Final R Project

36-350 – Statistical Computing

Week 11 – Spring 2021

Name: Jacky Liu

Andrew ID: jackyl1

Project Instructions (Read Carefully)

- There are 200 points possible for this project
- The dataset that you will examine has been provided by the CMU Libraries. It contains a listing of items (usually books, but not always, hence my use of the term "items") loaned out by the various libraries over a period of time.
- The dataset, available on Canvas in week-11 section, is `20200225_print-circulation_sample.csv`. The only thing I will say about the contents now is that some of the columns are best treated as factors and some as strings (that subsequently might need to be converted to other data types), so you should examine the data and try your best to preprocess all the input columns correctly. (Note: this process may be iterative, i.e., you might determine later that you need to alter how you process the input. That's fine. That's *normal* in the real world.)
- **To be clear**: there is generally no unique way of going about answering each question below. For instance, you may want to use base `R` sometimes, and `tidyverse` functions at other times. In the end, *I don't particularly care how you go about answering the questions, so long as you answer them correctly.* (Some of you may very well create more elegant solutions than what I have in the solution set. And that's good. Others will create coding monstrosities... but that's OK, as my attitude is that your coding will improve with practice. One cannot expect to leave a semester-long class with the same comfort level coding `R` as I have built up over 15 years of nearly continuous coding...)
- This project aims to test your knowledge. Therefore, **no code debugging or solution assistance will be provided.** Only clarifications on question wording can be provided during week-11 lectures or on Piazza.
- Typically, this project is submitted within 1-2 weeks. To accommodate everyone's needs, students will have more than 3 weeks to complete it. Please submit your project on Gradescope by **May 7th, 9PM EST**.
- There is 20% penalty applied to submissions that are made **after May 7th, 9PM EST and until May 7th, 11:59PM EST**.
- Submissions that are made **after May 7th, 11:59PM EST will not be graded**.
- Given you are provided one more week than the normal submission timeline, **no extensions will be provided under ANY circumstances.** `Students who have disability or require flexible submission timeline are encouraged to start early` and get advantage of the additional 1+ week that is added to the project submission timeline.

- Students are encouraged to submit their project earlier than the project deadline to avoid last-minute issues. Technical issues; including internet problems, family emergencies, or knitting problems **WILL NOT be excused** from the project submission deadline.
- You must submit **your own** project as a knitted PDF file on Gradescope. Students are expected to submit their project themselves without assistance from the course instructor or the TAs.

Question 1

(10 points)

Download the data file and read it into `R`. As stated above, how you go about doing that is up to you. Treat dates (including years!) and times as characters for now. One hint: if you use the `tidyverse`, then what you read in will be in `tibble` format. You may want to convert your data to a data frame format to avoid headaches later. However, if you are comfortable with tibbles, then keep the data in `tibble` format.

```
data = read.csv("20200225_print-circulation_sample.csv")
```

Question 2

(10 points)

Not all of the columns are useful. First, use `summary()` or a similar function to determine if any columns are wholly uninformative. If any are, eliminate them. Then check for redundancy: if any column is redundant, eliminate it as well. Display the dimension of your data frame when you are finished with this first round of processing.

```
summary(data)
```

```
## Loans..Not.In.House. LC.Classification.Top.Line Loan.Date
## Min.  :1 Unknown: 8182 3/21/18: 462
## 1st Qu.:1 ML410 : 625 8/26/19: 267
## Median :1 QA76.9 : 531 8/27/18: 244
## Mean :1 QA76.5 : 391 9/13/18: 242
## 3rd Qu.:1 QA76.73: 352 1/13/20: 223
## Max.  :1 (Other):49502 1/15/19: 220
## NA's : 5 (Other):57930
## Loan.Time Return.Date Patron.Group
## 8:00:00 :11806 :11453 Carnegie Mellon Grad Students:18511
## 15:13:00: 10 5/14/18 : 376 Carnegie Mellon Undergrads :16995
## 14:43:00: 9 5/13/19 : 305 Carnegie Mellon Faculty : 8687
## 10:16:00: 8 12/13/18: 297 Carnegie Mellon Staff : 5632
## 12:07:00: 8 5/21/18 : 292 Inter Library Loan : 2578
## 13:00:41: 8 5/10/18 : 285 Qatar Undergrads : 1848
## (Other) :47739 (Other) :46580 (Other) : 5337
## Material.Type Title
## Book :56161 Calculator, Qatar Library : 1043
## Music Score: 3144 Computer systems : a programmer's perspective / : 384
## Thesis : 53 Biological science / : 330
## Unknown : 230 Musical acoustics / : 293
## Introduction to engineering and the environment /: 198
## Essentials of management information systems / : 100
## (Other) :57240
```

```
## Begin.Publication.Date Publisher Language.Code
## 2013 : 4312 : 4803 eng :55645
## 2017 : 2418 Cambridge University Press: 2130 zxx : 1057
## 2016 : 2352 Oxford University Press : 2018 : 987
## : 2340 Pearson : 1052 N/A : 641
## 2012 : 2010 Routledge : 1008 ger : 355
## 2011 : 1997 Wiley : 860 fre : 220
## (Other):44159 (Other) :47717 (Other): 683
## Subjects
## : 7061
## Computer systems.; Computers.; Telecommunication.; User interfaces (Computer
systems): 370
## Biology--Textbooks.; Biology.; Textbooks.  : 324
## Music--Acoustics and physics.  : 294
## Environmental engineering.  : 201
## Management information systems.  : 103
## (Other) :51235
## LC.Classification.Top.Line.1 Library.Code
## Unknown: 8174 :11806
## ML410 : 625 ENGR-SCI: 8294
## QA76.9 : 531 HUNT :37120
## QA76.5 : 391 MELLON : 195
## QA76.73: 352 OFFSITE : 20
## (Other):49510 QATAR : 2152
## NA's : 5 SEI : 1
```

Loans Not In House seems to have 1's for all its values, so therefore it doesn't seem useful and we;ll eliminate it.

```
data = subset(data, select = -c(1))
```

Question 3

(10 points)

