Homework 3

Spencer Matthews

11/11/2021

Problem 1

Consider a discrete-time single-type branching process X_n with a Poisson progeny distribution and the corresponding generating function $Q(s) = E(s^{X_1}) = e^{\lambda(s-1)}$.

(a) Write a function to simulate X_n starting from $X_0 = 1$.

```
branch_sim <- function(n, lambda, N, x0 = 1) {
  sims <- integer(N)
  for (i in 1:N) {
    steps <- integer(n+1)
    steps[1] <- x0
    for (j in 1:n) {
      steps[j+1] <- sum(rpois(n = steps[j], lambda = lambda))
    }
    sims[i] <- steps[n+1]
  }
  sims
}
```

(b) Estimate $q_n = P(X_n = 0)$ for $n = 5, 15, 30$ and $\lambda = 1.4$ by simulating the branching process $N = 1000$ times.

```
set.seed(16)
n_of_5 <- branch_sim(n = 5, lambda = 1.4, N = 1000)
mean(n_of_5 == 0)
```

```
## [1] 0.45
```

Based on our simulation, we have $q_5 = 0.45$, or a 45% probability of the branching process dying out on or before the 5th step.

```
set.seed(16)
n_of_15 <- branch_sim(n = 15, lambda = 1.4, N = 1000)
mean(n_of_15 == 0)
```

```
## [1] 0.485
```

Based on our simulation, we have $q_{15} = 0.485$, or a 48.5% probability of the branching process dying out on or before the 15th step.

```r
set.seed(16)
n_of_30 <- branch_sim(n = 30, lambda = 1.4, N = 1000)
mean(n_of_30 == 0)
```

```
## [1] 0.48
```

Based on our simulation, we have $q_{30} = 0.48$, or a 48% probability of the branching process dying out on or before the 30th step.

(c) Compare your simulation-based estimate with a numerical solution that uses the Newton's method to solve equation $Q(s) = s$. Recall that Newton's method for solving an algebraic equation $f(x) = 0$ prescribes a recursion $x_{n+1} = x_n - f(x_n)/f'(x_n)$.

First, we will write functions to implement $Q(s) - s$ and $\frac{d}{dx}[Q(s) - s]$, and recursive function to implement Newton's method.

```r
qs <- function(s, lambda) {
  exp(lambda * (s-1)) - s
}

qs_derivative <- function(s, lambda) {
  lambda * exp(lambda * (s-1)) - 1
}

newtons_method <- function(n, lambda) {
  tmp <- n
  if (tmp == 0) return(0.4)
  else {
    val <- newtons_method(n - 1, lambda)
    val - qs(val, lambda)/qs_derivative(val, lambda)
  }
}
```

Now, we can implement these functions to get a numerical solution to find q_n. Here we will run Newton's method with 100 steps. Notice here that the n in this function is different than the n in our simulation in part (a), because here it is just the number of steps we take towards approximating the solution to the equation.

```r
newtons_method(100, 1.4)
```

```
## [1] 0.4889888
```

Thus, we see that the numerical solution for the extinction probability is 48.9%. We can see that this is quite close to the values we got in part (b), as we would expect it to be.

Problem 2

Suppose we are interested in estimating the mean of the progeny distribution μ of a discrete-time single-type branching process.

(a) One way of going about this task is to equate the number of particles present at time n, X_n, with the corresponding expectation $E(X_n|X_0 = 1)$. Show that the resulting estimate $\tilde{\mu}$ underestimates μ on average. In other words, prove that $E(\tilde{\mu}) \leq \mu$.

We know that $E[X_n] = \mu^n$ so creating an estimate we have $\tilde{\mu} = X_n^{1/n}$. We will now show that $E[\tilde{\mu}] \leq \mu$. By Jensen's inequality, we see the following

$$\begin{aligned} E[\tilde{\mu}] &= E[X_n^{1/n}] \\ &\leq E[X_n]^{1/n} \\ &= (\mu^n)^{1/n} \\ &= \mu \end{aligned}$$

Thus we have $E[\tilde{\mu}] \leq \mu$.

