

BasicProblems

May 26, 2017

0.1 Starter Problems

Note: These 4 problems are from https://lectures.quantecon.org/jl/julia_by_example.html

Strang Matrix Problem Use Julia's array and control flow syntax in order to define the $N \times N$ Strang matrix:

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix}$$

i.e. a matrix with -2 on the diagonal, 1 on the off-diagonals, and 0 elsewhere.

Factorial Problem* Using a `for` loop, write a function `my_factorial(n)` that computes the n th factorial. Try your function on integers like `15`, and then use `BigInt` inputs like `big(100)`. Make your function's output type match the input type for n .

Binomial Problem* A random variable $X \sim \text{Bin}(n, p)$ is defined the number of successes in n trials where each trial has a success probability p . For example, if $\text{Bin}(10, 0.5)$, then X is the number of coin flips that turn up heads in 10 flips.

Using only `rand()` (uniform random numbers), write a function `binomial_rv(n, p)` that produces one draw of $\text{Bin}(n, p)$.

Monte Carlo π Problem* Use random number generation to estimate π . To do so, mentally draw the unit circle. It is encompassed in the square $[-1, 1] \times [-1, 1]$. The area of the circle is $\pi r^2 = \pi$. The area of the square is 4. Thus if points are randomly taken evenly from $[-1, 1] \times [-1, 1]$, then the probability they land in the circle ($x^2 + y^2 \leq 1$) is $\frac{\pi}{4}$. Use this to estimate π .

0.2 Integration Problems

These problems integrate basic workflow tools to solve some standard data science and scientific computing problems.

Timeseries Generation Problem* An AR1 timeseries is defined by

$$x_{t+1} = \alpha x_t + \epsilon_{t+1}$$

where $x_0 = 0$ and $t = 0, \dots, T$. The shocks ϵ_t are i.i.d. standard normal ($N(0, 1)$), given by `randn()`. Using $T = 200$

- 1) $\alpha = 0$
- 2) $\alpha = 0.5$
- 3) $\alpha = 0.9$

use `Plots.jl` to plot a timecourse for each of the parameters. Label the lines for the values of α that generate them using the `label` argument in `plot`.

Regression Problem

```
In [ ]: ##### Prepare Data For Regression Problem

X = rand(1000, 3)                # feature matrix
a0 = rand(3)                     # ground truths
y = X * a0 + 0.1 * randn(1000); # generate response

# Data For Regression Problem Part 2
X = rand(100);
y = 2X + 0.1 * randn(100);
```

Given an $N \times 3$ array of data (`randn(N, 3)`) and a $N \times 1$ array of outcomes, produce the data matrix X which appends a column of 1's to the front of the data matrix, and solve for the 4×1 array β via $\beta X = b$ using `qrifact`, or `\`, or [the definition of the OLS estimator](#). (Note: This is linear regression).

Compare your results to that of using `llsq` from `MultivariateStats.jl` (note: you need to go find the documentation to find out how to use this!). Compare your results to that of using ordinary least squares regression from `GLM.jl`.

Regression Problem Part 2 Using your OLS estimator or one of the aforementioned packages, solve for the regression line using the (X, y) data above. Plot the (X, y) scatter plot using `scatter!` from `Plots.jl`. Add the regression line using `abline!`. Add a title saying "Regression Plot on Fake Data", and label the x and y axis.

Logistic Map Problem The logistic difference equation is defined by the recursion

$$b_{n+1} = r * b_n(1 - b_n)$$

where b_n is the number of bunnies at time n . Starting with $b_0 = .25$, by around 400 iterations this will reach a steady state. This steady state (or steady periodic state) is dependent on r . Write a function which plots the steady state attractor. This is done as follows:

- 1) Solve for the steady state(s) for each given r (i.e. iterate the relation 400 times).

- 2) Calculate “every state” in the steady state attractor. This means, at steady state (after the first 400 iterations), save the next 150 values. Call this set of values $y_s(r)$.
- 3) Do steps (1) and (2) with $r \in (2.9, 4)$, $dr = .001$. Plot r x-axis vs $y_s(r)$ =value seen in the attractor) using Plots.jl. Your result should be the [Logistic equation bifurcation diagram](#).