What is the range of times over which items were loaned out in this data sample? Display the date and time of the first loan, the last loan, and the difference in time between them, in days. This involves concatenating the contents of two columns and converting the concatenated quantity to something you can use. You need not add the concatenated quantity to the data frame; just use it to answer the initial question. Assume all times are local time for CMU (even if the loan occurred in, e.g., Qatar). Hint: you might want to process the dates before concatenation, particularly because some of the dates are from the 1900's (so a one-size-fits-all substitution of, e.g., "20" in the year field will not work for all the data). Note that I found a time difference of 9019.486 days.

```
frame = data[c(2,3)]
frame$datetime <- as.POSIXct(paste(frame$Loan.Date, frame$Loan.Time), format="%m/%d/%y %H:%M:%S", tz =
max(frame$datetime) - min(frame$datetime)
```

```
## Time difference of 9019.444 days
```

Question 4

(10 points)

Using your loan date and time from Q3 and the date from the `Return Date` column, create a histogram showing the length of time that items are loaned out, in *days*. (Limit your histogram to x-axis values 0 to 1000, and define the breaks to be every 10 days. Label the x-axis "Loan Duration (Days)". If using base `R`, set `main=NULL`. Color is up to you.) Don't worry about the fact that there is only a return date, and not a return time. Ignore items that had yet to be returned at the time the dataset was created (check for empty strings!), and note that the number of days will not be an integer, because you are incorporating the loan time. Filter out any data where the item was returned before it was loaned out(!). Note: no item was returned during the 1990s; this simplifies processing.

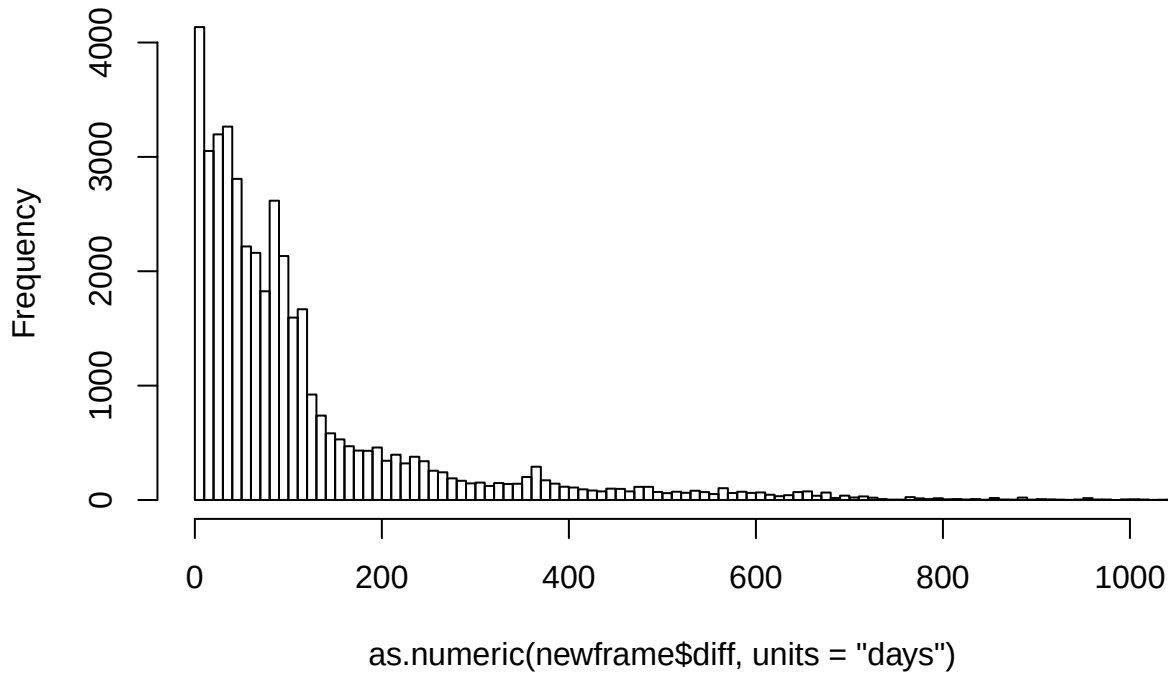
```
# create new dataframe with only Return.Date and Loan datetime
newframe = data[c(4)]
newframe$datetime <- as.POSIXct(paste(frame$Loan.Date, frame$Loan.Time), format="%m/%d/%y %H:%M:%S", tz

# filter out blank rows
newframe = newframe[newframe$Return.Date != "",]

# convert return date to posixct so we can subtract
newframe$Return.Date = as.POSIXct(newframe$Return.Date, format="%m/%d/%y", tz = "EST")

# remove rows where it was removed before it was loaned out
newframe = newframe[newframe$Return.Date > newframe$datetime,]

# calculate difference and plot
newframe$diff = difftime(newframe$Return.Date, newframe$datetime, units="days")
hist(as.numeric(newframe$diff, units="days"), xlim = c(0,1000), main=NULL, breaks=10*8*10)
```



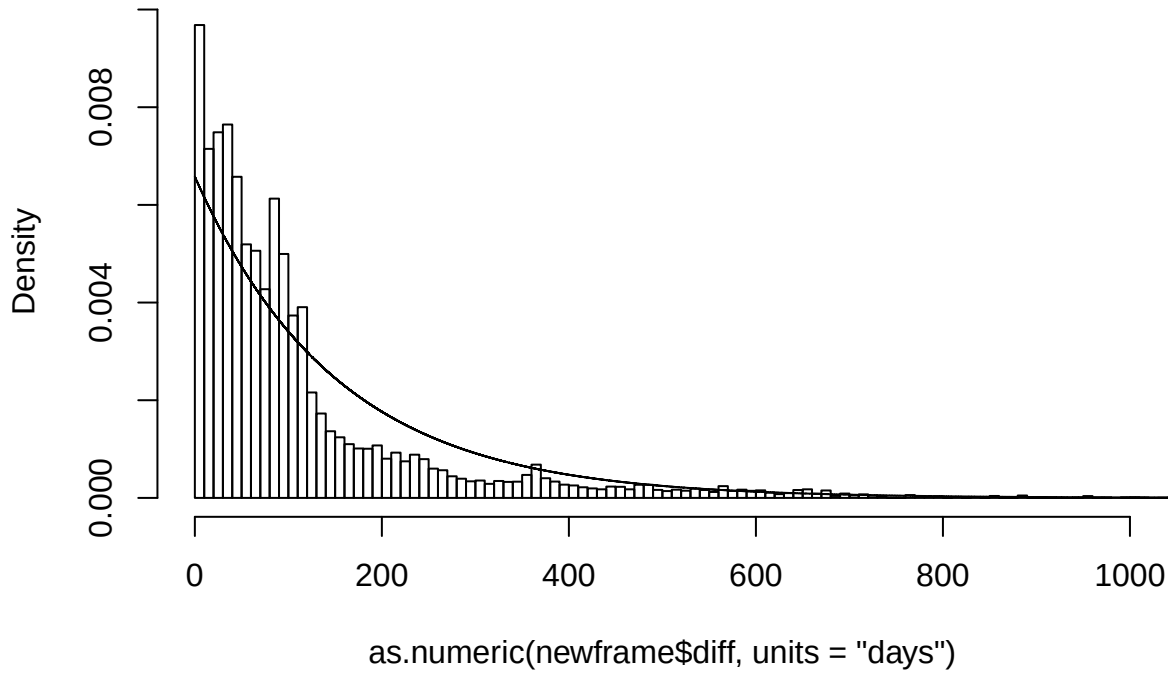
```
loandurations = as.numeric(newframe$diff, units="days")
```

