g— title: “Final R Project” author: “36-350 – Statistical Computing” date: “Week 11 – Spring 2022” output: pdf_document: toc: no html_document: toc: true toc_float: true theme: spacelab —

Name: Sophia Li

Andrew ID: sophiali

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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 **April 29th, 9PM EST**.
- Submissions that are made **after April 29th, 9PM 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.

```r
file = read.csv("20200225_print-circulation_sample.csv", header=TRUE)
# file
```

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.

```r
# I don't need Loans (Not In House) or LC Classification Top Line
#     (1) Loans (Not In House): all 1's
#     (2) LC Classification Top Line: Redundant with (13)
#     (12) Subjects: debating if useless or good for grepl
# NEW:
#  [1] "LC.Classification.Top.Line" "Loan.Date"
#  [3] "Loan.Time"                  "Return.Date"
#  [5] "Patron.Group"               "Material.Type"
#  [7] "Title"                      "Begin.Publication.Date"
#  [9] "Publisher"                  "Language.Code"
# [11] "Subjects"                   "Library.Code"
# 13) not actually a duplicate
file.edit = file
file.edit[,c(1)] = NULL
# file.edit
dim(file.edit)
```

```
## [1] 59588    13
```

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.

```
# get days
loaned.day = apply(data.frame(file.edit$Loan.Date), 1, function(x){as.Date(x, format="%m/%d/%y")}) %>% a

# get days + times
temp = paste(loaned.day, file.edit$Loan.Time)
loaned = apply(data.frame(temp), 1, function(x){as.POSIXct(x, format="%Y-%m-%d %H:%M:%S")}) %>% as.POSI


max(loaned) - min(loaned)
```

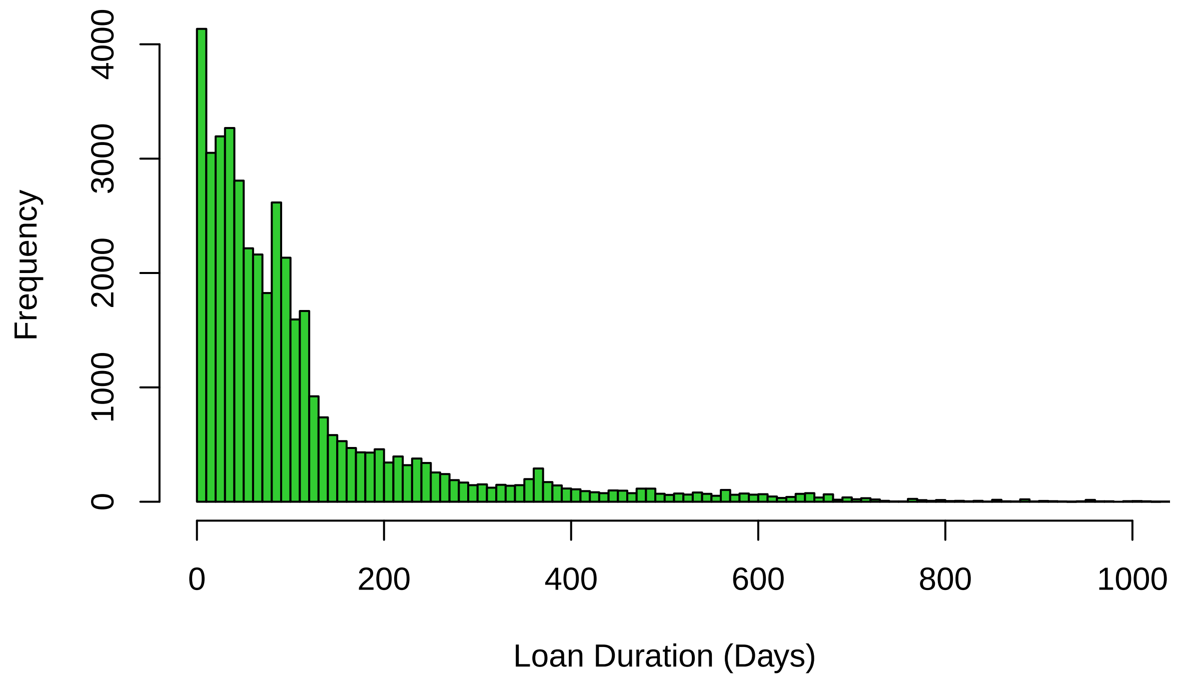
```
## Time difference of 9019.486 days
```

