

# Exercises

June 12, 2019

[1]: `library(tidyverse)`

```
Warning message:  
"package 'tidyverse' was built under R version 3.5.3"-- Attaching packages  
----- tidyverse 1.2.1 --  
v ggplot2 3.1.0      v purrr   0.2.5  
v tibble   2.1.1      v dplyr    0.7.6  
v tidyr    0.8.1      v stringr  1.4.0  
v readr    1.1.1      vforcats  0.3.0  
Warning message:  
"package 'ggplot2' was built under R version 3.5.3"Warning message:  
"package 'tibble' was built under R version 3.5.3"Warning message:  
"package 'stringr' was built under R version 3.5.3"-- Conflicts  
----- tidyverse_conflicts() --  
x dplyr::filter() masks stats::filter()  
x dplyr::lag()   masks stats::lag()
```

## 0.1 1. Define a vector $t$ with numbers from 0 to 5 by 0.01 Tabulate values of functions:

$f_n(t) = 1/n * \sin(2 * \pi * n * t)$  for  $n$  from 1 to 20. Plot functions  $f_1, f_1 + f_2, f_1 + f_2 + f_3, \dots$

[2]:

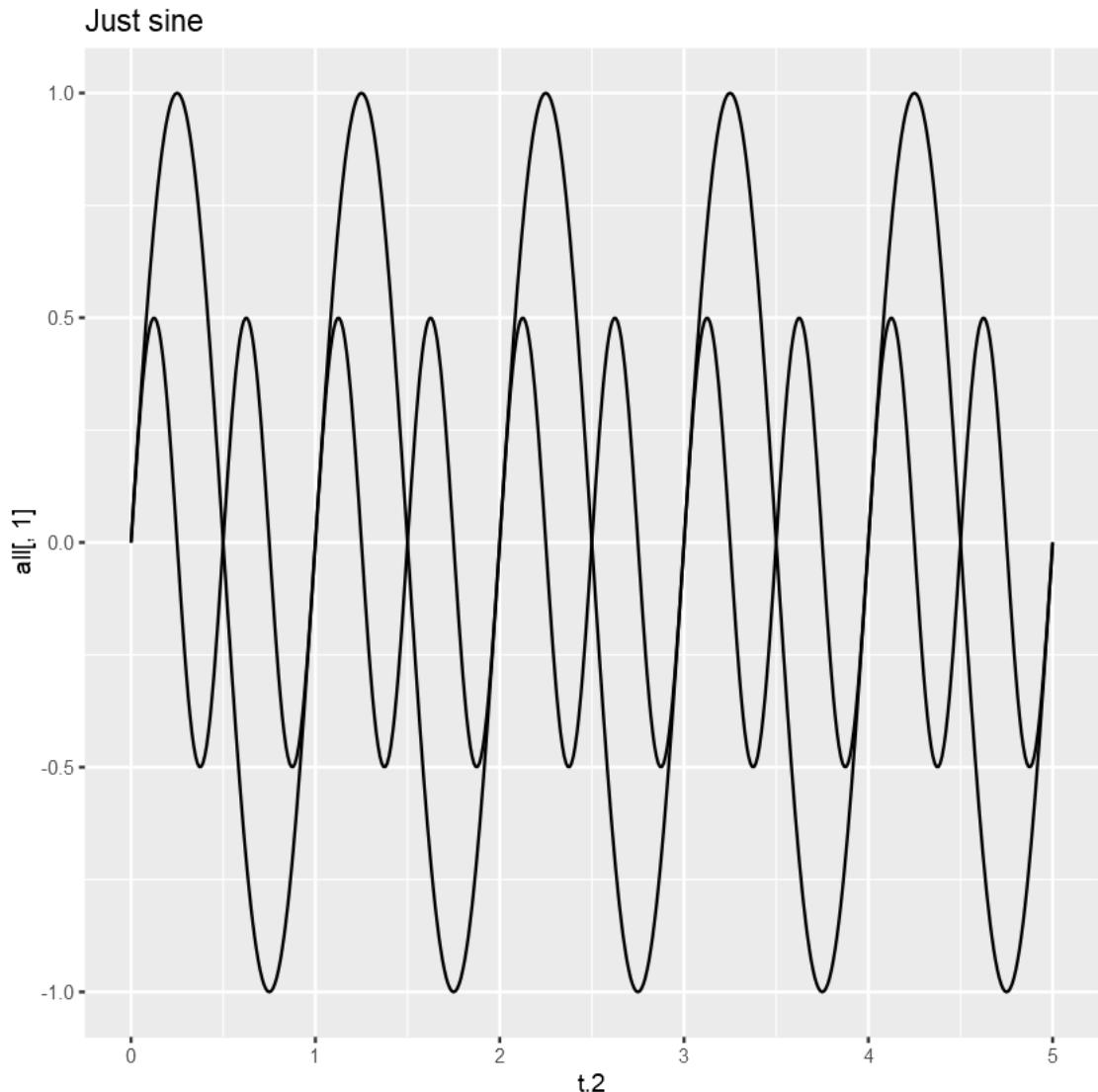
```
t.2 <- seq(0, 5, 0.01)  
n <- c(1:20)  
n <- diag(n)  
t <- matrix(t.2, length(t.2), 20)  
all <- sin(t %*% n * pi * 2)  
for (i in 1:20) {  
  all[,i] <- all[,i] * 1/i  
}  
  
plot <- ggplot()  
for (i in 1:10){  
  if (i>1) {  
    geom.line <- geom_line(aes(x = t.2, y = rowSums(all[,1:i])))  
    plot <- plot + geom.line  
  } else {  
    print(ggplot() + geom.line + ylab("y"))  
  }
```

```

        plot <- plot + geom_line(aes(x = t.2, y = all[,1]))
#      print(ggplot() + geom_line(aes(x = t.2, y = all[,1])) + ylab("y"))
    }
}
# plot <- plot + ggtitle("All on one")
# print(plot)
ggplot() + geom_line(aes(x = t.2, y = all[,1])) +
geom_line(aes(x = t.2, y = all[,2])) + ggtitle('Just sine')

# wave.1 <- sin(2 * pi * f.1 * t)
# ft.1 <- fft(wave.1)
# my.data <- data.frame(cbind(t[0:50], 2 * Mod(ft.1)[0:50]))
# names(my.data)[1] <- "x"
# names(my.data)[2] <- "y"
# ggplot(data = my.data, aes(x, y)) + geom_col()