Question 5

(10 points)

The data displayed in your histogram in Q4 appear, at first glance, to be exponentially distributed, at least approximately. Fit an exponential distribution to these data using an appropriate optimizer. (You need not include the gradient here.) Display the optimized value of the `rate` parameter. Redisplay your histogram from Q4 with the optimized exponential pdf superimposed. (If you cannot see it: are you creating a frequency histogram, or a density histogram?) Don't expect the model to be a "good" one. Hint: if you have a hard time finding the optimum value, try plotting a few times with lines superimposed with different values of the `rate` parameter. This will help build your intuition.

```
mydata = as.numeric(newframe$diff, units="days")
negloglike = function(x,theta) {
  sum(-dexp(x=x,rate=theta,log=T))
}
mle = optimize(f=negloglike,x=mydata,interval = c(0,2))$minimum
hist(as.numeric(newframe$diff, units="days"), prob=TRUE, xlim = c(0,1000), main=NULL, breaks=10*8*10)
x = seq(min(mydata), max(mydata), by=0.01)
lines(x, dexp(x, rate=mle))
```



```r
cat("Rate:", mle, "\n")
```

```
## Rate: 0.006577748
```

Question 6

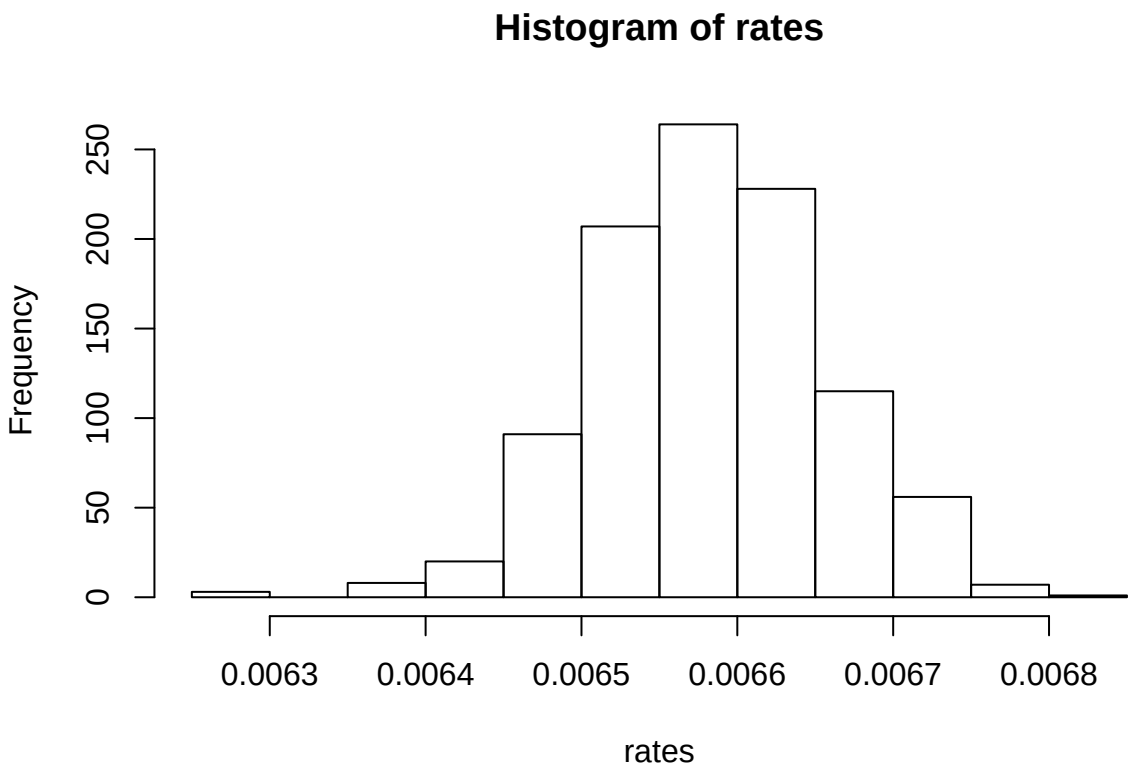
(10 points)

Estimate the uncertainty for the `rate` parameter via bootstrapping. Display a histogram showing the estimated values of `rate` and display the 2.5% and 97.5% quantiles. You have a lot of data, so the difference between quantiles should be small.

```r
B=1000
rates = vector(length=B)
for (i in 1:B) {
  indices = sample(length(mydata), length(mydata), replace=TRUE)
  mle = optimize(negloglike, interval=c(0,2), x=mydata[indices])
  rates[i] = mle$minimum
}
rates = sort(rates)
quantiles = c(rates[25], rates[975])
cat("2.5% and 97.5% quantiles respectively are:", quantiles)
```

```
## 2.5% and 97.5% quantiles respectively are: 0.006442682 0.006723907
```

```r
hist(rates)
```

