HW: Week 10

36-350 – Statistical Computing

Week 10 – Spring 2021

Name: Jacky Liu

Andrew ID:jackyl1

You must submit **your own** HW as a knitted PDF file on Gradescope.

```
suppressWarnings(library(tidyverse))
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.1
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

HW Length Cap Instructions

- If the question requires you to print a data frame in your solution e.g. `q1_out_df`, you must first apply **head(q1_out_df, 30)** and **dim(q1_out_df)** in the final knitted pdf output for such a data frame.
- Please note that this only applies if you are knitting the `Rmd` to a `pdf`, for Gradescope submission purposes.
- If you are using the data frame output for visualization purposes (for example), use the entire data frame in your exploration
- The **maximum allowable length** of knitted pdf HW submission is **30 pages**. Submissions exceeding this length *will not be graded* by the TAs. All pages must be tagged as usual for the required questions per the usual policy
- For any concerns about HW length for submission, please reach out on Piazza during office hours

---

Question 1

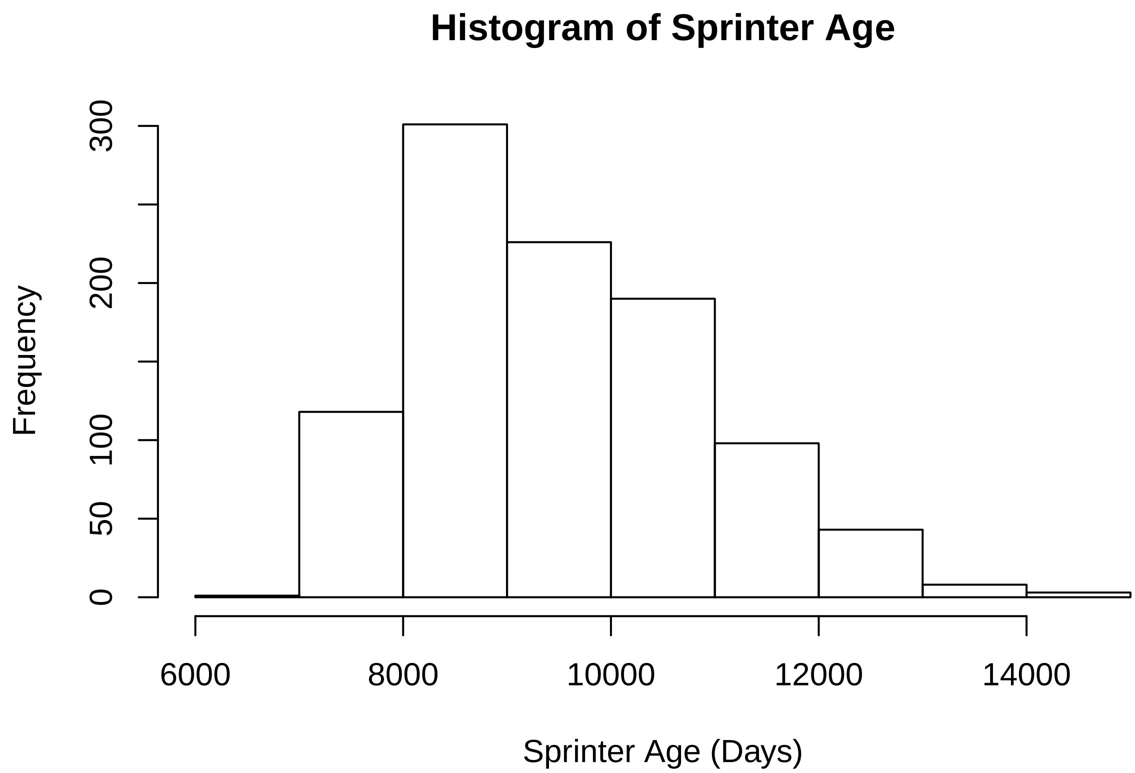
(20 points)

How old (in days) were sprinters on the days they achieved fast times? Below, we read in `sprint.lines`. Your goal: examine `sprint.lines`, extract the birthdays and the sprint days for each line, and determine the difference. Histogram your result. (Change the x-axis label to "Sprinter Age (Days)". You should observe a skew distribution that peaks between 8,000 and 9,000 days.