```



**0.2 Generate data for two periods of triangular wave. Plot it. Calculate FFT and plot the complex module vs. frequency. Use the template from the NOTES part. Do it also for square wave and random noise signal (`rnorm()`).**

```
[3]: plot.frequency.spectrum <-function(X.k, xlims=c(0,length(X.k)/2)) {
  #horizontal axis from zero to Nyquist frequency
  #we plot complex modules of data vs. index numbers
  plot.data <-cbind(0:(length(X.k)-1), Mod(X.k))

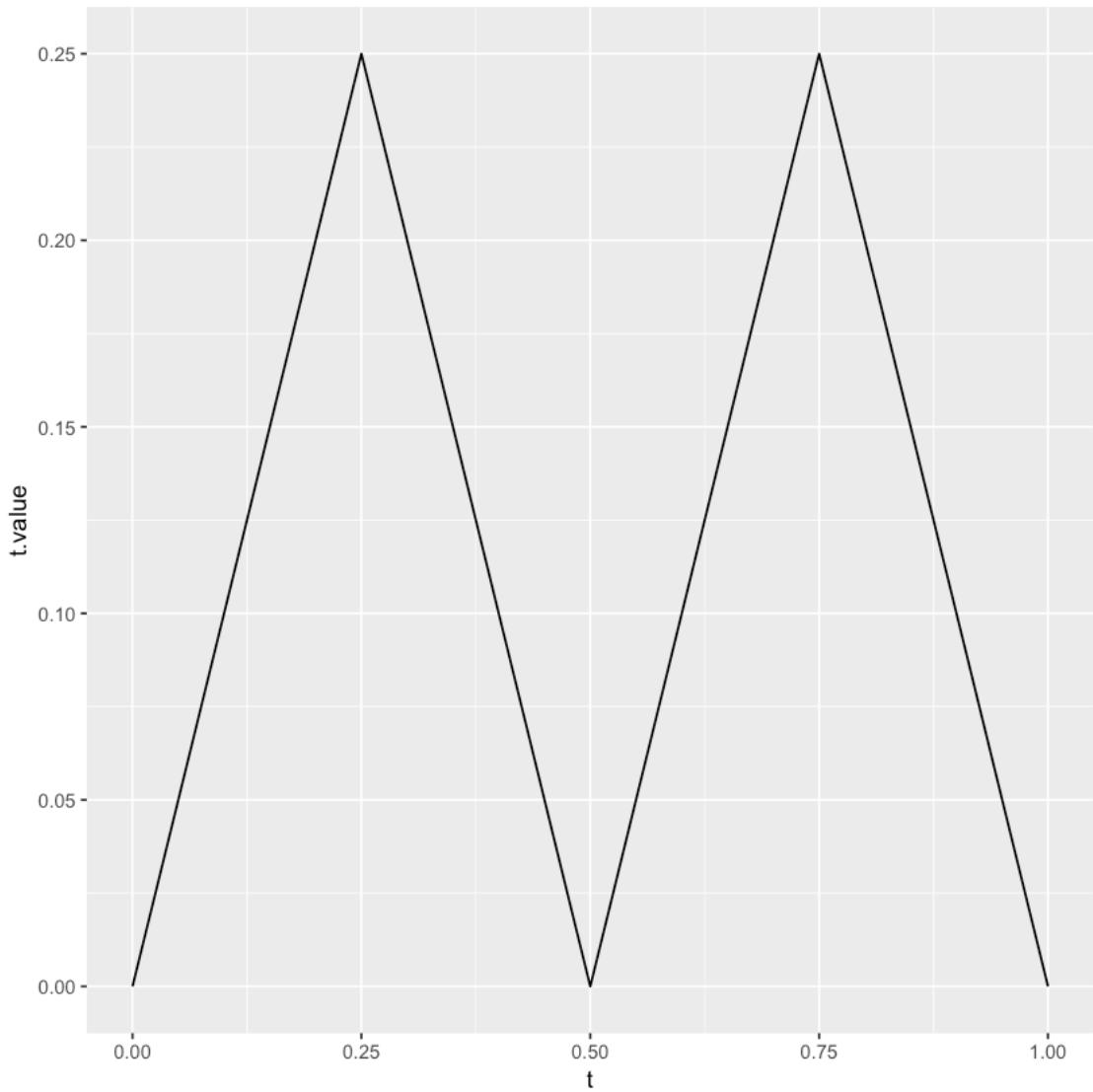
  # TODO: why this scaling is necessary?
  plot.data[2:length(X.k),2] <- 2*plot.data[2:length(X.k),2]

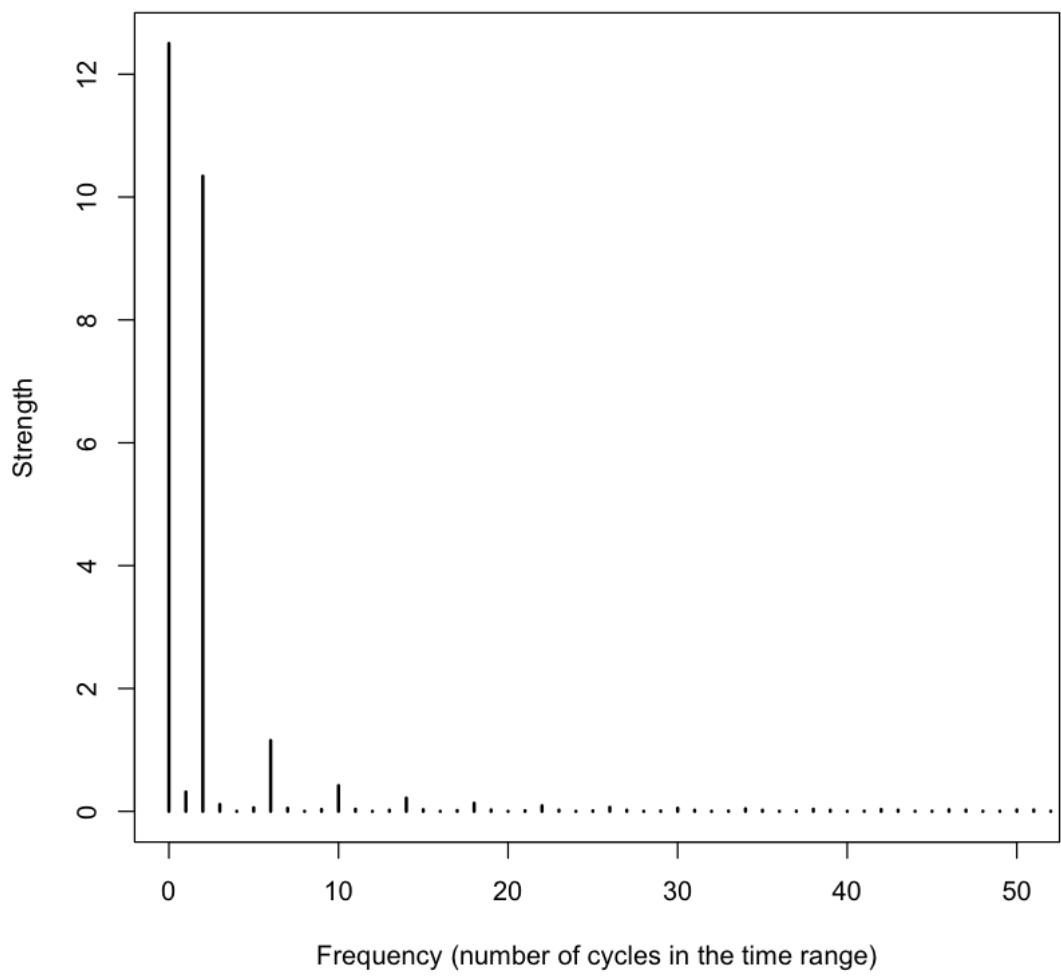
  plot(plot.data, t="h", lwd=2, main="", xlab="Frequency (number of cycles in
  →the time range)",
       ylab="Strength", xlim=xlims,
       ylim=c(0,max(Mod(plot.data[,2])))
  )
}
```