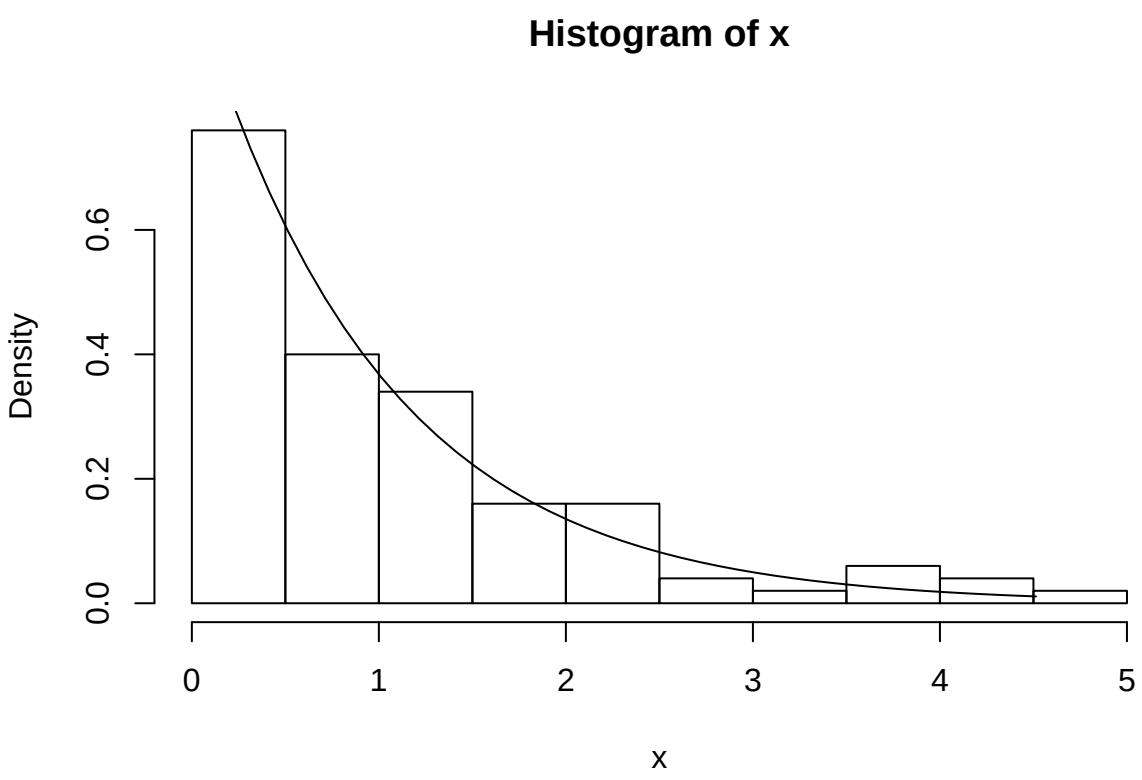


Question 7

(10 points)

Using *inverse-transform sampling*, sample 100 data from the optimal exponential distribution. Note: `R` defines the exponential distribution to be $f(x) = re^{-rx}$, where r is the `rate` parameter. Display a histogram of your sampled data, and overlay the optimal exponential distribution, like you did in Q5.

```
set.seed(666)
u = runif(100)
x = -log(1-u) # inverse CDF of the exponential distribution
hist(x, probability = TRUE)
x = seq(min(x), max(x), by=0.1)
lines(x, dexp(x, rate=1))
```



Question 8

(10 points)

Determine the average length in days for a loan for each group of Carnegie Mellon-affiliated patrons, by which we mean each element of the `Patron Group` column that begins with "Carnegie Mellon". (There are five in all.) To be clear: you are to approach this as a "split-apply-combine" problem, as opposed to extracting information for each group separately one after the other. A few notes here that might be helpful: (1) you have to initially filter the `Patron Group` column *in the exact same manner* that you filtered the loan dates in Q4, so that each element of the filtered `Patron Group` column matches 1-to-1 with a loan duration; (2) after this, you filter the patron group and loan duration vectors *again*, to limit yourself to CMU-affiliated patrons; and (3) if `pg` is your filtered patron group vector, do `pg = droplevels(pg)` so that the other non-CMU-related factor levels are dropped (otherwise you'll get output for *all* patron groups, with most displaying `NA` for the average loan duration). You can apply either base `R` or `tidyverse` functions to compute your answers. Note that I personally get 184.16142 for CMU faculty: we keep items for six months on average.

```
library(tidyverse)
# remove rows with no return date
datab = data[data$Return.Date != "",]
# convert all times to POSIXct for calculation
datab$Loan.DateTime <- as.POSIXct(paste(datab$Loan.Date, datab$Loan.Time), format="%m/%d/%y %H:%M:%S", t
datab$Return.Date = as.POSIXct(datab$Return.Date, format="%m/%d/%y", tz = "EST")
datab$Loan.Duration = difftime(datab$Return.Date, datab$Loan.DateTime, units="days")
# remove rows where it was removed before it was loaned out
```

```
datab = datab[datab$Return.Date > datab$Loan.DateTime,]
# filter out non CMU affiliated patrons
pg = filter(datab, grepl("^Carnegie Mellon", Patron.Group))
# create table showing the average length in days for loans
pg %>% group_by(Patron.Group) %>% summarize(mean(Loan.Duration))
```

```
## # A tibble: 5 x 2
##   Patron.Group                          `mean(Loan.Duration)`
##   <fct>                                 <drtn>
## 1 Carnegie Mellon Faculty               184.16411 days
## 2 Carnegie Mellon Grad Students         148.57962 days
## 3 Carnegie Mellon Staff                 278.06983 days
## 4 Carnegie Mellon Undergrads             82.92624 days
## 5 Carnegie Mellon Visiting Faculty/Staff 190.11853 days
```

Question 9

(10 points)

How many items were loaned on average on each day of the week in 2018? In other words, how many loans were made on average on Mondays, and on average on Tuesdays, etc.? In Q3, you created a vector of dates and times for all loans. Subset this vector to include only loans during 2018, and then determine the day associated with each date/time. To force the days to be in order, cast the vector of day names to a factor variable, and explicitly set the levels of that variable to "Monday", "Tuesday", etc. Last thing to remember: most days occur 52 times during a year, but one occurred 53 times in 2018. This will affect your computation of the mean! Note: I found an average of 23.09615 loans on each Saturday of 2018.

```
recallnewsom = newframe
recallnewsom = subset(recallnewsom, datetime >= as.Date("2018-01-01"))
recallnewsom = subset(recallnewsom, datetime < as.Date("2019-01-01"))
recallnewsom$dayofweek = weekdays(recallnewsom$datetime)
counts = recallnewsom %>% count(dayofweek)
counts$averageloans = counts$n/52
counts
```