```r
sprint.lines = readLines("http://www.stat.cmu.edu/~mfarag/350/men_100m.html")
data.lines = grep(" +(9|10)\\.",sprint.lines)
sprint.lines = sprint.lines[min(data.lines):max(data.lines)]
sprint.lines[1] = substr(sprint.lines[1],10,nchar(sprint.lines[1]))

# FILL ME IN
differences = vector(length=length(sprint.lines))
birthday.pattern = "[0-9]{2}\\.[0-9]{2}\\.[4-9][0-9]"
sprintday.pattern = "[0-9]{2}\\.[0-9]{2}\\.[0-9]{4}"

for(i in 1:length(sprint.lines)){
  birthdaystring = regmatches(sprint.lines[i], regexpr(birthday.pattern, sprint.lines[i]))
  birthday = as.Date(sub("(..)$","19\\1", birthdaystring),format="%d.%m.%Y")
  sprintdaystring = regmatches(sprint.lines[i], regexpr(sprintday.pattern, sprint.lines[i]))
  sprintday = as.Date(sprintdaystring, format="%d.%m.%Y")
  differences[i] = sprintday - birthday
}
hist(differences, main = "Histogram of Sprinter Age", xlab = "Sprinter Age (Days)")
```



Here we read in data containing the dates that objects were loaned by the CMU libraries in April 2019:

```r
load(url("http://www.stat.cmu.edu/~mfarag/350/HW_10_Q2.Rdata"))
```

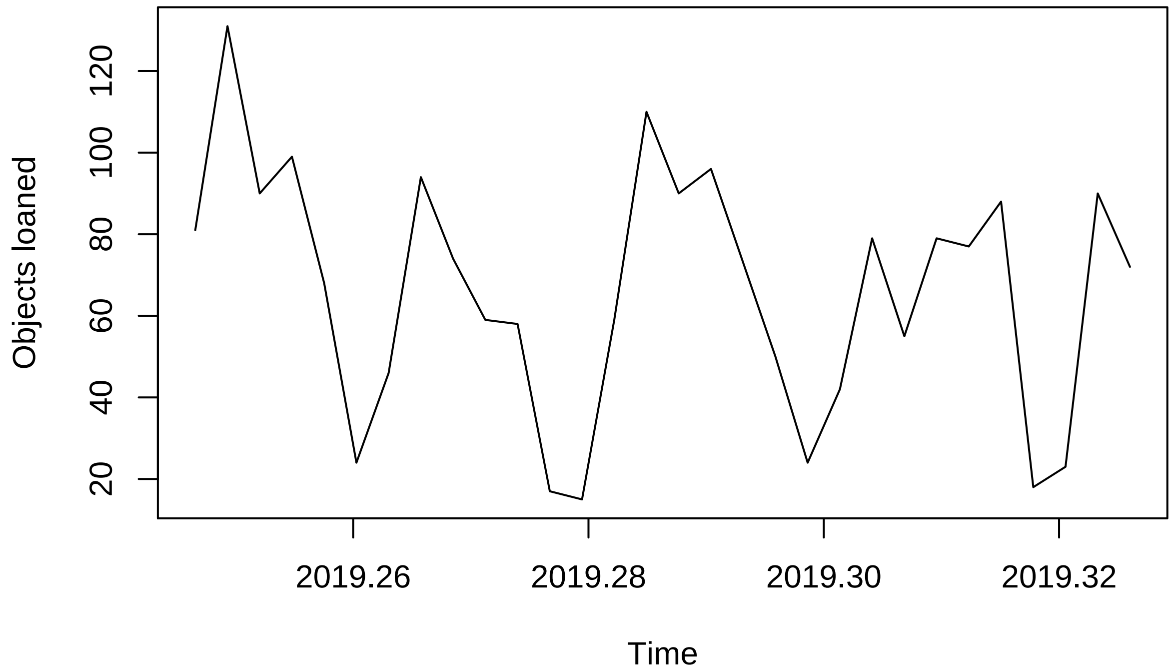
The variable that is loaded is `loan.dates`.

Question 2

(20 points)

From these data, create an object of the `ts` class that shows the total number of objects loaned each day. Note that to define a daily time series, the variable passed as `start` has to include the year (2019) and then the day of the year (e.g., January 1st is 1, February 1st is 32, etc.), and `frequency` should be set to 365. To get the day of the year: see `format()`: if you pass in the first day of the month and the argument `"%j"`, you will get out the day of the year. (Cast this to `numeric`, though!) Plot your result. The x-axis will show decimals indicating the fraction of the way through the year, so seeing, e.g., "2019.26" is OK. Change your y-axis label to something more appropriate than a variable name. (Hint: you'll want to make sure your dates `sort()` correctly when using `table()` to determine the number of loans per day. In other words, 4/10 should not immediately follow 4/1, but it might. If you convert to `Date` format first, sorting should work out OK.)