(b) Another method of moments uses equation $E(X_n) = E(X_{n-1})\mu$ to arrive at an estimate

$$\hat{\mu} = \begin{cases} \frac{X_n}{X_{n-1}} & \text{if } X_{n-1} > 0 \\ 1 & \text{if } X_{n-1} = 0 \end{cases}$$

Show that $E(\hat{\mu}) = \mu P(X_{n-1} > 0) + P(X_{n-1} = 0)$.

If we assume that X_{n-1} is a constant, we see that

$$\begin{aligned} E(\hat{\mu}) &= E\left[\frac{X_n}{X_{n-1}}\right] P(X_{n-1} > 0) + E[1]P(X_{n-1} = 0) \\ &= \frac{1}{X_{n-1}} E[X_n] P(X_{n-1} > 0) + P(X_{n-1} = 0) \\ &= \frac{1}{X_{n-1}} \mu E[X_{n-1}] P(X_{n-1} > 0) + P(X_{n-1} = 0) \\ &= \frac{\mu X_{n-1}}{X_{n-1}} P(X_{n-1} > 0) + P(X_{n-1} = 0) \\ &= \mu P(X_{n-1} > 0) + P(X_{n-1} = 0) \end{aligned}$$

Thus showing that $E(\hat{\mu}) = \mu P(X_{n-1} > 0) + P(X_{n-1} = 0)$

(c) Calculate $E(\hat{\mu})$ for the Poisson progeny distribution in the problem 1(a) using $\lambda = 1.1$ and $n = 20$.

We just showed above that $E(\hat{\mu}) = \mu P(X_{n-1} > 0) + P(X_{n-1} = 0)$, and so our first step in solving this problem will be to compute $P(X_{19} > 0)$ and $P(X_{19} = 0)$, which we will do using our function above.

```
set.seed(16)
p_eq_0 <- mean(branch_sim(n = 20, lambda = 1.1, N = 1000) == 0)
p_eq_0
```

```
## [1] 0.779
```

```
1 - p_eq_0
```

```
## [1] 0.221
```

Furthermore, we know that μ is equal to 1.1, because lambda is the mean of our Poisson progeny distribution. Thus, we have

$$E(\hat{\mu}) = \mu P(X_{n-1} > 0) + P(X_{n-1} = 0) = 1.1(0.211) + 0.779 = 1.011$$

So the expected value is 1.011, which again seems to be biased as lower than μ.

Problem 3

Download and install an `EpiEstim` R package. Use the provided `get_oc_data.R` script to download SARS-CoV-2 cases in Orange County, CA. Estimate changes in the effective reproductive R_t in Orange County. You may find this demo helpful to get you started: `https://cran.r-project.org/web/packages/EpiEstim/vignettes/demo.ht`

First, we will use the provided code to get the OC counts data.

```
first_date = as.Date("2020-03-30")
last_date = as.Date("2021-10-31")

oc_counts_data <- httr::GET(
  paste0(
    "https://services2.arcgis.com/LORzk2hk9xzHouw9/",
    "arcgis/rest/services/oc_covid_count/FeatureServer",
    "/0/query?where=1%3D1&outFields=*&outSR=4326&f=json"
  ),
  query = list(
    outFields = "*",
    where = "1=1"
  )
) %>%
  httr::content(as = "text", encoding = "UTF-8") %>%
  jsonlite::fromJSON(flatten = T) %>%
  magrittr::use_series(features) %>%
  as_tibble() %>%
  transmute(
    dates = as_date(as_datetime(attributes.date/1000)),
    I = attributes.dailycases_specimen,
    Region = "OC"
  ) %>%
  filter(dates >= first_date)  %>%
  filter(dates <= last_date) %>%
  arrange(dates)
```