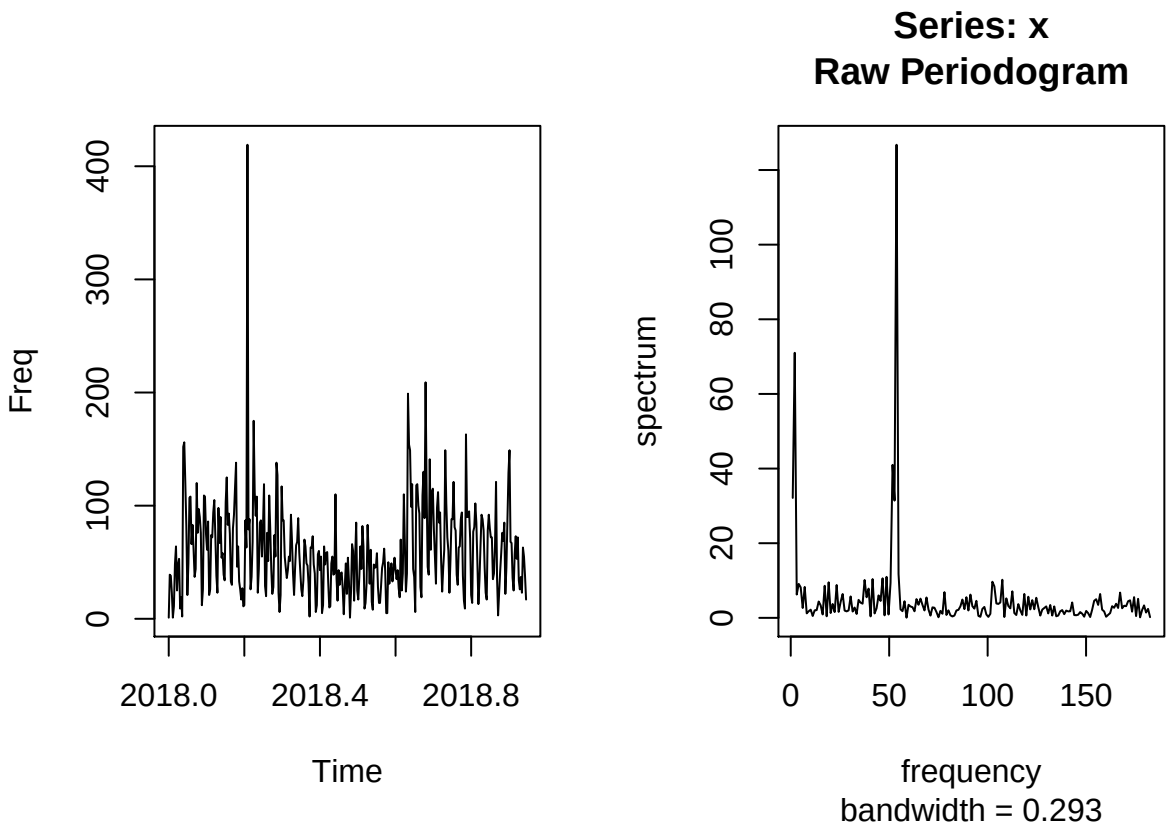
```
##   dayofweek    n averageloans
## 1    Friday 2884     55.46154
## 2    Monday 3639     69.98077
## 3  Saturday 1035     19.90385
## 4    Sunday 1359     26.13462
## 5  Thursday 3740     71.92308
## 6   Tuesday 3977     76.48077
## 7 Wednesday 4174     80.26923
```

Question 10

(10 points)

Create a time series for the loans in 2018, and then plot both the time series and its periodogram. Note that you should cast your date-and-time vector that you created in Q9 to `Date`, then use `table()` to determine the number of loans per date. You will probably have to look back at HW 10 (Q2, in particular) to recall how to define a time series for daily data. Finally: after you create your periodogram, determine the two most dominant frequencies and interpret them below.

```
df = recallnewsom
df$datetime = as.Date(df$datetime)
dates = sort(df$datetime)
dates.df = as.data.frame(table(dates))
dates.df = dates.df[c(-1)]
loan.ts = ts(dates.df,start=c(2018,1),frequency=365)
par(mfrow=c(1,2))
plot(loan.ts)
s = spectrum(loan.ts,log="no")
```




```
w = which.max(s$spec)
max(s$freq)/(f = s$freq[w])
```

```
## [1] 3.396226
```

```
FILL ME IN (with line breaks)
```

Question 11

(10 points)

Create a function that takes in a date (as a character string), converts it to `Date` format, and returns how many loans were made on that date. A few notes: (1) you will also need to pass in your vector of dates-and-times from Q3, as cast to `Date`, because you are going to determine how many times your test date

appears in that vector of dates; (2) you will also have the date format as an argument, with a default value of "%Y-%m-%d", because your function should allow the user to express the date in different formats; and (3) you will utilize `tryCatch()` with an `error` argument (where the error message should be "ERROR: bad date format."), so that if the input date cannot be converted to `Date` format and thus gets converted to `NA`, your code can clean things up without the code chunk itself failing. Note that inside `tryCatch()`, you should check if the converted date is `NA` and if it is, then you should issue a `stop()` function call (which will then trigger your error message to be displayed.) Note that for April 18, 2019, I find that 73 items were loaned. (Test your function with "4/18/2019" to see if you get this total.) Important: do not test your function with, or correct for, input that includes a two-digit year. This keeps things simpler. In a real-world code, you'd need to do that, but you need not do that here.

```
frame$loandate = as.Date(frame$Loan.Date, format = "%m/%d/%y")

numloans = function(datestring, vec, fmt = "%Y-%m-%d") {
  tryCatch({
      dt = as.Date(datestring, format = fmt)
      if(is.na(dt)) {
        stop()
      }
    },
    error = function(c) "ERROR: bad date format.")
  vec = as.Date(vec, format = fmt)
  tab = table(vec)
  return(tab[datestring])
}
numloans("2019-04-18", frame$loandate, fmt = "%Y-%m-%d")
```

```
## 2019-04-18
##         73
```

Question 12

(10 points)