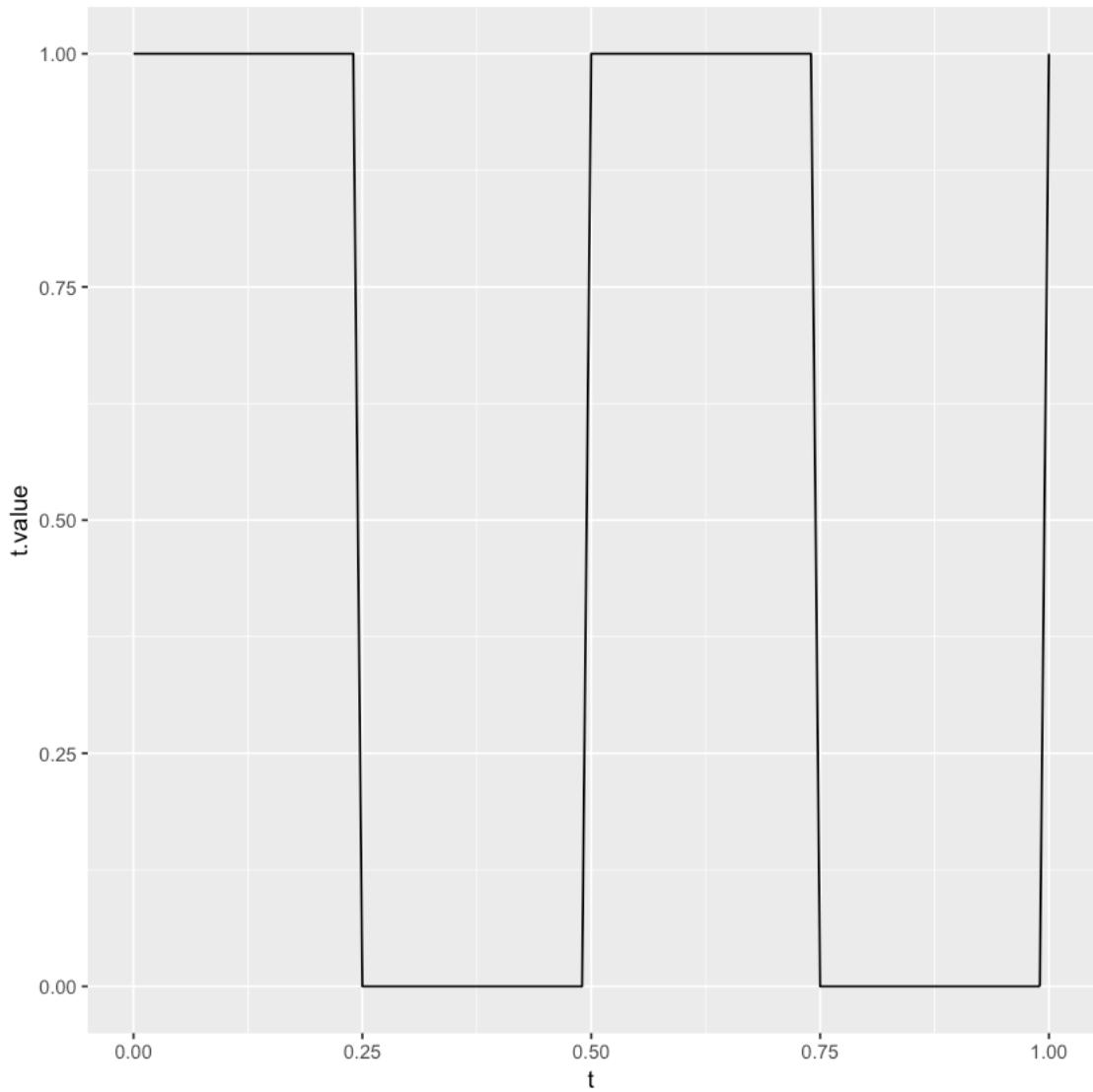
```
[4]: t <- seq(0, 1, 0.01)
t.value <- 0.25 - abs(t %% 0.5 -0.25)
ggplot() + geom_line(aes(t, t.value))
t <- fft(t.value)
plot.frequency.spectrum(t)

t <- seq(0, 1, 0.01)
t.value <- replace(replace(t , (t %% 0.5) < 0.25, 1), (t %% 0.5) >= 0.25, 0)
ggplot() + geom_line(aes(t, t.value))
t <- fft(t.value)
plot.frequency.spectrum(t)

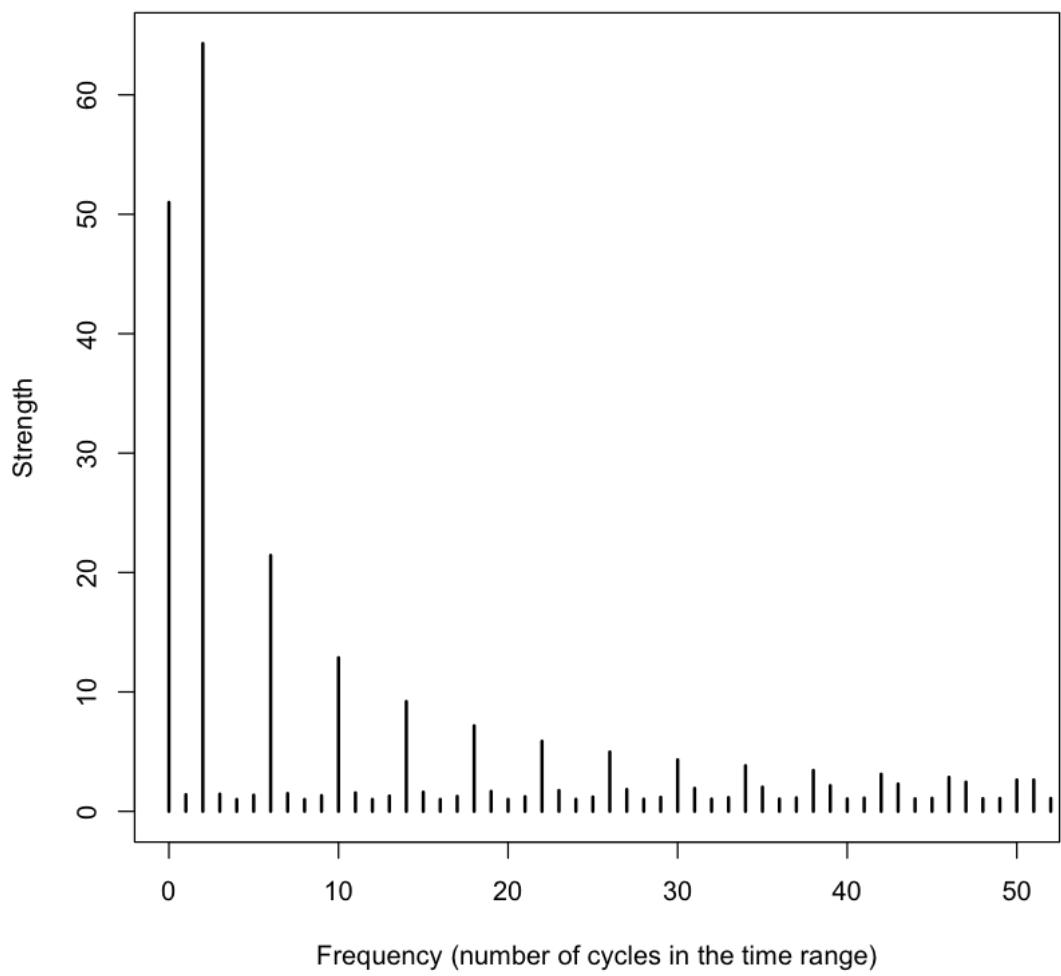
t <- seq(0, 1, 0.01)
length(t)
t.value <- rnorm(101,0.5,0.5)
ggplot() + geom_line(aes(t, t.value))
t <- fft(t.value)
plot.frequency.spectrum(t)
```