```r
dates = sort(as.Date(sub("(..)$","20\\1", loan.dates),format="%m/%d/%Y"))
dates.df = as.data.frame(table(dates))
dates.df = dates.df[c(-1)]
loan.ts = ts(dates.df,start=c(2019,91),frequency=365)
plot(loan.ts, ylab="Objects loaned")
```

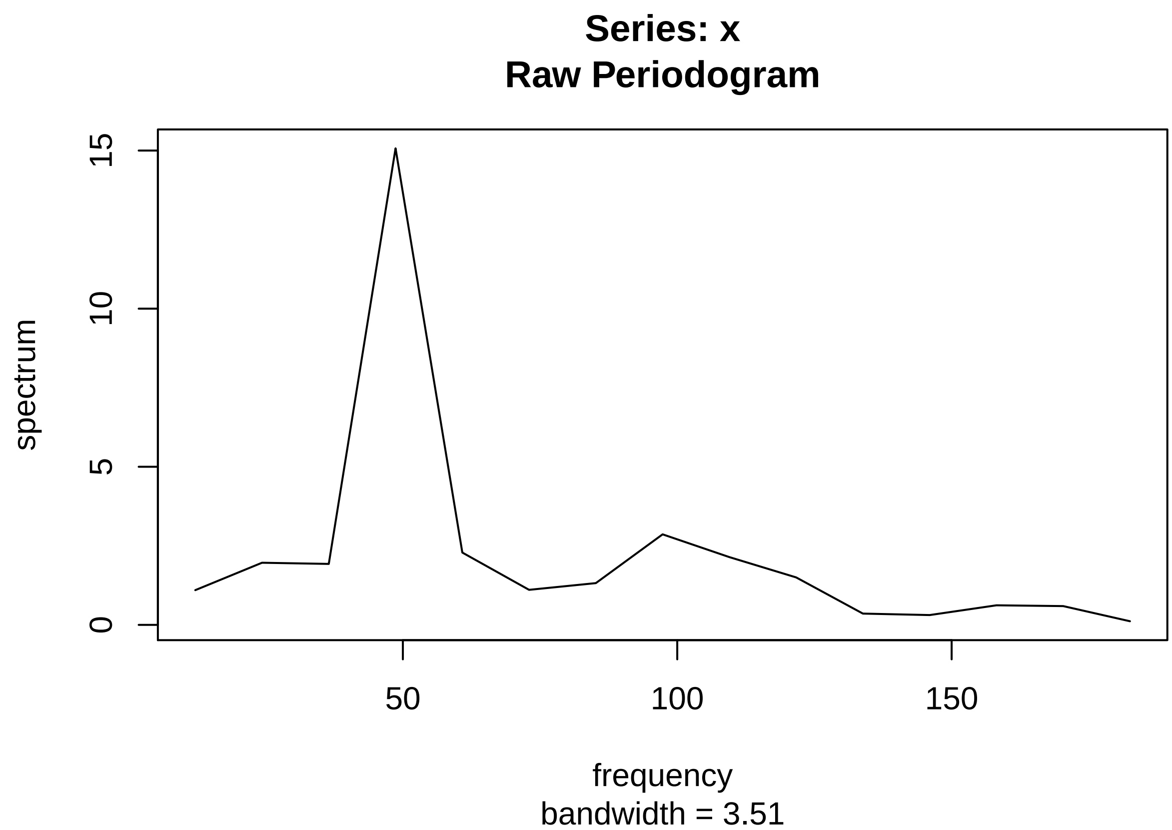


Question 3

(20 points)

Construct a periodogram for the time-series data in Q2. Determine how many cycles correspond to the maximum spectral value by dividing the maximum `frequency` value by the `frequency` value associated with the maximum `spectrum` value. Interpret that number of cycles.

