HW: Week 10

36-350 – Statistical Computing

Week 10 – Fall 2020

Name: Kimberly Zhang

Andrew ID: kyz

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

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

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

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

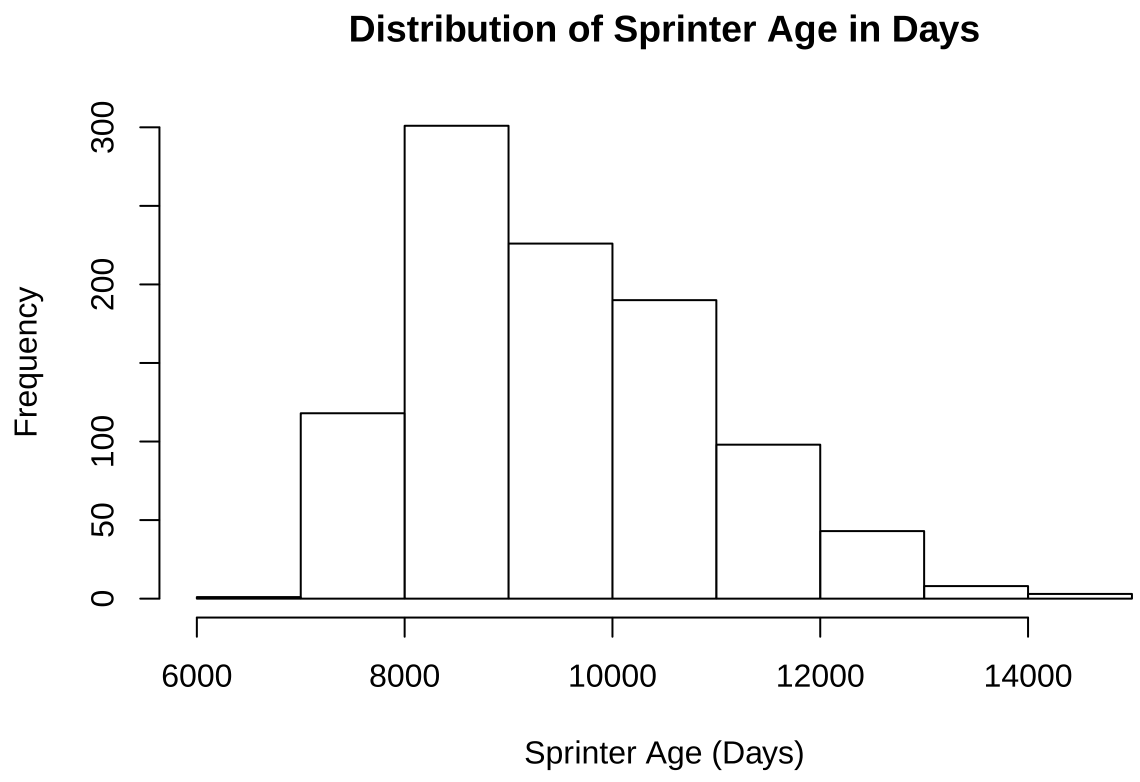
Question 1

(15 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.

```
sprint.lines = readLines("http://www.stat.cmu.edu/~pfreeman/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]))
birth.pattern = "[0-9]{2}.[0-9]{2}.[0-9]{2}"
comp.pattern = "[0-9]{2}.[0-9]{2}.[0-9]{4}"
birth.exp = regexpr(birth.pattern, sprint.lines, useBytes = TRUE)
comp.exp = regexpr(comp.pattern, sprint.lines, useBytes = TRUE)
birth.dates = regmatches(sprint.lines, birth.exp)
comp.dates = regmatches(sprint.lines, comp.exp)
birth.dates = as.Date(birth.dates, format= "%d.%m.%y")
comp.dates = as.Date(comp.dates, format = "%d.%m.%Y")
invalid = birth.dates[which(birth.dates > Sys.Date())]
birth.dates[which(birth.dates > Sys.Date())] =
  format(as.Date(invalid, "%Y-%m-%d"), "19%y-%m-%d")
```

```r
hist(as.numeric(comp.dates - birth.dates), xlab = "Sprinter Age (Days)",
     main = "Distribution of Sprinter Age in 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/~pfreeman/HW_10_Q2.Rdata"))
```