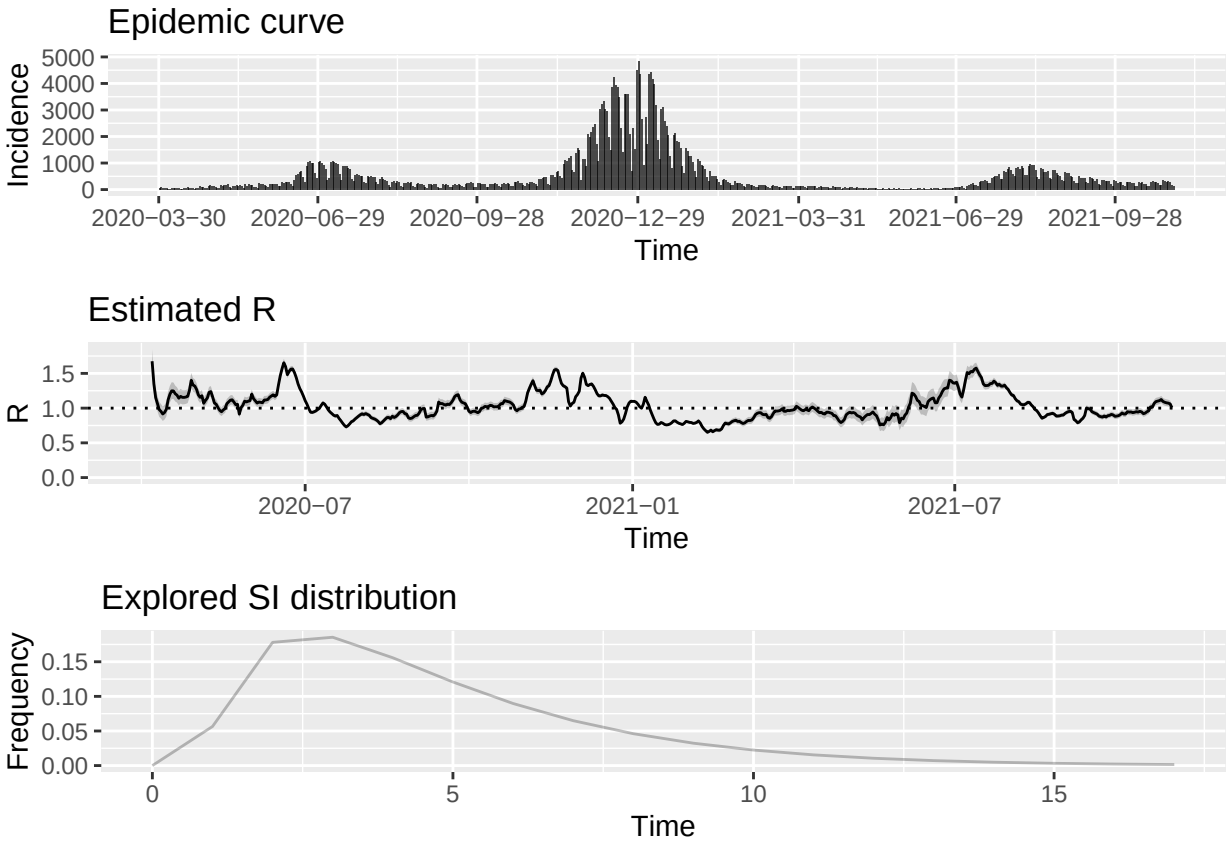
Now we can use the `EpiEstim` R package to estimate R_t over time. For this part, we will use a mean serial interval time of 4.7 with a standard deviation of 2.9. These are consistent with numbers reported in the following article:

> Nishiura, Hiroshi, Natalie M. Linton, and Andrei R. Akhmetzhanov. "Serial interval of novel coronavirus (COVID-19) infections." International journal of infectious diseases 93 (2020): 284-286.

```
estimates <- estimate_R(
  incid = oc_counts_data,
  method = "parametric_si",
  config = make_config(
    list(
      mean_si = 4.7,
      std_si = 2.9
    )
  )
)
```