```r
s = spectrum(loan.ts,log="no")
```



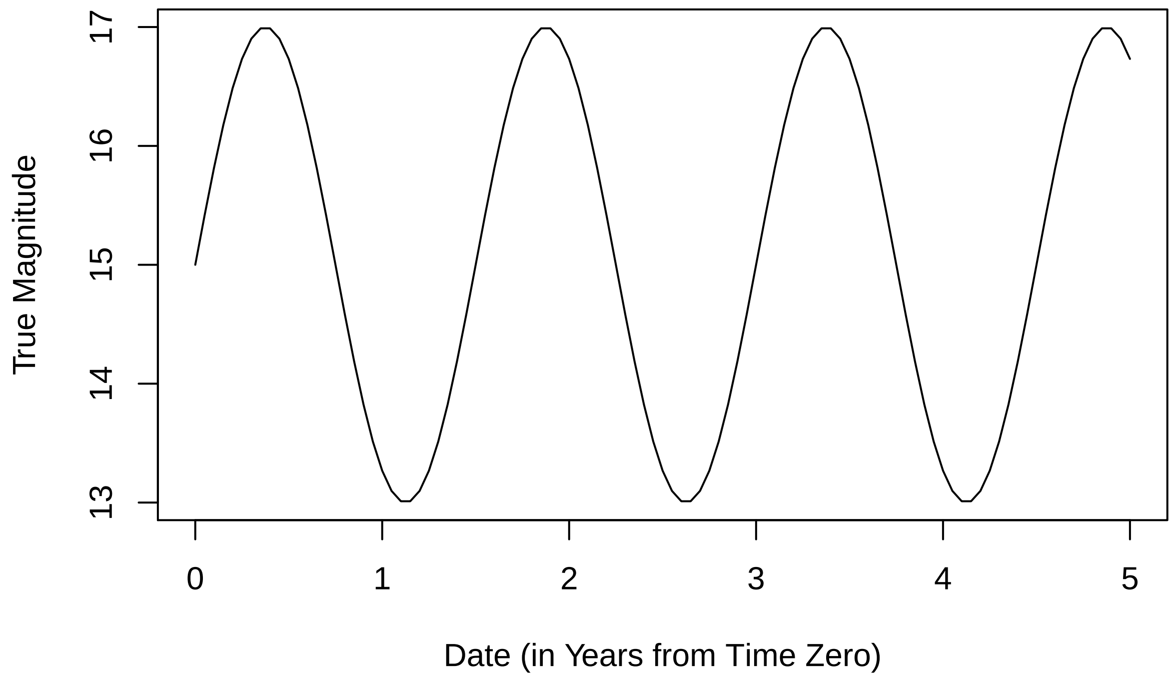

```
w = which.max(s$spec)
max(s$freq)/(f = s$freq[w])
```

```
## [1] 3.75
```

```
We get 3.75 cycles, which means a maximum wave cycles 3.75 times more in a day compared to the wave tha
```

Let's say you have a source of light whose magnitude (a logarithmic measure of brightness) varies sinusoidally:

```
t = seq(0,5,by=0.05)
y = 15 + 2*sin(2*pi*t/1.5)
plot(t,y,typ="l",xlab="Date (in Years from Time Zero)",ylab="True Magnitude")
```



Question 4

(20 points)

How well can you estimate the mean magnitude of this source if you observe it at n random times sampled uniformly over five years? (This isn't really about `Date` or `POSIXlt`, but a more general exercise that reminds you that features that you extract from any set of data—like a time series of measurements—are random variables... and that the more times you look, the better your estimate.) Write a function that generates n data given the model above. Assume each measurement has an additive uncertainty ϵ $N(0,(0.2)^2)$. Call your function $k = 1000$ times, and save the mean of the magnitude observed with each call. Try n = 10, n = 20, and n = 40, and record (and display!) the sample standard deviations of the magnitude means. You should see that the sample standard deviation for n = 40 is roughly half that for n = 10. $\sqrt{n}$ n'at.