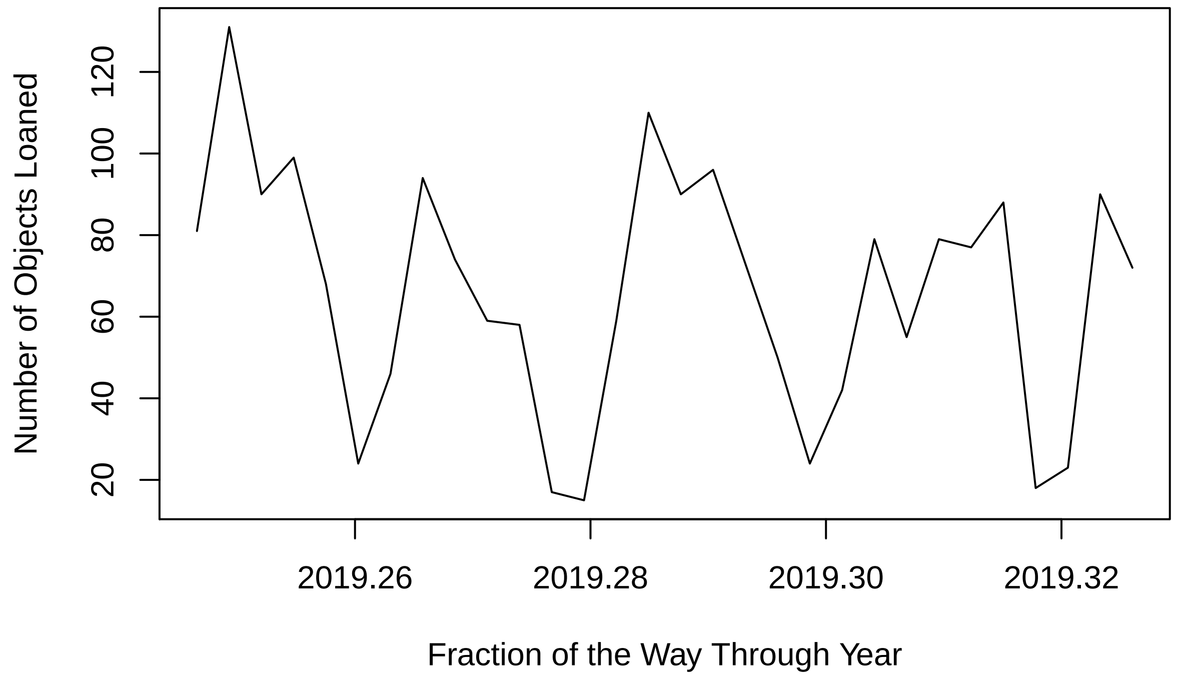
The variable that is loaded is `loan.dates`.