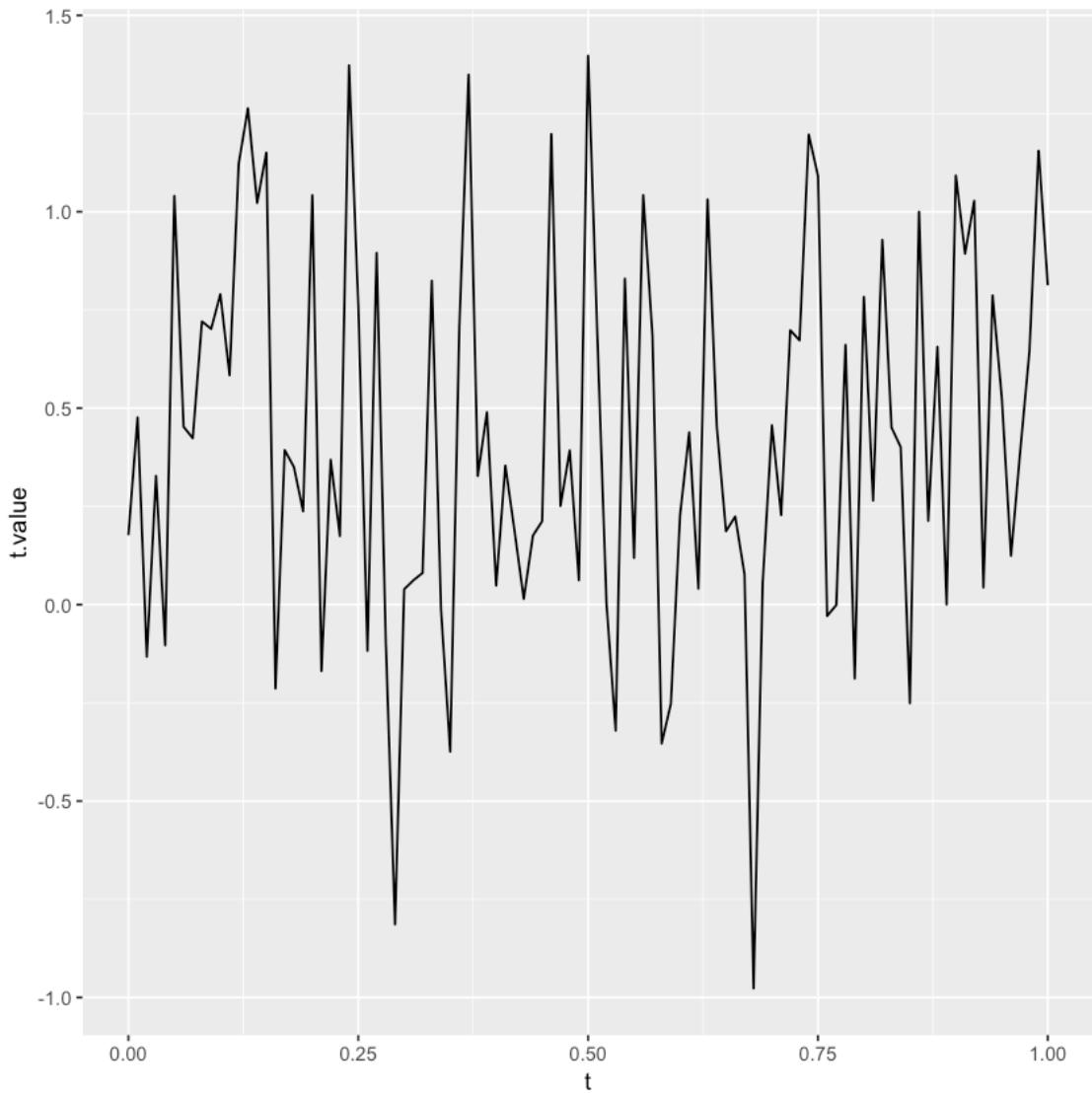


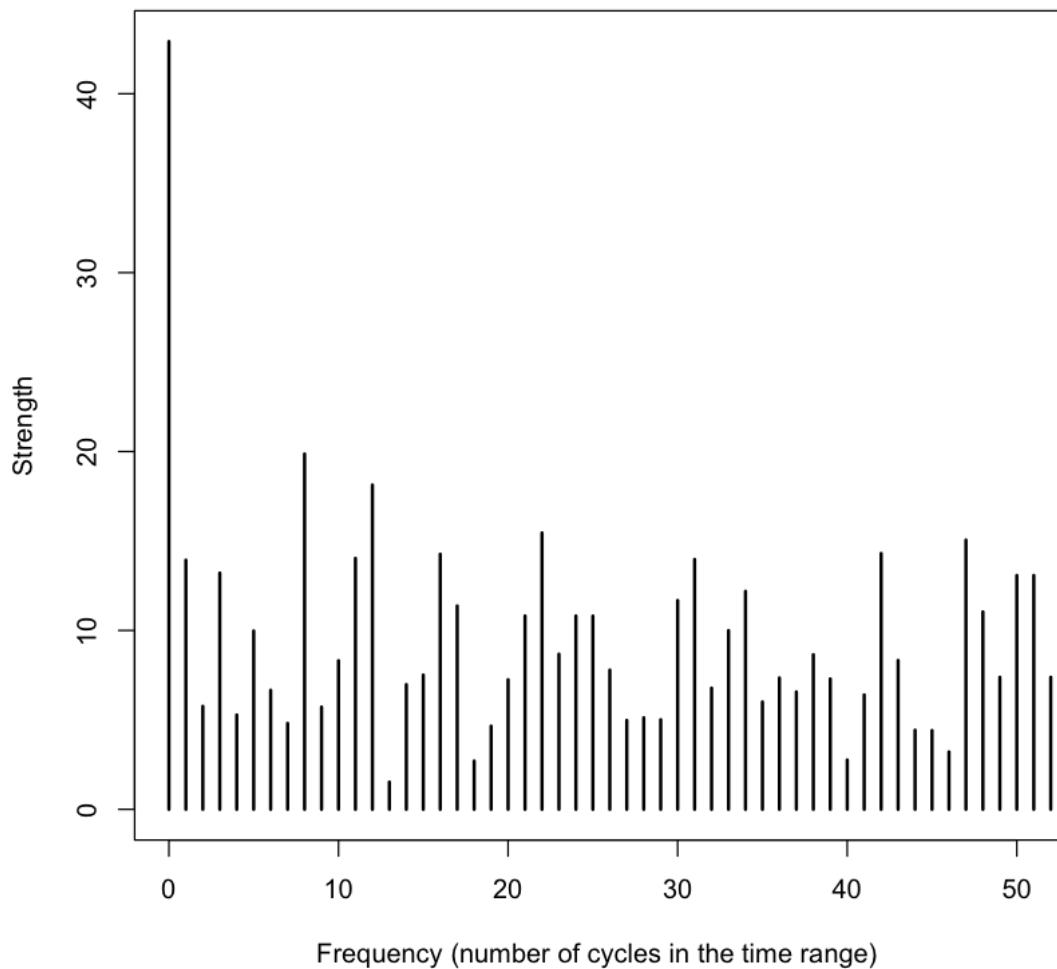




101





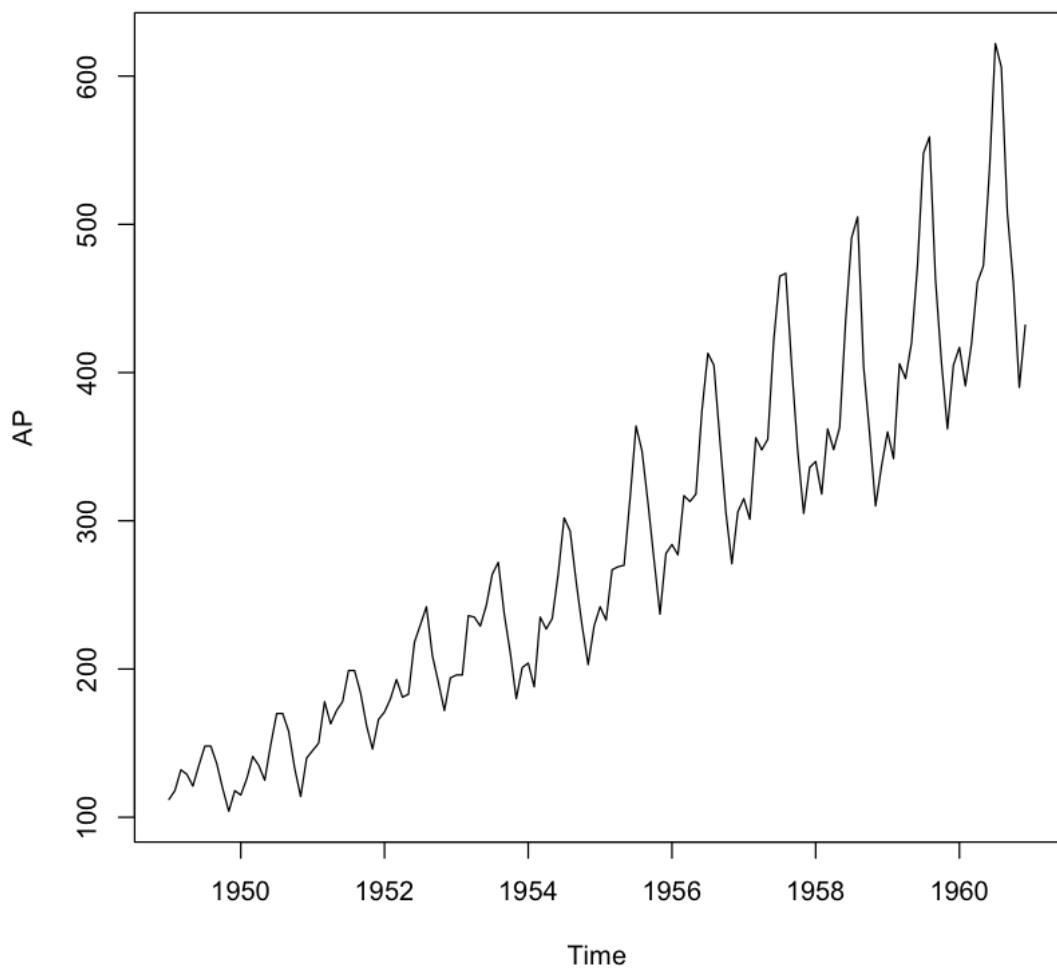