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.

```
# get return days + times
returned = apply(data.frame(file.edit$Return.Date), 1, function(x){as.POSIXct(x, format="%m/%d/%y")}) %

## Calculated days out
days.out = as.numeric((returned - loaned)/(60*60*24))
df.days.out = data.frame(days.out)

## Filter rest of file
new.file.edit = file.edit
new.file.edit["days.out"] = days.out
new.file.edit = new.file.edit %>% filter(!is.na(new.file.edit$days.out))
new.file.edit = new.file.edit %>% filter(new.file.edit$days.out >= 0)

## Filter out items that we're returned or were returned before loaned out
days.out = data.frame(days.out) %>% filter(., !is.na(days.out)) %>%
  filter(., days.out >= 0)


hist = hist(days.out$days.out, xlim=c(0,1000), breaks = 10000/10, xlab = "Loan Duration (Days)", main=NU
```



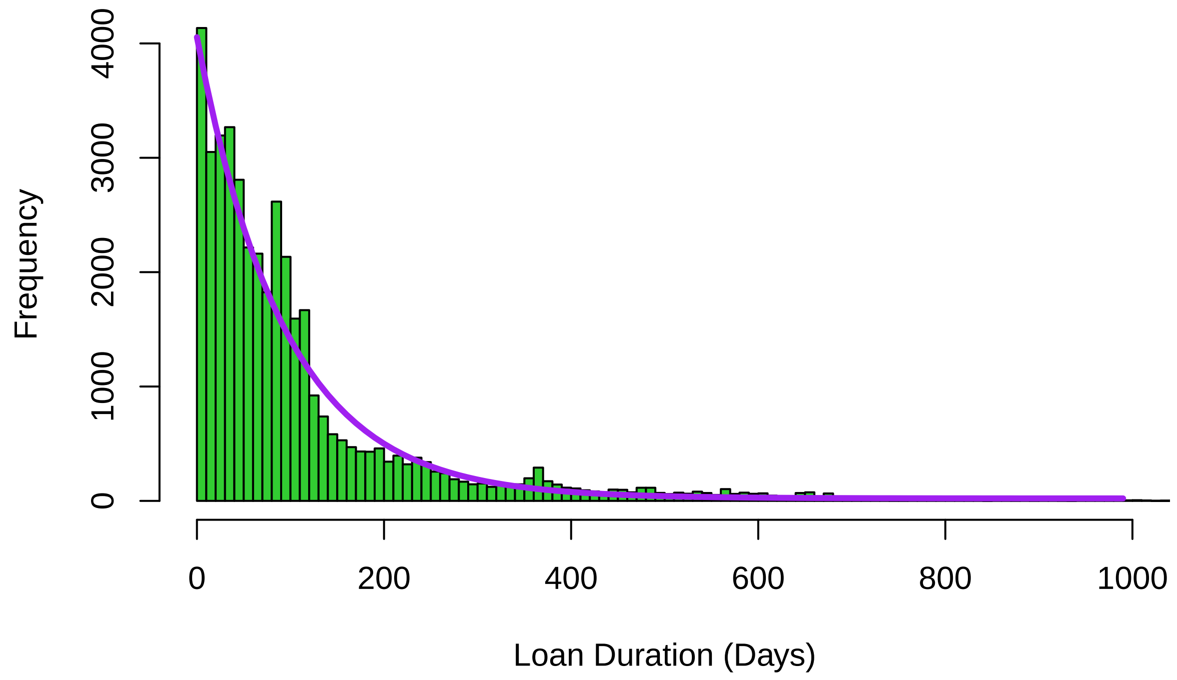
Question 5

(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.

```
## FROM: https://rpubs.com/mengxu/exponential-model
# Parameters
alpha <- 20
beta <- -0.05
theta <- 10

# Sample some points along x axis
n <- 100
x <- seq(1,100)

# Make  y = f(x) + Gaussian_noise
# data.df <- data.frame(x = x,
#                        y = alpha * exp(beta * x) + theta + rnorm(n))
```

```
data.df = data.frame(x = hist$breaks[seq(1,100)],
                     y = hist$counts[seq(1,100)])

# plot data
# plot(data.df$x, data.df$y)

# Select an approximate $\theta$, since theta must be lower than min(y), and greater than zero
theta.0 <- min(data.df$y) * 0.5

# Estimate the rest parameters using a linear model
data.df$y[data.df$y == 0] = 0.0001 # have to ensure there are no ln(0)'s
model.0 <- lm(log(y - theta.0) ~ x, data=data.df)
alpha.0 <- exp(coef(model.0)[1])
beta.0 <- coef(model.0)[2]

# Starting parameters
start <- list(alpha = alpha.0, beta = beta.0, theta = theta.0)

model <- nls(y ~ alpha * exp(beta * x) + theta , data = data.df, start = start)

# Plot fitted curve
hist = hist(days.out$days.out, xlim=c(0,1000), breaks = 10000/10, xlab = "Loan Duration (Days)", main=NU
lines(data.df$x, predict(model, list(x = data.df$x)), col = 'purple', lwd = 3)
```



```
alpha.0
```

```
## (Intercept)
##    2547.143
```

```
beta.0
```

```
##            x
## -0.007320147
```

```
model.0
```

```
##
## Call:
## lm(formula = log(y - theta.0) ~ x, data = data.df)
##
## Coefficients:
## (Intercept)            x
##     7.84273     -0.00732
```

