PSTAT 10 Final Exam Solutions

July 28, 2022

Name:

Section (circle one):

- Olivier MW 8:00AM
- Jeff MW 9:30AM
- Jeff MW 12:30PM

Instructions

You have 80 minutes to complete 6 problems, some of which contain subproblems.

You are allowed one sheet, two sides of handwritten notes.

If you run out of space for your code, write your code on scratch paper and clearly indicate which problem it corresponds to. You can always ask for more scratch paper.

Please make sure your handwritten code is legible. I cannot give credit to code that is unreadable.

Score: _____ out of 36

Problem 1: Last Negative (3 pts)

Write a function `last_negative` that takes a numeric vector `v` and returns the last element that is less than zero.

```
last_negative(c(1, -9, 0, 6, -7, 0))
```

```
## [1] -7
```

```
last_negative(c(-1, -4, 0, 2))
```

```
## [1] -4
```

```
last_negative <- function(v) {
  which_vec <- which(v < 0)
  v[which_vec[length(which_vec)]]
}
```

Problem 2: Primes (6 pts)

The function `is_prime(x)` returns `TRUE` if `x` is a prime number and `FALSE` otherwise. It is a vectorized function. You do not need to implement `is_prime`; assume it is already implemented and available.

---

1. Write code that outputs the number of prime numbers below 100. Your code does not have to be a function, but should output the answer to the console when executed. (3 pts)

```
sum(is_prime(1:100))
```

```
## [1] 25
```

2. Write code that outputs the sum of the prime numbers whose square is below 2,000. (3 pts)

```
p <- (1:100)[is_prime(1:100)]
p[p^2 < 2000]
```

```
##  [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43
```

Problem 3: Exponential distribution (8 pts)

How long will a lightbulb last? How long until a bird arrives at my bird feeder?

A distribution that models such questions is the *exponential distribution*. We write $X \sim \text{Expo}(\lambda)$ if X is exponentially distributed with *rate* λ; the parameter λ is the rate of failure of a lightbulb or the rate that birds arrives at my bird feeder. X is a random variable with support $(0, \infty)$.

R has built-in functions for the exponential distribution. They are:

- `dexp(x, rate = 1)`
- `pexp(x, rate = 1)`
- `rexp(x, rate = 1)`

The waiting time until the Old Faithful geyser erupts is exponentially distributed with rate 0.014 (eruptions/minute). Determine the following.

1. Is the exponential distribution discrete or continuous? (Circle one) (1 pt)

- Discrete
- **Continuous**

2. The probability that the geyser erupts after the next 30 minutes, but within the next 60 minutes. (2 pts)

```
pexp(60, rate = 0.014) - pexp(30, rate = 0.014)
```

```
## [1] 0.2253363
```

3. The probability of waiting longer than 90 minutes for the next eruption. (2 pts)

```
1 - pexp(90, rate = 0.014)
```

```
## [1] 0.283654
```

4. Use 10,000 replications to determine the expected waiting time until the next eruption. (3 pts)

```
mean(rexp(10000, rate = 0.014))
```

```
## [1] 71.37969
```

Problem 4: Gambler (5 pts)

A gambler plays the following game.

- The gambler starts with $10.
- Each round, a coin with probability 5/6 of landing heads is flipped.
- If heads, $2 is added to the gambler's account.
- If tails, the gambler's account is cut in half.
- The game lasts for n rounds.

Part A

Write a function `gamble(n)` that returns the amount in the gambler's account at the end of n rounds, where n is guaranteed to be a positive integer. (3 pts)

```
gamble <- function(n) {
  account <- 10
  for (i in seq_len(n)) {
    coin <- rbinom(1, 1, 5/6)
    if (coin) {
      account <- account + 2
    }
    else {
      account <- account / 2
    }
  }
  return(account)
}
```

Part B

Write code that estimates, using 10,000 replications, the *expected amount* in the gambler's account after playing the game for 15 rounds. (2 pts)

```
mean(replicate(10000, gamble(15)))
```

```
## [1] 17.2955
```

Problem 5: astro (8 pts)

A relational database called `astro` contains three tables: `star`, `planet`, and `moon`. These tables contain data describing celestial objects in the observable universe.

A small portion of data in these tables is provided below. I emphasize that these are not all of the records, only a small sample for each table.

Table 1: star

StarId	Name	Mass	Radius
1	Sun	1.00	1.00
2	Alpha Centauri A	1.07	1.22

Table 2: planet

PlanetId	PrimaryId	Name	Mass	Radius
3	1	Earth	1.00	1.00
5	1	Jupiter	317.80	11.21
6	1	Saturn	95.16	9.45
19	2	Proxima b	1.07	1.30

Table 3: moon

MoonId	PrimaryId	Name	Mass	Radius
1	3	The Moon	0.012	0.27
13	5	Ganymede	0.025	0.41
20	6	Titan	0.023	0.40

We know that the Earth revolves around the Sun. The Sun is referred to as the Earth's *primary*. Similarly, the Earth is the Moon's *primary*. This data is reflected in `astro`. Take some time to understand it.