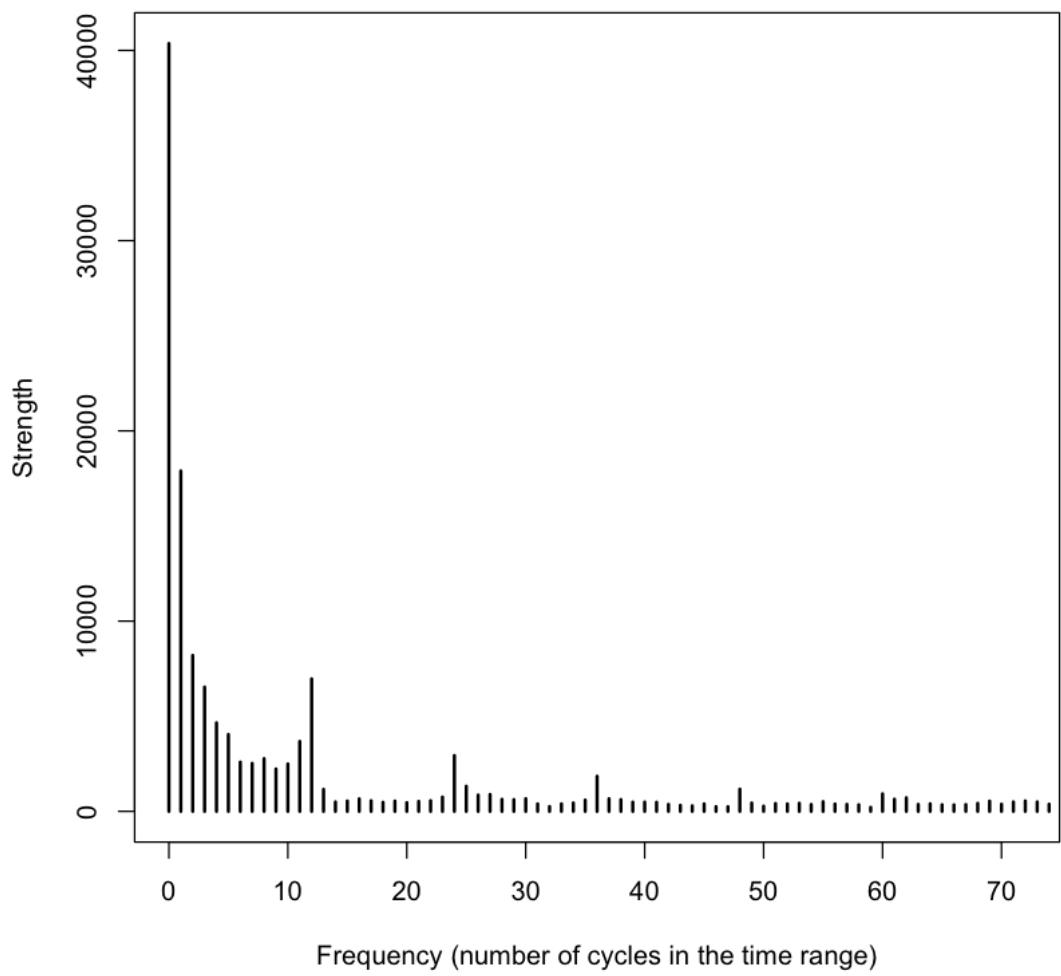
```
[5]: AP <-AirPassengers
print(AP)
plot(AP)
class(AP)
start(AP)
end(AP)
frequency(AP)
raw.data <-AP[1:length(AP)]
class(raw.data)
AP.fft <- fft(raw.data)
plot.frequency.spectrum(AP.fft)
raw.data <- raw.data - mean(raw.data)
AP.fft <- fft(raw.data)
```