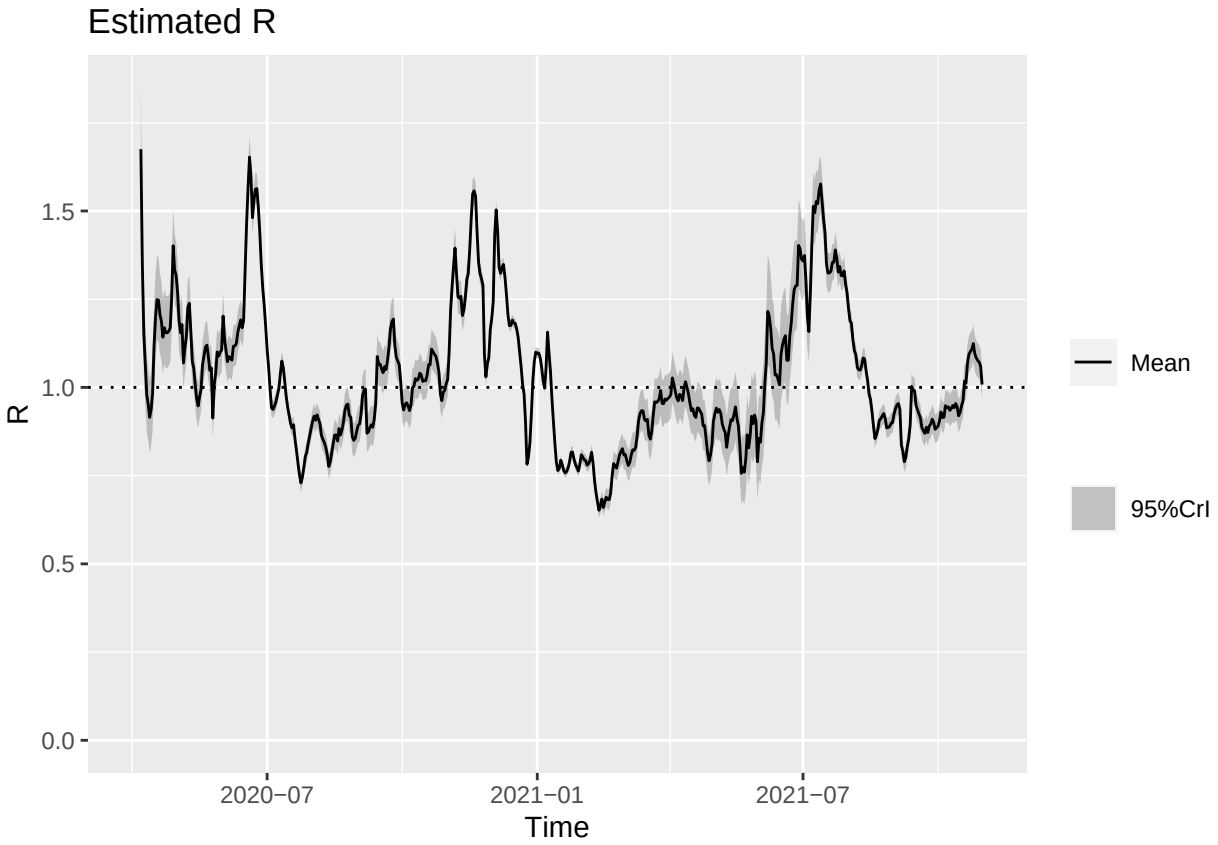
```
## Default config will estimate R on weekly sliding windows.
##     To change this change the t_start and t_end arguments.
```

```
plot(estimates, legend = FALSE)
```



Or, we could look at the just the estimated R_t.

```
plot(estimates, "R")
```



Based on this plot, we see that R_t has been quite variable over the course of the pandemic, with spikes corresponding to the times when there were large spikes in cases. I see why lots of people have used the `EpiEstim` R package because it is quite easy to use and produces pretty quick results with not a lot of effort. But at the same time, there are some issues with it that Isaac taught us in class so we need to be considerate when using it and reporting it.