```
##** Need to do someting to show rate parameter.
```

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.

```
# FILL ME IN
```

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.

```
# FILL ME IN
```

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.

```
Q8.file.edit = new.file.edit
Q8.file.edit$Patron.Group = as.factor(new.file.edit$Patron.Group)
pg = Q8.file.edit$Patron.Group

temp.df = data.frame(days.out = Q8.file.edit$days.out, Patron.Group = Q8.file.edit$Patron.Group)


pg =droplevels(pg, c("Items charged to the bindery",
                     "Academy of Lifetime Learning & Alumni",
                     "Faculty/Staff Family Members",
                     "In-House Loans",
                     "Inter Library Loan",
                     "Lyrasis users",
                     "Materials borrowed from EZ-Borrow",
                     "NonCMU PALCI Faculty borrowing in person",
                     "public",
                     "Qatar Education City ILL",
                     "Qatar Faculty",
                     "Qatar Staff Users",
                     "Qatar Undergrads",
                     "Retired CMU staff",
                     "Scholars Students from Central Catholic",
                     "Special Program Users",
                     "U. of Pittsburgh users",
                     "Users non-affiliated with Carnegie Mellon", ""))

ave.patron = tapply(temp.df$days.out, INDEX=pg, FUN=mean) %>% data.frame()
ave.patron
```

```
##                                               .
## Carnegie Mellon Faculty               184.16142
## Carnegie Mellon Grad Students         148.57759
## Carnegie Mellon Staff                 278.06885
## Carnegie Mellon Undergrads             82.92509
## Carnegie Mellon Visiting Faculty/Staff 190.12953
```

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.

```
only.2018 = loaned.day[loaned.day >= as.Date("2018-01-01", origin="1970-01-01") &
             loaned.day <= as.Date("2018-12-31", origin="1970-01-01")]

weekdays = weekdays(only.2018) %>%
  factor(., levels = c("Monday","Tuesday","Wednesday",
                                  "Thursday", "Friday","Saturday",
                                  "Sunday"))%>% as.data.frame()
weekdays.split = split(weekdays, f=weekdays)

days = sapply(weekdays.split, FUN=function(x){
  if (x[1,1] == "Monday") { nrow(x)/53}
  else { nrow(x)/52} } )
data.frame(days)
```