Question 2

(15 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
convert.to.days = function(d){
                  as.numeric(format(as.Date(d),
                  format = "%j"))
}
dates = as.Date(loan.dates, format= "%m/%d/%y")
counts = table(dates)
sorted.dates = names(counts)
ldates.ts = ts(counts, start = c(2019, convert.to.days(min(dates))),
               frequency = 365)
plot(ldates.ts, xlab = "Fraction of the Way Through Year",
     ylab = "Number of Objects Loaned", pch=19)
```

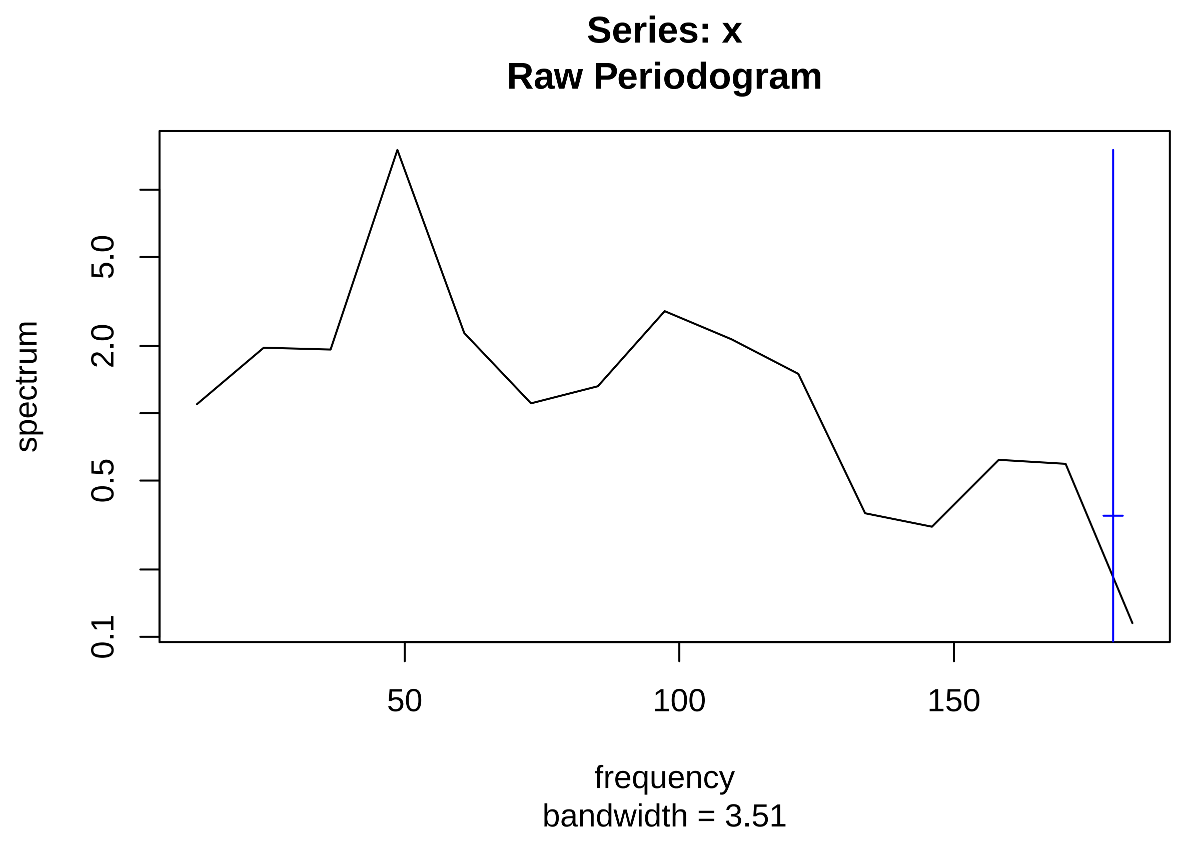


Question 3

(10 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.

```
periodogram = spectrum(ldates.ts)
```



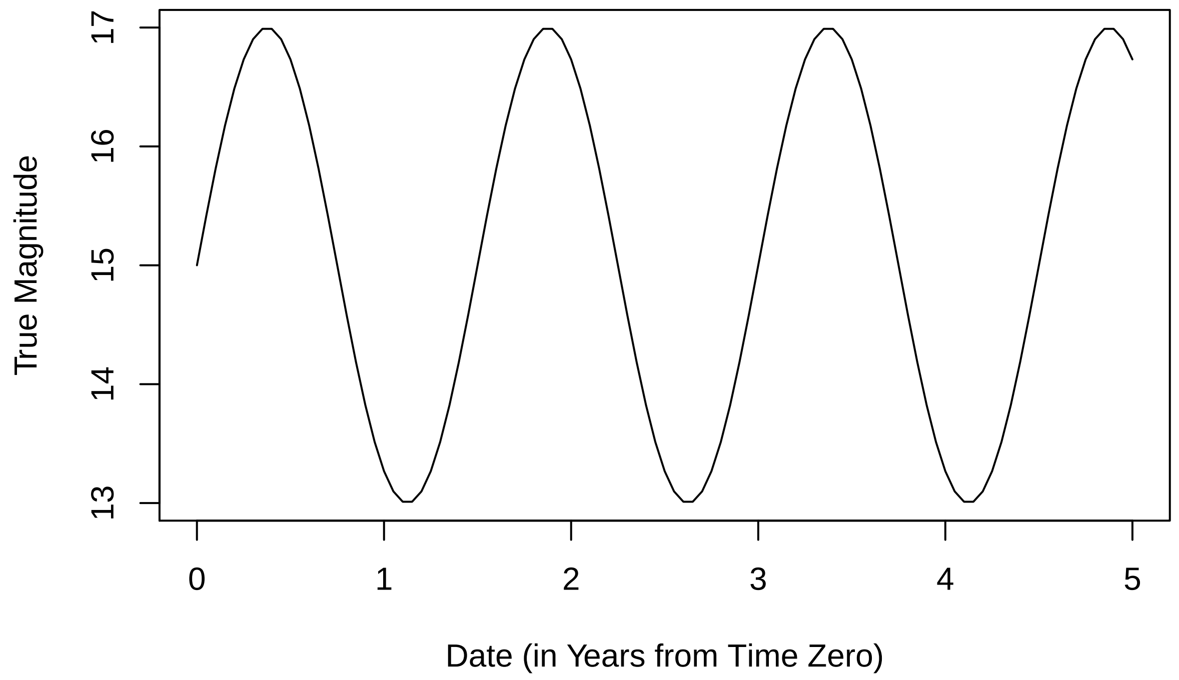
```r
max(periodogram$freq)/periodogram$freq[which.max(periodogram$spec)]
```

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

`Every 3.75 cycles, we will observe the maximal spectral value again.`

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