```
set.seed(666)
generate = function(n){
  t = seq(0,5,by=0.05)
  y = 15 + 2*sin(2*pi*t/1.5)
  res = vector(length=n)
  for(i in 1:n){
    res[i] = sample(y, 1) + rnorm(1, 0, 0.2^2)
  }
  return(mean(res))
}
```

```r
k=1000
sds10 = vector(length=k)
sds20 = vector(length=k)
sds40 = vector(length=k)
for(i in 1:k){
  sds10[i] = generate(10)
  sds20[i] = generate(20)
  sds40[i] = generate(40)
}
cat("n=10:", sd(sds10), "\n")
```

```
## n=10: 0.4428067
```

```r
cat("n=20:", sd(sds20), "\n")
```

```
## n=20: 0.3130416
```

```r
cat("n=40:", sd(sds40), "\n")
```

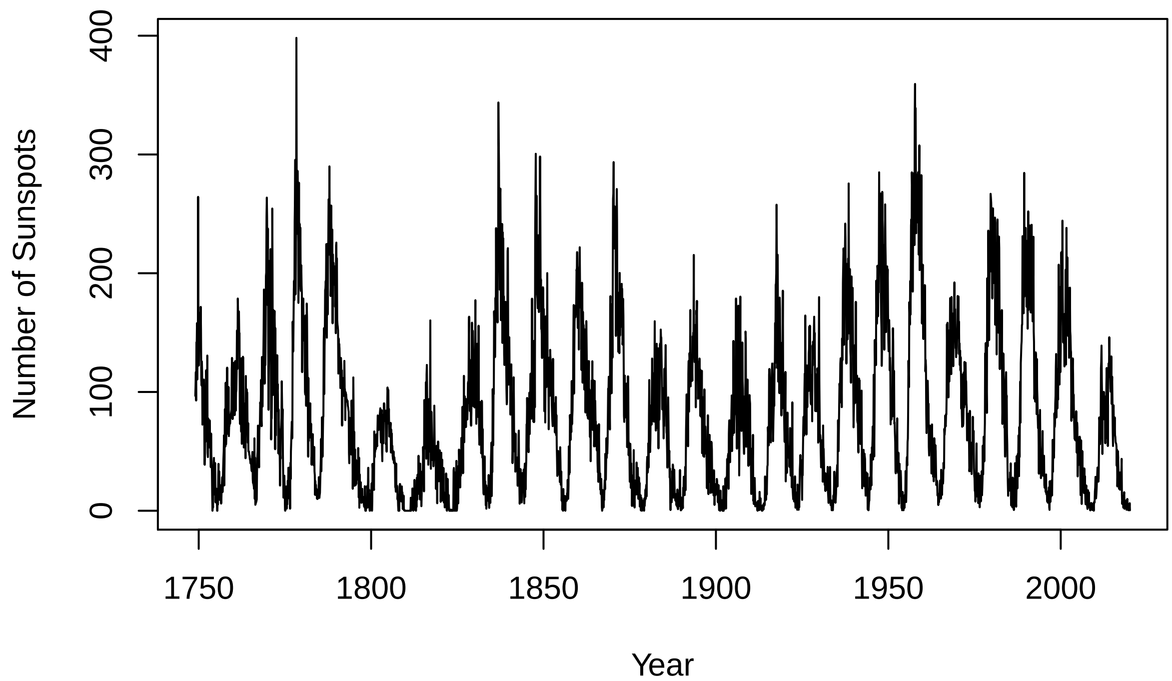
```
## n=40: 0.225932
```

Question 5

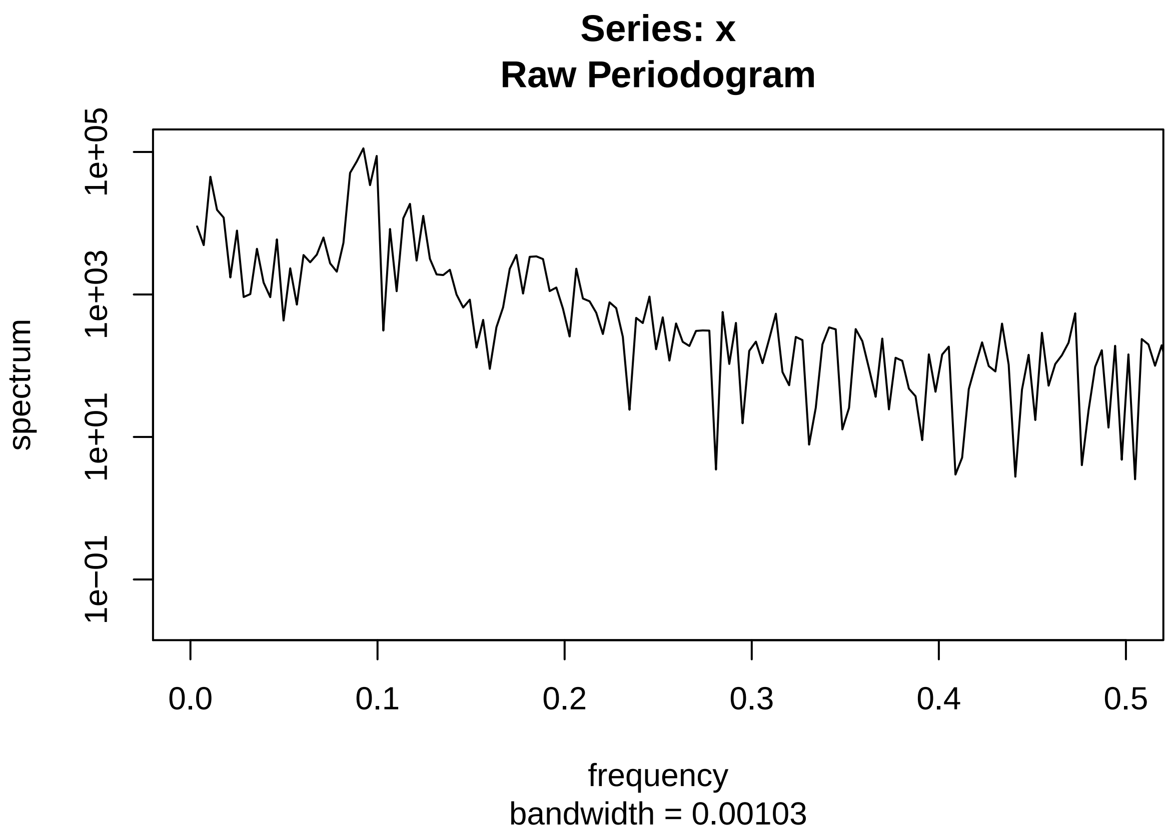
(20 points)

Data on monthly sunspot number are contained in the file `http://www.stat.cmu.edu/~mfarag/350/SN_m_tot_V2.0.csv`. Examine the file and read it into `R`, then define a time-series object with the data of the fourth column. Plot the time series (change the x-axis label to "Year" and the y-axis label to "Number of Sunspots"), and then plot the periodogram. For the latter, change the limits along the x-axis so as to zoom in on the peak that you should see. Determine the time-scale associated with the highest peak (using code, not by hand: examine the captured output of `spectrum()`... you need to capture the output, as otherwise `spectrum()` operates with `invisible()` return). Interpret this time-scale, using Google if necessary. (Hint: since you are inputting a time-series object into `spectrum()`, a `frequency` of 1 corresponds to 1 year.)