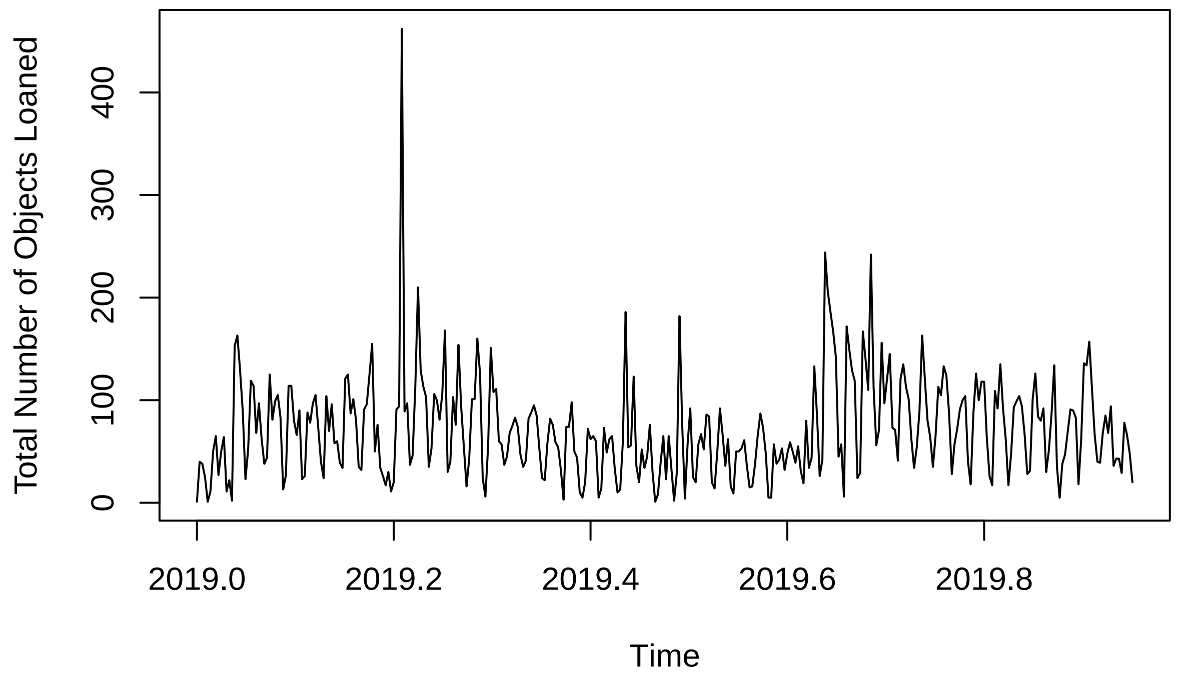
```
##               days
## Monday    83.11321
## Tuesday   93.30769
## Wednesday 94.07692
## Thursday  85.88462
## Friday    64.80769
## Saturday  23.09615
## Sunday    31.63462
```

Question 10

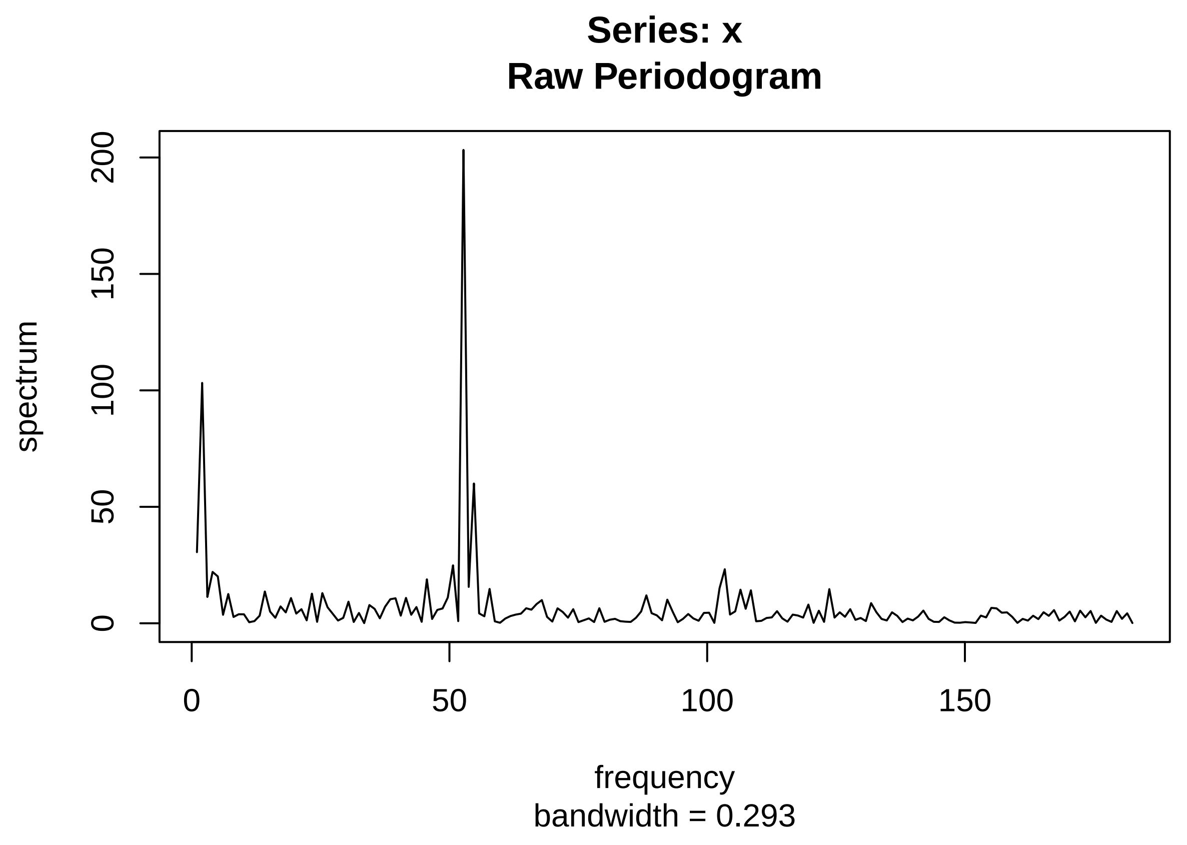
(10 points)