```r
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

(15 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.

```r
f = function(n){
  sampled = sample(y, n, replace = TRUE)
  epsilon = rnorm(n, 0, 0.2)
  mean(sampled + epsilon)
}
k = 1000
res10 = rep(NA, k)
res20 = rep(NA, k)
res40 = rep(NA, k)
for (i in 1:k) {
  res10[i] = f(10)
  res20[i] = f(20)
  res40[i] = f(40)
}
sd(res10)
```

```
## [1] 0.4411085
```

```
sd(res20)
```

```
## [1] 0.323361
```

```
sd(res40)
```

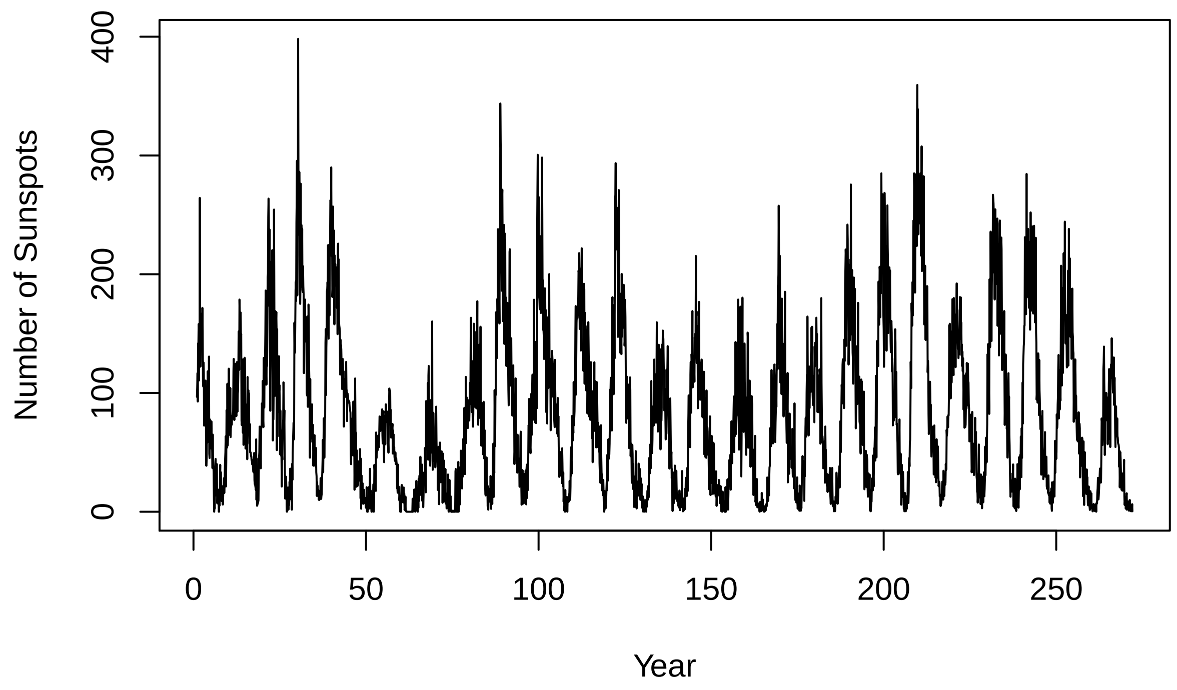
```
## [1] 0.2230421
```

Question 5

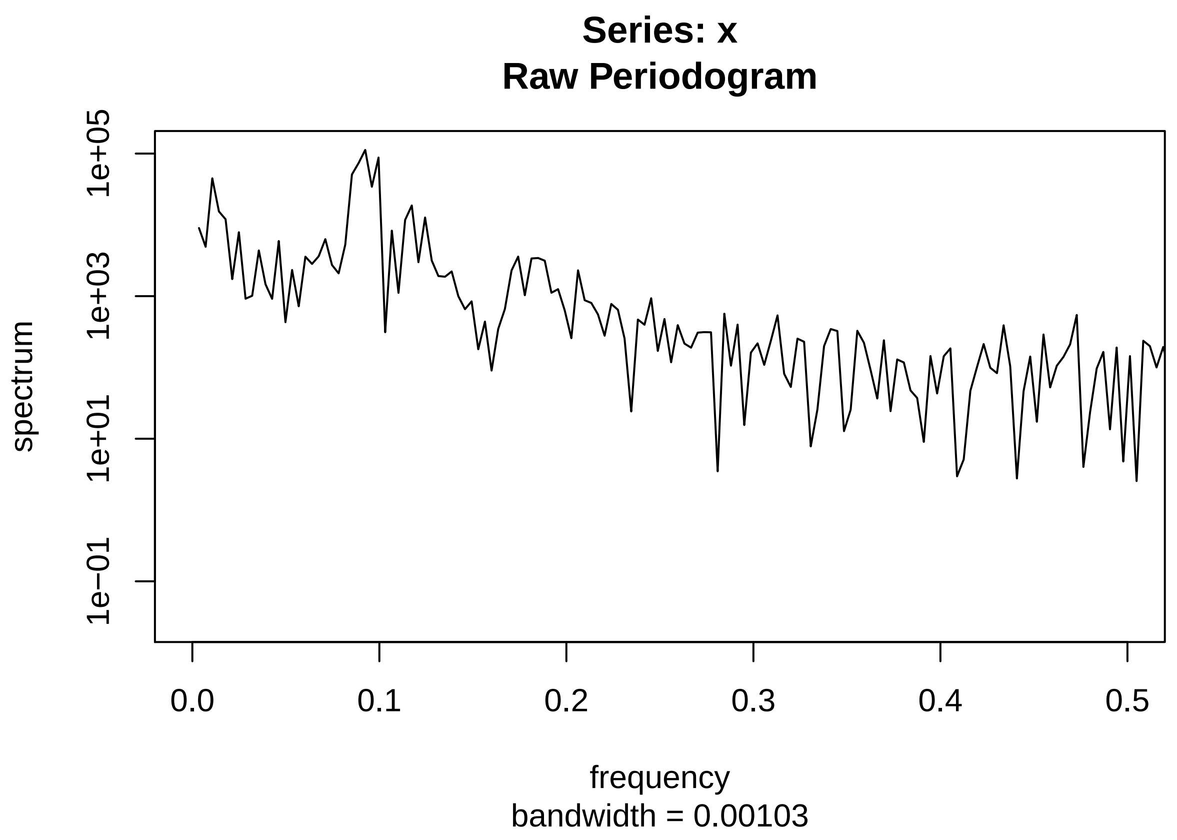
(15 points)

Data on monthly sunspot number are contained in the file `http://www.stat.cmu.edu/~pfreeman/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.)

```
sunspot = read.csv("http://www.stat.cmu.edu/~pfreeman/SN_m_tot_V2.0.csv",
                   sep = ";", header = FALSE)
sunspot.ts = ts(sunspot[,4], frequency = 12)
plot(sunspot.ts, xlab = "Year", ylab = "Number of Sunspots")
```