Note: I purposefully left out the units of measurement in order to simplify the problem. In case you are curious, the measurements of `planet` and `moon` are given in *Earth units* while those of `star` are given in *Solar units*.

Answer the questions on the next page.

Part A: (2 pts)

Write down the primary key and foreign keys of each table. If a key does not exist, write "None".

1. `star`

- Primary key: **StarId**
- Foreign key(s): **None**

2. `planet`

- Primary key: **PlanetId**
- Foreign key(s): **PrimaryId**

3. `moon`

- Primary key: **MoonId**
- Foreign key(s): **PrimaryId**

Write a **single** SQL query that retrieves the data specified below. For full credit, retrieve only the fields specified and **write only SQL** and nothing else. Using more than one query to solve the problem will be awarded half credit. **Do not include** `dbGetQuery` and other R code. It is not SQL.

Part B: (2 pts)

Retrieve the `StarId`, `Name`, and `Radius` of all stars with radius greater than 12.5.

```
select StarId, Name, Radius from star where Radius > 12.5
```

Part C: (2 pts)

Retrieve the `Name` of the moon, `Name` of its planet, and `Mass` of its planet for all moons orbiting a planet with mass less than 20. Result should have the three fields described.

```
-- Joining in any order is okay (e.g. planet first and then moon)
-- This is how you do comments in SQL, by the way.
select m.Name, p.Name, p.Mass from moon m
  inner join planet p on m.primaryid = p.planetid
  where p.mass < 20
```

Part D: (2 pts)

Retrieve the `Name` of the planet, `Name` of its star, and its number of moons for all planets that have at least one moon. Result should have the three fields described.

```
select p.Name, s.Name, count(*) from planet p
  inner join star s on p.primaryid = star.starid
  inner join moon m on m.primaryid = p.planetid
  group by p.planetid
    having count(*) >= 1
```

Problem 6: Emails (6 pts)

I have two inboxes, labeled **A** and **B**. The `email` tibble gives how many emails arrive in each inbox for 360 days of the year. The variables give the inbox, month, day, and number of new emails.

```
email
```

```
## # A tibble: 720 x 4
##    Inbox Month   Day Count
##    <fct> <int> <int> <int>
##  1 A         1     1     2
##  2 B         1     1     2
##  3 A         1     2     3
##  4 B         1     2     1
##  5 A         1     3     3
##  6 B         1     3     3
##  7 A         1     4     4
##  8 B         1     4     2
##  9 A         1     5     3
## 10 B         1     5     1
## # ... with 710 more rows
```

Here is a random sample of 10 observations from `email`.

```
set.seed(100)
email |> slice_sample(n=10)
```

```
## # A tibble: 10 x 4
##    Inbox Month   Day Count
##    <fct> <int> <int> <int>
##  1 B        12    27     4
##  2 A         9    12     2
##  3 B         6    29     2
##  4 B        11    12     2
##  5 B         8    25     2
##  6 B         9    18     0
##  7 B         2    19     1
##  8 A         1     4     4
##  9 A         4     2     6
## 10 A         5    30     3
```

1. Since I am such a hard worker, I even answer emails on my birthday, January 28th. Write code to determine how many new emails arrived on my birthday, in either inbox. Your code must produce the provided output, which is a tibble with 1 row and 3 columns. (2 pts)

```
(email1 <- email |>
             group_by(Month, Day) |>
             summarize(TotalCount = sum(Count)) |>
             filter(Month == 1, Day == 28))
```

```
## # A tibble: 1 x 3
## # Groups:   Month [1]
```

```
##   Month   Day TotalCount
##   <int> <int>      <int>
## 1     1    28          5
```

2. Create a tibble `email2` which gives the average number of emails for each month for each inbox. The output is given, arranged by increasing `Month` for clarity. Your output need not be sorted as shown.

```
(email2 <- email |>
            group_by(Inbox, Month) |>
            summarize(MonthAvg = mean(Count)))
```

```
## # A tibble: 24 x 3
## # Groups:   Inbox [2]
##    Inbox Month MonthAvg
##    <fct> <int>    <dbl>
##  1 A         1     2.9
##  2 A         2     3.27
##  3 A         3     2.93
##  4 A         4     2.83
##  5 A         5     2.7
##  6 A         6     3.17
##  7 A         7     2.83
##  8 A         8     2.7
##  9 A         9     3.17
## 10 A        10     3.3
## # ... with 14 more rows
```

3. Using `email2` create the ggplot shown below. (2 pts)

```
ggplot(email2, aes(x = Month, y = MonthAvg)) +
  geom_line(size = 0.7) +
  facet_wrap(~ Inbox)
```