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.

```
# spectrum(only.2018)
start = as.numeric(format(as.Date("2018-01-01", origin="01-01-1970"), format="%j"))

count.table = table(only.2018)
sorted = names(count.table)
dates.ts = ts(count.table, start = c(2019, start), frequency = 365)

plot(dates.ts, xlab="Time", ylab="Total Number of Objects Loaned")
```



```
periodogram = spectrum(dates.ts,log="no")
```



```
max(periodogram$freq)/periodogram$freq[which.max(periodogram$spec)]
```

```
## [1] 3.461538
```

```
sort = sort(periodogram$spec)
which(periodogram$spec==sort[length(sort)])
```

```
## [1] 52
```

```
which(periodogram$spec==sort[length(sort)-1])
```

```
## [1] 2
```

```
The most dominant frequency is 52. THis means there are about 52 cycles per month.

The most dominant frequency is 3.461538 which is about April 15th. Carnival that year was from April 19
```

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.

```
date_to_num_loans = function(date, vect_date_times, d.format = "%Y-%m-%d") {
  as.date = as.Date(date, format= d.format)
  tryCatch(
    { if (is.na(as.date)) stop() },
    error = function(c) { message("ERROR: bad date format") }
    )
  num = vect_date_times[vect_date_times == as.date]
  return (length(num))
}

date_to_num_loans("4/18/2019", loaned.day, d.format = "%m/%d/%Y")
```

```
## [1] 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?

```
## Get library as factors
Q12.file.edit = file.edit
Q12.file.edit$Library.Code = as.factor(file.edit$Library.Code)
lev = Q12.file.edit$Library.Code %>%
  droplevels(., c("", "QATAR", "MELLON", "OFFSITE", "SEI"))

## Create new data frame with library and year and filter for
##  four numeric digits
temp.df = data.frame(Library = lev,
                     Year = Q12.file.edit$Begin.Publication.Date)

temp.df$Year = as.numeric(temp.df$Year)
```

```
## Warning: NAs introduced by coercion
```

```r
temp.df = filter(temp.df, !is.na(temp.df$Year)) %>%
  filter(., !is.na(.$Library))
idx = temp.df$Library

## Get the mean
mean.year = tapply(temp.df$Year, INDEX=idx, FUN=mean) %>%
  data.frame()
mean.year
```

```
##                   .
## ENGR-SCI 2001.209
## HUNT     1999.814
```

```r
sdv.year = tapply(temp.df$Year, INDEX=idx, FUN=sd) %>%
  data.frame()
sdv.year
```