Which library has older items? Hunt Library (library code `HUNT`) or Sorrells (library code `ENGR-SCI`)? And are the items in one library older in a statistically significant sense? You should only work with years in the `Begin Publication Date` field that are complete (four numerical digits, as opposed to 19??). Use a split-apply-combine function to display the mean publication date for items in each of the two indicated libraries (and those two only. . . note, use `droplevels()` similarly to the way you used it in Q8, to limit output to those two libraries). Also show the sample standard deviations. Then create two vectors—publication dates for Hunt, and publications dates for Sorrells—and perform a two-sample t-test. (Google how to do this if it is not obvious how to do so right away.) If the null hypothesis is that both samples are drawn from the same population of publication dates, do you reject the null, or fail to reject the null?

```
pattern = "[0-9]{4}$"
library = filter(data, grepl(pattern, data$Begin.Publication.Date))
library = filter(library, library$Library.Code == "HUNT" | library$Library.Code == "ENGR-SCI")
library$Begin.Publication.Date = as.numeric(as.character(library$Begin.Publication.Date))
library %>% group_by(Library.Code) %>% summarize(mean(Begin.Publication.Date))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   Library.Code `mean(Begin.Publication.Date)`
##   <fct>                                  <dbl>
## 1 ENGR-SCI                               2001.
## 2 HUNT                                   2000.
```

```
hunt = filter(library, library$Library.Code == "HUNT")
sorrells = filter(library, library$Library.Code == "ENGR-SCI")
t.test(hunt$Begin.Publication.Date, sorrells$Begin.Publication.Date)
```

```
##
##  Welch Two Sample t-test
##
## data:  hunt$Begin.Publication.Date and sorrells$Begin.Publication.Date
## t = -7.2659, df = 14785, p-value = 3.891e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.770841 -1.018390
## sample estimates:
## mean of x mean of y
##  1999.814  2001.209
```

```
The pvalue is very small (3.891e-13). We reject our null and have sufficient evidence \n
to conclude that both samples are likely drawn from different populations of publication dates.
```

Question 13

(10 points)

Which top-level Library of Congress classification codes appear the most at Hunt and Sorrells? By "top-level," I mean the first capital letter in each code, which maps to a particular subject area. For instance, if the first letter is "B", the item is a work of philosophy, psychology, or religion. Only include elements in the `LC Classification Top Line` column that *begin* with a capital letter and which are not listed as "Unknown". Output the top three first letters for each library (e.g., you might find that R, S, and T appear the most for a given library, in that order). Look up Library of Congress classification via Google and identify what each of the letters in your output stands for. (Note that I found that "N" appears 5009 times for items loaned from Hunt Library. Use this as a check of your output.) Hint: if Military Science appears a lot in your final answer, it means you didn't filter out the "Unknown"s. ("U" is the Military Science LoC classification code.)

```
h = filter(data, Library.Code == "HUNT")
s = filter(data, Library.Code == "ENGR-SCI")

h = filter(h, LC.Classification.Top.Line != "Unknown")
s = filter(s, LC.Classification.Top.Line != "Unknown")

h = filter(h, grepl("^[A-Z]", h$LC.Classification.Top.Line))
s = filter(s, grepl("^[A-Z]", s$LC.Classification.Top.Line))

h$letters = substr(h$LC.Classification.Top.Line, 1,1)
s$letters = substr(s$LC.Classification.Top.Line, 1,1)
```

```
tabh = h %>% group_by(letters) %>% summarize(n=n())
t.hunt = data.frame(tabh)
t.hunt = t.hunt[with(t.hunt, order(n, decreasing=TRUE)),]
head(t.hunt,3)
```

```
##    letters    n
## 15       P 8573
## 13       N 5009
## 8        H 4382
```

```
tabs = s %>% group_by(letters) %>% summarize(n=n())
t.sorrells = data.frame(tabs)
t.sorrells = t.sorrells[with(t.sorrells, order(n, decreasing=TRUE)),]
head(t.sorrells,3)
```

```
##    letters    n
## 12       Q 4988
## 15       T 1567
## 9        M  310
```

```
At Hunt, the 3 most are Language and Literature (P), Fine Arts (N),
and Social Sciences (H) in that order.
At Sorrells, the 3 most are Science (Q), Technology(T), and Music and Books on Music (M)
```

Question 14

(10 points)

What words appear most commonly in the titles of loaned items? To determine this, you should do the following: (1) limit yourself to items with a language code of "eng"; (2) only deal with lower-case letters (meaning, convert all letters to lower case); (3) replace all instances of "'s" (i.e., apostrophe s) and punctuation with empty strings (e.g., `gsub()`); and (4) concatenate all the words into a single vector (while removing any empty strings that make it into that vector). But seeing that "and" and "the" appear most often is boring. So use the `data_stopwords_smart$en` vector from the `stopwords` package to identify words which you should not include in your final table of the ten most common words. To be clear: if it is a stop word, it should not appear in your final table! (Hint: see `match()` or `%in%` to figure out how to identify which words are stop words and which are not.) (Note: I see that the word "theory" appeared in item titles 1667 times.)

```
if ( require(stopwords) == FALSE ) {
  install.packages("stopwords",repos="https://cloud.r-project.org")
  library(stopwords)
}
```

```
## Loading required package: stopwords
```

```
eng = filter(data, Language.Code == "eng")
eng$Title = tolower(eng$Title)
eng$Title = gsub("'s", "", eng$Title)
eng$Title = gsub("[[:punct:]]+", "", eng$Title)
words = unlist(strsplit(eng$Title, split=" "))
head(sort(table(words[!words %in% c(data_stopwords_smart$en, "")]), decreasing=TRUE), 5)
```

```
##
##      history introduction     american       design       theory
##         2328         1919         1727         1690         1667
```

Question 15

(10 points)

Display how many records are either `NA` or empty strings in each column. Note that you should not have a combination of each, i.e., there should be no columns in your data frame that have *both* `NA`s and empty strings in them. This simplifies coding. Note that I find that there's no publication date for 2340 items.

```
sapply(data, function(x) sum(is.na(x) | x==""))
```

```
##   LC.Classification.Top.Line                    Loan.Date
##                            5                            0
##                    Loan.Time                  Return.Date
##                            0                        11453
##                 Patron.Group                Material.Type
##                           60                            0
##                        Title       Begin.Publication.Date
##                            0                         2340
##                    Publisher                Language.Code
##                         4803                          987
##                     Subjects LC.Classification.Top.Line.1
##                         7061                           13
##                 Library.Code
##                        11806
```

Question 16

(10 points)