```r
sunspot = read.csv("http://www.stat.cmu.edu/~mfarag/350/SN_m_tot_V2.0.csv", sep=";", header=FALSE)
sunspot.df =
sunspot.subset = sunspot[c(4)]
sunspot.ts = ts(sunspot.subset, start=c(1749,1), frequency=12)
plot(sunspot.ts, xlab="Year", ylab="Number of Sunspots")
```




```
s = spectrum(sunspot.ts, xlim=c(0,0.5))
```




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

```
## [1] 0.09244444
```

```
The peak value of the periodogram is the 26th value which corresponds to a frequency of 0.09244444. The
```

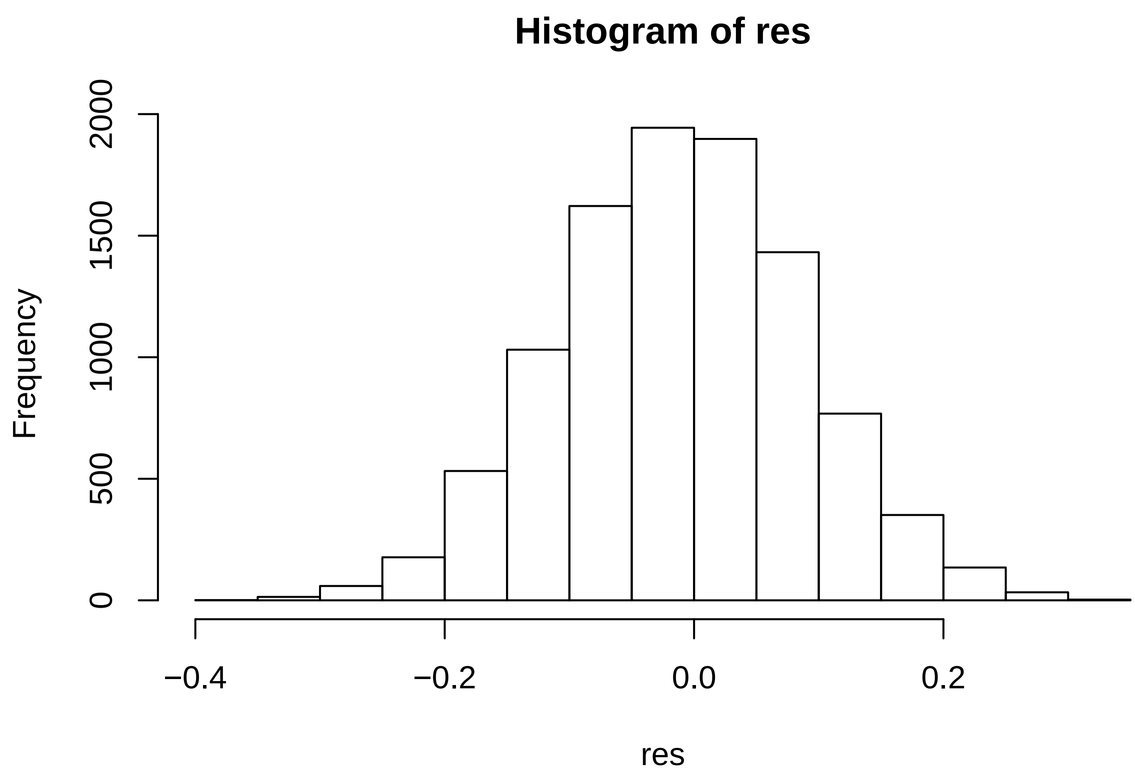
Question 6

(20 points)

What is observed correlation between two consecutive measurements in a white-noise time series? (The population value is zero.) Simulate 10,000 separate sequences that each contain 100 samples from a standard normal, input each sequence into `acf()`, and from the captured output, determine the acf value for consecutive data. (Note: for computational efficiency, pass the argument `plot=FALSE` to `acf()`.) Histogram your output. Last, determine the proportion of data that lie outside the confidence band given by $-1/n \pm 2/\sqrt{n}$... one would hope that this value is near 0.05. (It need not be exactly that.)

```
set.seed(666)
k=10000
res = vector(length=k)
for(i in 1:k){
  val = as.numeric(unlist(acf(rnorm(100), plot=FALSE))[2])
  res[i] = val
```

```r
}
hist(res)
```




```r
count = 0
for(num in res){
  if(num < -1/100 - 2/sqrt(100) || num > -1/100 + 2/sqrt(100)) {
    count = count + 1
  }
}
cat(count/length(res), "of the data lie outside the confidence band\n")
```

```
## 0.0414 of the data lie outside the confidence band
```