```
##                  .
## ENGR-SCI 14.8693
## HUNT     18.8466
```

```r
## Publication vectors
hunt.vect = temp.df$Year[temp.df$Library == "HUNT"] %>% data.frame()
sorr.vect = temp.df$Year[temp.df$Library == "ENGR-SCI"] %>% data.frame()
test = t.test(hunt.vect, sorr.vect, alternative = "two.sided")
test
```

```
##
##  Welch Two Sample t-test
##
## data:  hunt.vect and sorr.vect
## 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
```

```r
print(paste0("p-value of t test = ", test$p.value))
```

```
## [1] "p-value of t test = 3.89088207647401e-13"
```

```
The p-value of the t teat is ~3.7*10^-13. This means that we should reject the null hypothesis because a
It's good there is a t test because, for me, seeing an average of 2001 vs 1999 wasn't that significant.
```

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.)

```r
# file.edit$LC.Classification.Top.Line
Q13.file.edit = file.edit
Q13.file.edit$Library.Code = as.factor(Q13.file.edit$Library.Code)

# Get library as factors
Q13.file.edit = file.edit
Q13.file.edit$Library.Code = as.factor(file.edit$Library.Code)
lev = Q13.file.edit$Library.Code %>%
  droplevels(., c("", "QATAR", "MELLON", "OFFSITE", "SEI"))

## Get first letters
let = apply(data.frame(Q13.file.edit$LC.Classification.Top.Line), 1, FUN=substr, start=1, stop=1)

temp.df = data.frame(Library = lev,
                     Classification = let) %>%
  filter(., .$Classification != "Unknown")
# temp.df

## Top 3
codes = as.factor(temp.df$Library)
top.three = tapply(temp.df$Classification, INDEX=codes, FUN= function(x){
  labels(sort(table(x), decreasing = TRUE)[1:3])})
top.three.val = tapply(temp.df$Classification, INDEX=codes, FUN= function(x){sort(table(x), decreasing =
top.three.val
```

```
## $`ENGR-SCI`
## x
##    Q    T    U
## 4988 1567  641
##
## $HUNT
## x
##    P    U    N
## 8573 5609 5009
```

```
Library of Congress:
Q -- Science
T -- Technology
U -- Military Science

P -- Language and Literature
U -- Military Science
N -- Fine Arts
```

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.)

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

```
## Loading required package: stopwords

## Warning: package 'stopwords' was built under R version 4.1.3
```

```r
## Make dataframe with lower case titles in english
Q14.file.edit = data.frame(Title = file.edit$Title,
                           Language = as.factor(file.edit$Language.Code))
eng = filter(Q14.file.edit, Q14.file.edit$Language == "eng") %>% data.frame()
eng$Title = tolower(eng$Title)
# eng$Title

## Remove punctuation
titles = eng$Title
reg.expr = gregexpr("'[a-z]+ ", titles, useBytes=FALSE)
regmatches(titles, reg.expr) = " "
reg.expr = gregexpr("[[:punct:]]", titles, useBytes=FALSE)
regmatches(titles, reg.expr) = ""
titles = data.frame(Title = titles)

## Get the most common words
common = paste(titles$Title, collapse=" ") %>%
  strsplit(., split="( )+") %>%
  data.frame(.)
notstopped = filter(common, !(common[[1]] %in% data_stopwords_smart$en)) %>% data.frame()
t = sort(table(notstopped[[1]]), decreasing = TRUE)[1:10]
data.frame(t)
```

```
##           Var1 Freq
## 1      history 2328
## 2  introduction 1919
## 3     american 1727
## 4       design 1690
## 5       theory 1667
## 6          art 1472
## 7  architecture 1366
## 8         world 1263
## 9       library 1197
## 10       modern 1193
```

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.