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



```r
1/p$freq[which.max(p$spec)]
```

```
## [1] 10.81731
```

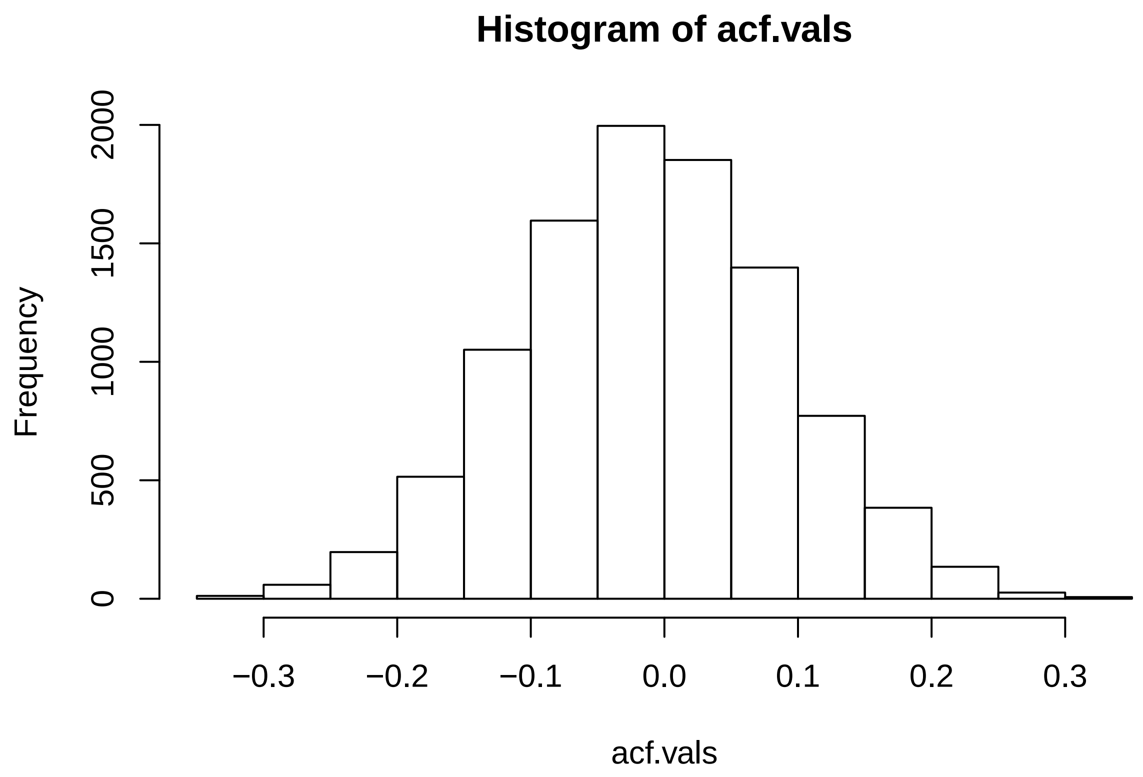
```
The dominant cyclical behavior has a frequency of approximately 0.1,
corresponding to about 10.8 years for a single cycle to complete. This makes
sense because a solar cycle takes 11 years to complete.
```

Question 6

(15 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.)

```r
wn.series = matrix(rnorm(10000*100), nrow = 10000)
acf.vals = apply(wn.series, 1, function(x){acf(x, plot = FALSE)$acf[2]})
hist(acf.vals)
```



```
(sum(acf.vals < (-1/100 - 2/sqrt(100))) +
  sum(acf.vals > (-1/100 + 2/sqrt(100))))/10000
```