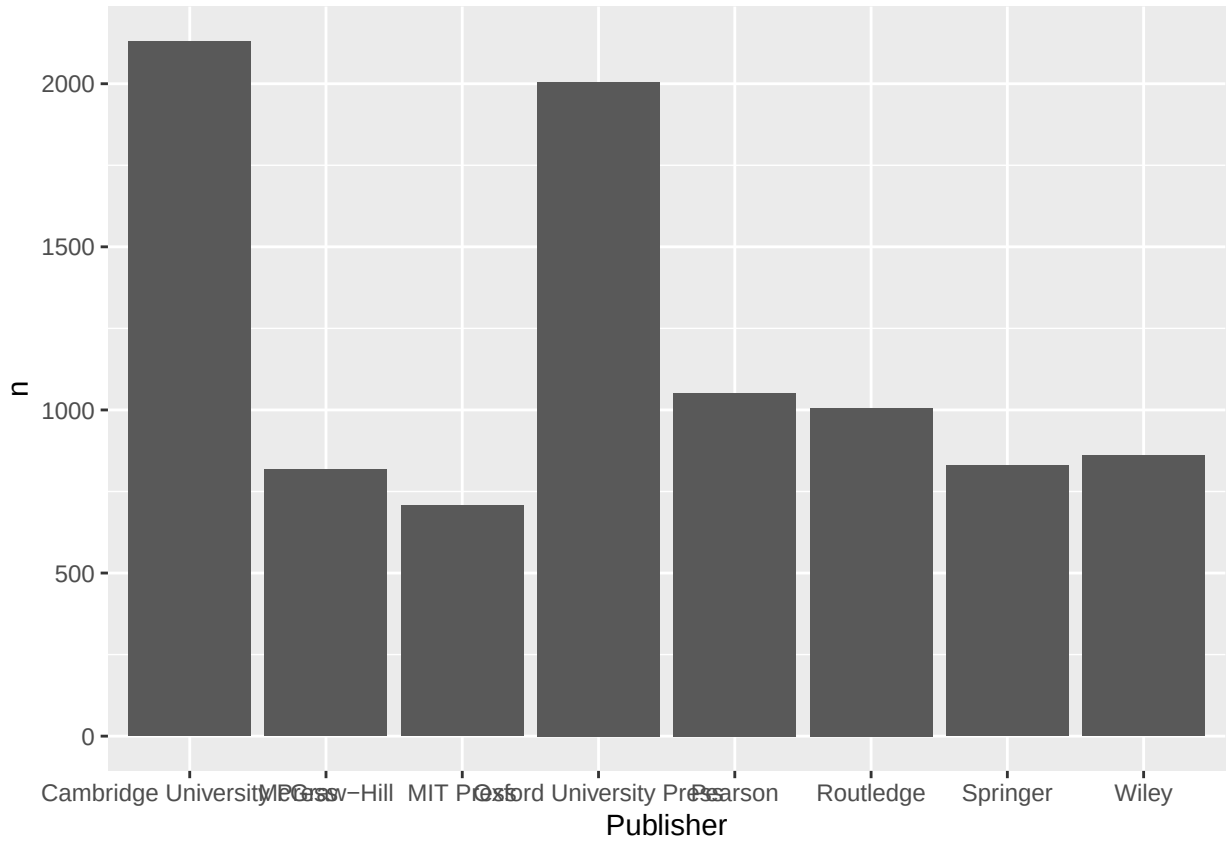
Display a bar chart via `ggplot()` that shows how many items were published by each of the top eight publishers of items in the dataset. Note that you should limit yourself to items with language code "eng", and you should eliminate any empty strings from the publisher vector before counting up how many items each publisher published. Your plot need not show the publishers in order of decreasing number of publications; by default it should show bars in publisher alphabetical order (with "Cambridge University Press" coming first). Pick a good color for your bars. Note: for reasons not entirely clear to me, you should pass the argument `stat="identity"` to `geom_bar()` in order for the plot to display correctly.

```
pub = filter(data, Publisher != "" & Language.Code == "eng")
t = head(count(pub, Publisher, sort=T), 8)
t = as.data.frame(t)
t
```

```
##                    Publisher    n
## 1 Cambridge University Press 2130
## 2    Oxford University Press 2005
## 3                    Pearson 1052
## 4                  Routledge 1006
## 5                      Wiley  860
```

```
## 6                  Springer  829
## 7               McGraw-Hill  818
## 8                 MIT Press  706
```

```
ggplot(t, aes(x = Publisher, y=n)) + geom_bar(stat="identity")
```

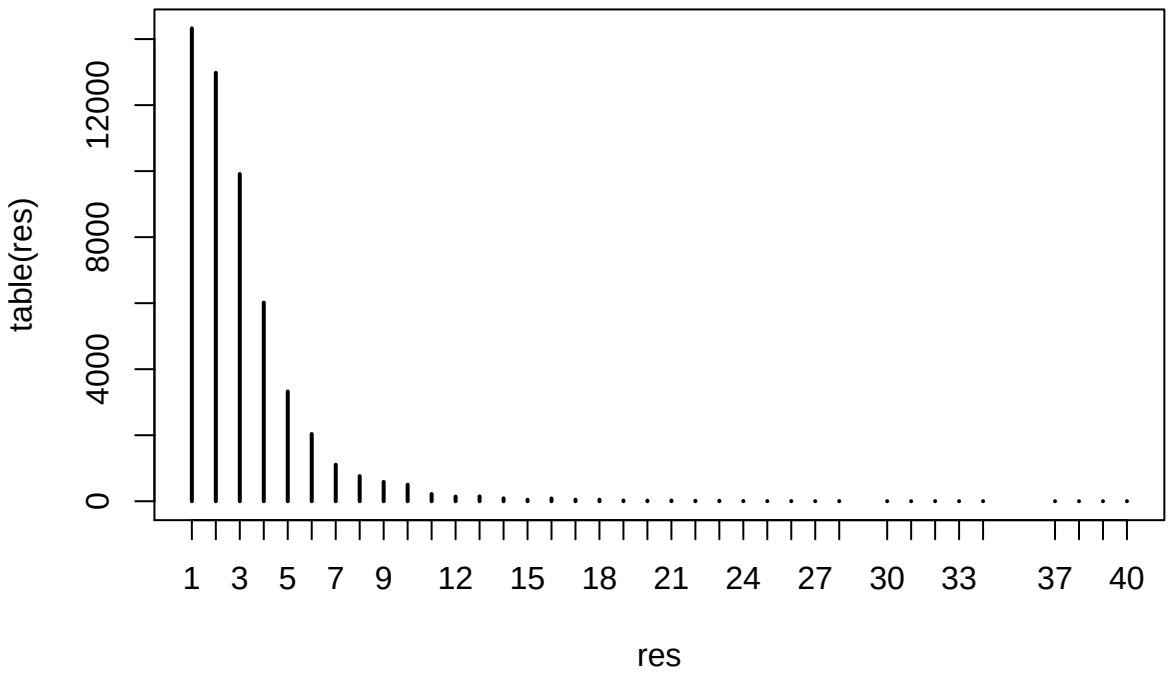


Question 17

(10 points)