```
num.na = apply(file.edit, MARGIN = 2,
               FUN = function(x){ na_if(x, "")}) %>%
  data.frame() %>%
  apply(., MARGIN = 2, FUN = function(x){
  sum(is.na(x))
})
data.frame(num.na)
```

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

Question 16

(10 points)

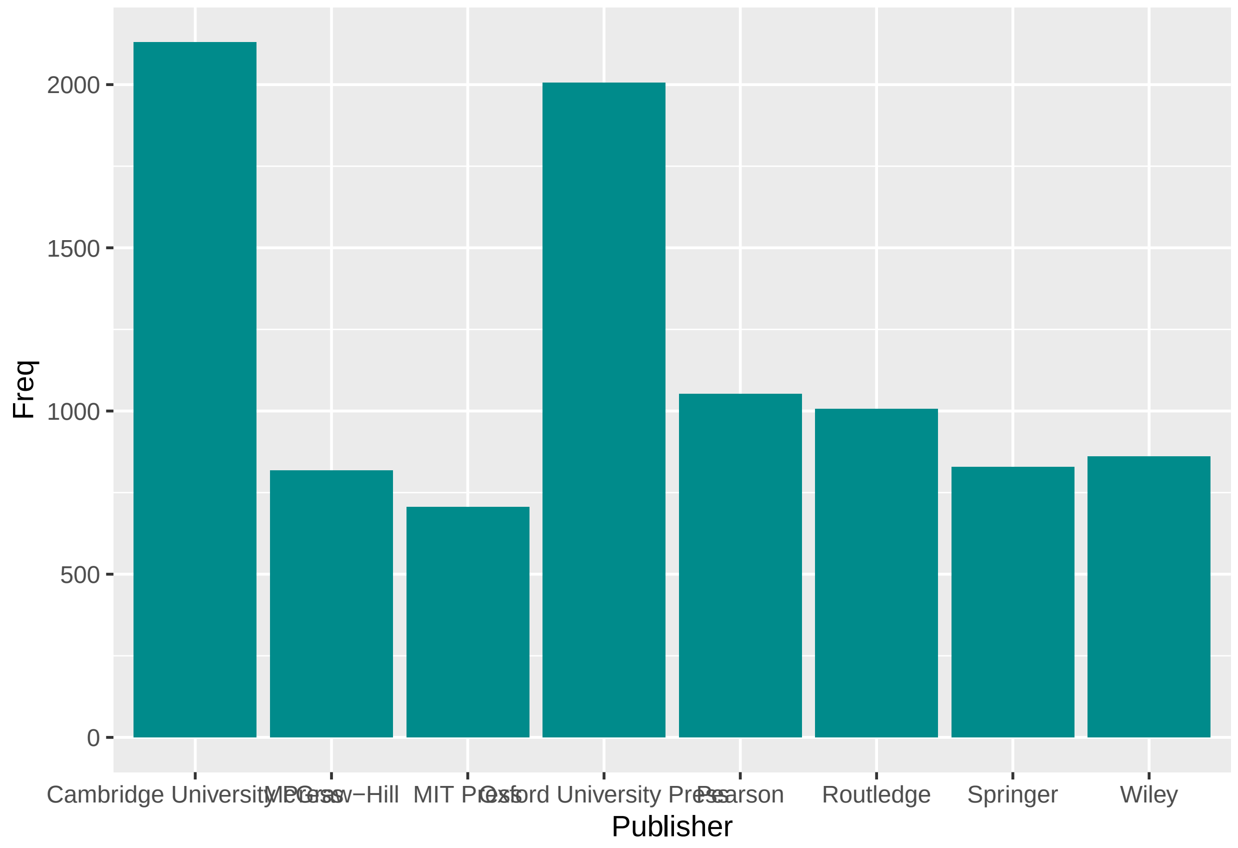
Display a bar chart via `ggplot()` that shows how many items were published by each of the top eight publishers of items in the dataset. Note that you should limit yourself to items with language code "eng", and you should eliminate any empty strings from the publisher vector before counting up how many 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.

```
# data.frame(file.edit$Publisher)
# The top most publisher is ""
Q16.file.edit = data.frame(Publisher = file.edit$Publisher,
                           Language = as.factor(file.edit$Language.Code)) %>%
  filter(., .$Language == "eng") %>% data.frame()
```

```r
top.8 = sort(table(Q16.file.edit$Publisher), decreasing = TRUE)[2:9]

## Plot
pub = data.frame(top.8)
colnames(pub) = c("Publisher","Freq")
pub$Publisher = as.character(pub$Publisher)
o = pub[order(pub$Publisher),]
o
```

```
##                    Publisher Freq
## 1 Cambridge University Press 2130
## 7                McGraw-Hill  818
## 8                  MIT Press  706
## 2    Oxford University Press 2005
## 3                    Pearson 1052
## 4                  Routledge 1006
## 6                   Springer  829
## 5                      Wiley  860
```