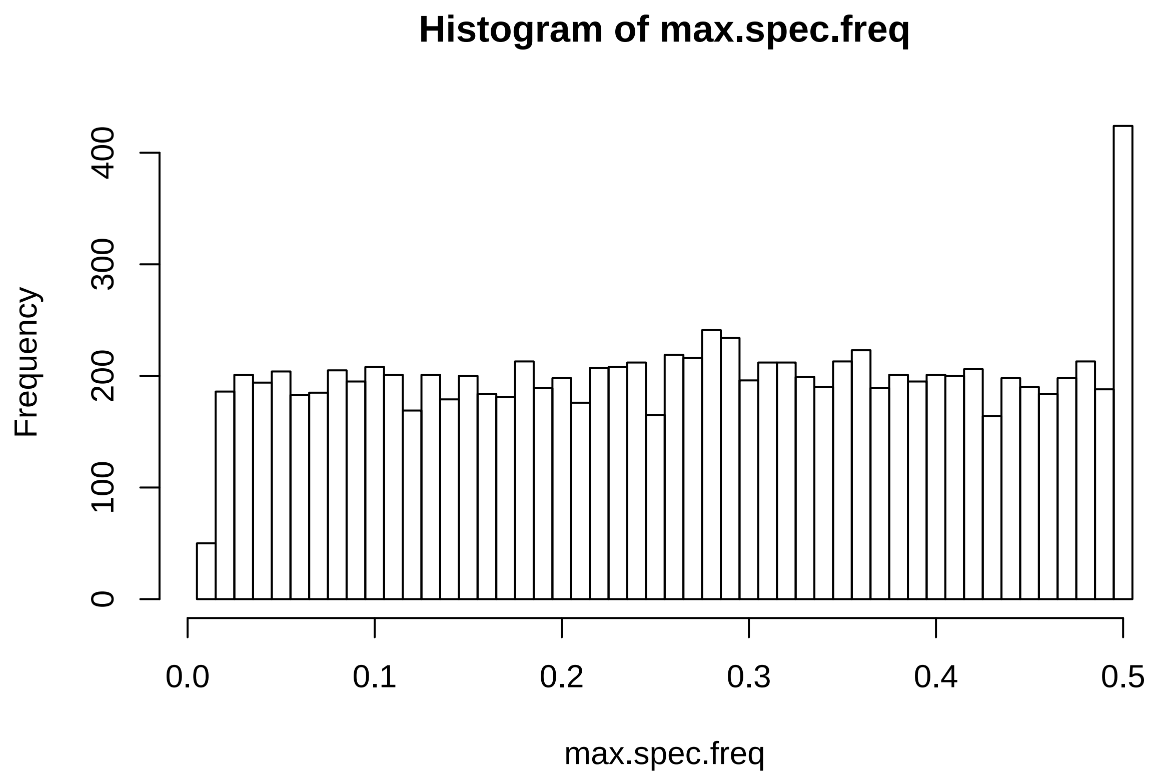
```
## [1] 0.0422
```

Question 7

(15 points)

Use the same data that you generated above, but now record for each sequence the frequency associated with the maximum periodogram value. For instance, if the largest value of `spec` for a given periodogram is at frequency 0.32, record the value 0.32. Given that we simulated white noise, we would expect that the distribution of the recorded frequencies should be uniform between 0.01 and 0.50. Histogram the frequencies, setting the breaks to be a sequence from 0.005 to 0.505 by 0.01. Then use an appropriate hypothesis test to test the hypothesis that the recorded frequencies are sampled from a uniform distribution with minimum value 0.01 and maximum value 0.50. (This isn't quite the right thing to do−you should utilize the discrete uniform distribution here−but this allows for more straightforward coding.) What happens if you exclude the first and last frequencies? You might see a much larger p value (meaning, much closer to 1).

```
max.spec.freq = apply(wn.series, 1, function(x){
                                  wn.spec = spectrum(x, plot = FALSE)
                                  wn.spec$freq[which.max(wn.spec$spec)]
                                  })
hist(max.spec.freq, breaks = seq(0.005, 0.505, by = 0.01))
```



```
ks.test(max.spec.freq, "punif", 0.01, 0.5)
```

```
## Warning in ks.test(max.spec.freq, "punif", 0.01, 0.5): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  max.spec.freq
## D = 0.0424, p-value = 4.441e-16
## alternative hypothesis: two-sided
```

```
excluded = max.spec.freq[max.spec.freq > 0.01 & max.spec.freq < 0.5]
ks.test(excluded, "punif", 0.01, 0.5)
```

```
## Warning in ks.test(excluded, "punif", 0.01, 0.5): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  excluded
## D = 0.024204, p-value = 2.842e-05
## alternative hypothesis: two-sided
```