Each item in the dataset is associated with some number of subjects. In the subject field, the subjects are separated by semi-colons (;). Utilize `strsplit()` and `sapply()` to determine the number of subjects associated with each item. Then plot the number of occurrences of each number of subjects. (For instance, 1098 items may have 4 subjects, while 780 have 5... plot with 4 and 5 along the x-axis, the 1098 and 780 along the y-axis.) Do not include the data point for no subjects. You should convert the y-axis to logarithmic scale; once you do so, you should see a roughly linear trend.

```
subjects = filter(data, Subjects != "")
subjects$Subjects = as.character(subjects$Subjects)
res = sapply(strsplit(subjects$Subjects, split=";"), length)
plot(table(res))
```

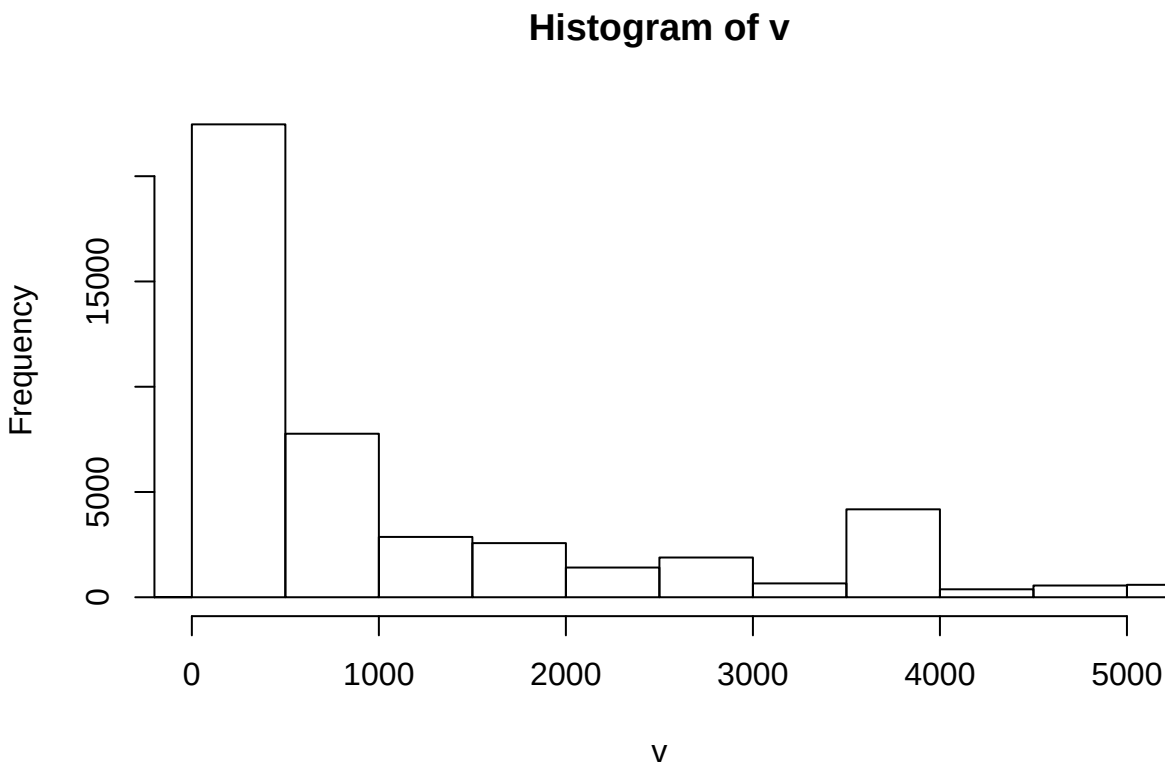


Question 18

(10 points)

Construct a histogram showing the distribution of the numerical components of each library classification code record. For instance, if the library classification code is "AZ874.5", then you need to extract the 874.5 and save it to a vector; once all extractions are done, histogram the vector. You should begin by filtering the data so as to retain only those classification codes that begin with a capital letter, then use tools of pattern matching to extract the numbers. If, when you make the histogram, you get the warning message "NAs introduced by coercion," it means that the vector of numbers has `NA`s in it (because you tried to coerce an empty string to a number, etc.)... go back and introduce code that forcibly removes the `NA`s from the vector.

```
ImpeachBiden = filter(data, grepl("^[A-Z]", data$LC.Classification.Top.Line))
vec = ImpeachBiden$LC.Classification.Top.Line
v = as.numeric(unlist(regmatches(vec,gregexpr("[-+]?[0-9]*\\.?[0-9]+([eE][-+]?[0-9]+)?",vec))))
hist(v, xlim=c(0,5000), breaks=1000)
```



Question 19

(10 points)

How many complete cases are there in your dataset? A "complete case" is a row for which all the fields have data. An incomplete case is a row in which there is an `NA` or an empty string. (Don't worry about cases like the classification code being "Unknown": just check for `NA`s and empty strings.) I find 32,345 complete cases.

```
f = function(x){
  for(n in x){
    if(is.na(n) | n == "") {
      return(FALSE)
    }
  }
  return(TRUE)
}
sum(apply(data, 1, f))
```

```
## [1] 32341
```

Question 20

(10 points)

Edit your file `dark_and_stormy.R` in your 36-350 `Git` repo so that it prints "It was a dark and stormy night so I stayed in to complete my R project. (Helped me avoid Covid-19 too.)" Then push your change to `GitHub` and use `source_url()` from the `devtools` package to run the code in the chunk below.

```
library(devtools)
```

```
## Error in get(genname, envir = envir) : object 'testthat_print' not found
```

```
source_url("https://raw.githubusercontent.com/Jackyrobot/36-350/main/dark_and_stormy.R")
```

```
## [1] "It was a dark and stormy night so I stayed in to complete my R project
(Helped me avoid Covid-19 too.)"
```

And with that, you're done.