```r
ggplot(data=o, aes(x=Publisher, y=Freq)) +
  geom_bar(stat="identity", fill="cyan4")
```



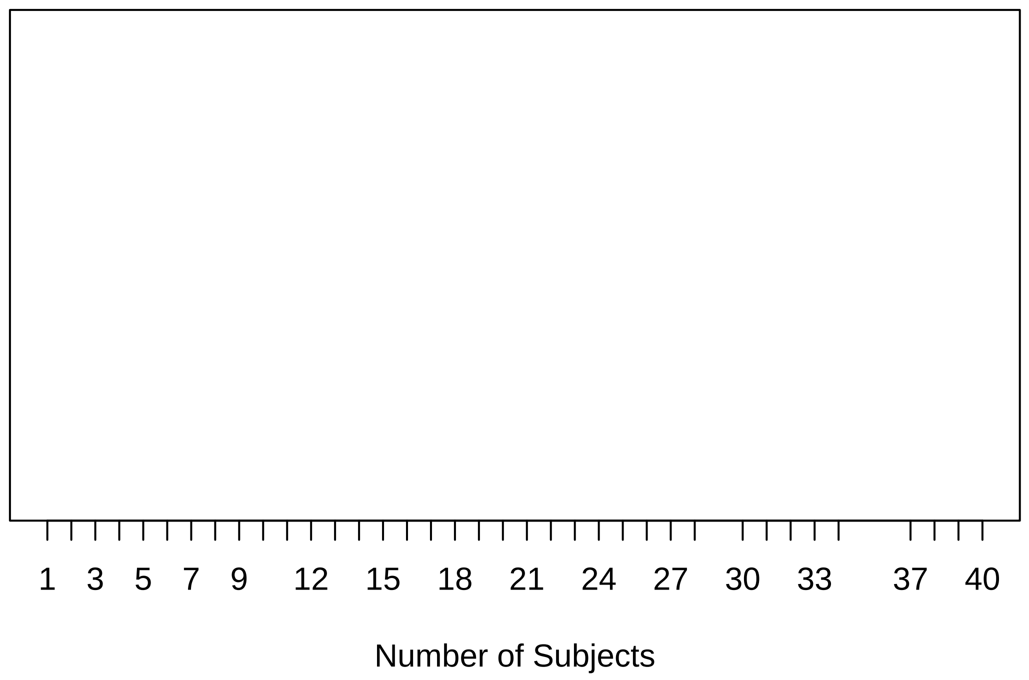
Question 17

(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.

```
# file.edit$Subjects
## !!! NEED TO USE SAPPLY !!!!
# num = apply(data.frame(sub = file.edit$Subjects), 1, FUN = function(x){
#   num = strsplit(x, split=";")
#   return(length(num[[1]]))
#   }) %>% data.frame(Nums = .)
#

## use sapply
num = sapply(file.edit$Subjects, FUN = function(x){
  num = strsplit(x, split=";")
  return(length(num[[1]]))
  })
# data.frame(factored)
factored = data.frame(Num = as.factor(num)) %>%
  filter(., . != 0)
removed = table(factored)[2:38] %>% data.frame()
plot(table(factored)[2:38], xlab = "Number of Subjects",
     ylab="Occurances", log="y") # can use log="y" but it makes the graph bad
```

```
## Warning in plot.window(...): nonfinite axis limits [GScale(-inf,4.15609,2, .);
## log=1]
```



```
# ggplot(data=table(factored)[2:38], aes(x=, y=freq))
```

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.

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.

```
num.na = apply(file.edit, MARGIN = 2,
               FUN = function(x){ na_if(x, "")}) %>%
  data.frame() %>%
  apply(., MARGIN = 1, FUN = function(x){
  sum(is.na(x))
})
sum(num.na == 0)
```

```
## [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)
```

```
## Warning: package 'devtools' was built under R version 4.1.3
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 4.1.3
```

```
#source_url(...)
source_url(paste0("https://raw.githubusercontent.com/sophialiCMU/36-350/main/",
                  "Project/dark_and_stormy.R"))
```

```
## i SHA-1 hash of file is 08f876926a9c35a635534c593f73b9cb36abcedc
```

```
## [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.