar{y}^2\right) \\ &= \frac{1}{N}\left(\sum_{i=1}^{N} y_i^2 - N\bar{y}^2\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \bar{y}^2 \qquad (2) \end{aligned}$$

Thus, we have at least two algebraically-equivalent expressions to calculate s^2: equations (1) and (2). In this question, you will estimate the variance of a random sample using both expressions and compare the results.

In `question1.jl`, you will find a few lines to help you get started. This code will import the packages we need to generate random numbers in a reproducible way (by setting a random seed). Then, it will assemble an array of 10 million observations randomly drawn from a uniform distribution between $10,000,000$ and $10,000,001$.

For this question, do the following:

1. Calculate the sample mean and store it in variable `y_mean`
2. Calculate the variance using equation (1) and store it in variable `s2_eq1`
3. Calculate the variance using equation (2) and store it in variable `s2_eq2`
4. Compare the results from steps 2 and 3 with each other and with the true variance (you don't have to code the true variance calculation. Look it up if you don't remember the variance of uniform distributions).
    1. Are the estimated value close to each other?
    2. If not, which one is closest to the true variance? Why are the results different?

Question 2: Logit demand and numerical derivatives

The logit model is widely used for discrete demand models. In this question, we will explore a simple version of this model with only two products and no product characteristics.

Consumers have 3 options:

- Buy one car of model A with price $p_a = 2.5$ (thousand dollars)
- Buy one car of model B with price $p_b = 5.2$ (thousand dollars)
- Not to buy a car, i.e., the outside option

This is a *random utility model*. If consumers don't buy a car, they get a utility level normalized to 0. If consumer i buys car k, they get

$$u_{ik} = -\alpha p_k + \nu_{ik},$$

where α is a model parameter (it's the marginal utility of income) and ν_{ik} is the idiosyncratic taste of consumer i for product k. Assuming ν_{ik} is distributed in the population following a Type I Extreme Value distribution, we can derive a closed-form expression for the equilibrium demand for product k:

$$s_k\left(\vec{p}\right) = \frac{e^{-\alpha p_k}}{1 + \sum_{j=1}^{2} e^{-\alpha p_j}}$$

where $\vec{p}$ is the price vector. The demand for the outside option is given by $s_0 = 1 - s_A - s_B$

In this question, $\alpha = 0.25$. You are going to estimate how demand shares are affected by small price changes by calculating derivatives of the form ds_k/dp_k.

Here are the steps:

1. Define a `share` function and use it to calculate the shares of products A and B
2. Using finite forward differences, calculate
    1. Own-price derivatives ds_A/dp_A and ds_B/dp_B
    2. Cross-price derivates ds_A/dp_B, ds_B/dp_A
3. Then, using the `ForwardDiff` package, calculate own- and cross-price derivatives using automatic differentiation.

- *Tips: Remember you will need to code your `share` function in a way that `ForwardDiff` can understand (i.e., you need to convert mathematical operators into function calls). Also, if your function takes array as argument (it's a good idea), you should use `ForwardDiff.jacobian` instead of `ForwardDiff.gradient`.*

4. Calculate these derivatives with closed-form analytic expressions and compare them to your previous results (it is a good exercise to derive these expressions yourself, but you don't need to do it in this problem set):
   - $ds_k/dp_j = \alpha s_k s_j$ if $k \neq j$
   - $ds_k/dp_j = -\alpha s_k(1 - s_k)$ if $k = j$

Question 3: Expected profits, numerical integration, and functional programming

A profit-maximizing firm faces a demand curve given by $P(q) = a - bq$ and has a cost function given by $C(q) = cq$. a and c are fixed parameters, but b is a random variable following distribution $\Gamma(\alpha, \theta)$. The firm chooses q after it observes the realized value of b.

We are interested in the expected value of q, which is a nonlinear function of b.

1. Write a function called `profit_max_q` that returns the numerical optimal quantity given a set of parameters `(a, c, α, θ, method, n)`, where `method` is a string that takes on a value of `"mc"` or `"quad"` and determines whether you integrate using Monte Carlo or quadrature methods, and `n` is the number of Monte Carlo draws or quadrature nodes. Make sure your code is type-stable by using the code introspection macros (e.g. `@code_warntype`)
   - For Monte Carlo integration, you can code it yourself or use the function `QuantEcon.quadrect`
   - For Gaussian quadrature, use `QuantEcon.qnwdist` and pass a distribution gamma as input using `Distributions.Gamma`
   - Use limits of integration (for Monte Carlo) and number of nodes (for quadrature) that you judge adequate for the problem
2. Calculate $E[q]$ using Monte Carlo and Gaussian quadrature for parameters: $a = 85, c = 8, \alpha = 5, \theta = 1$.