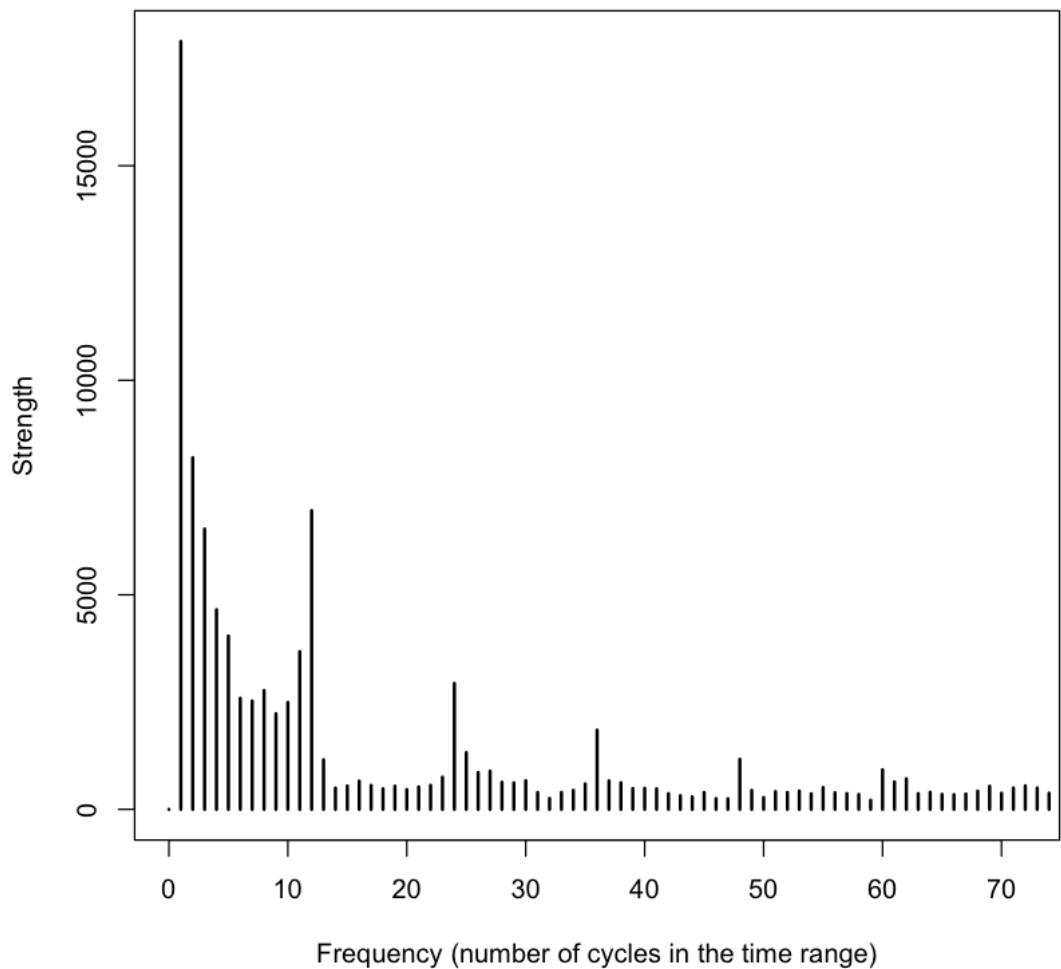
```
plot.frequency.spectrum(AP.fft)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

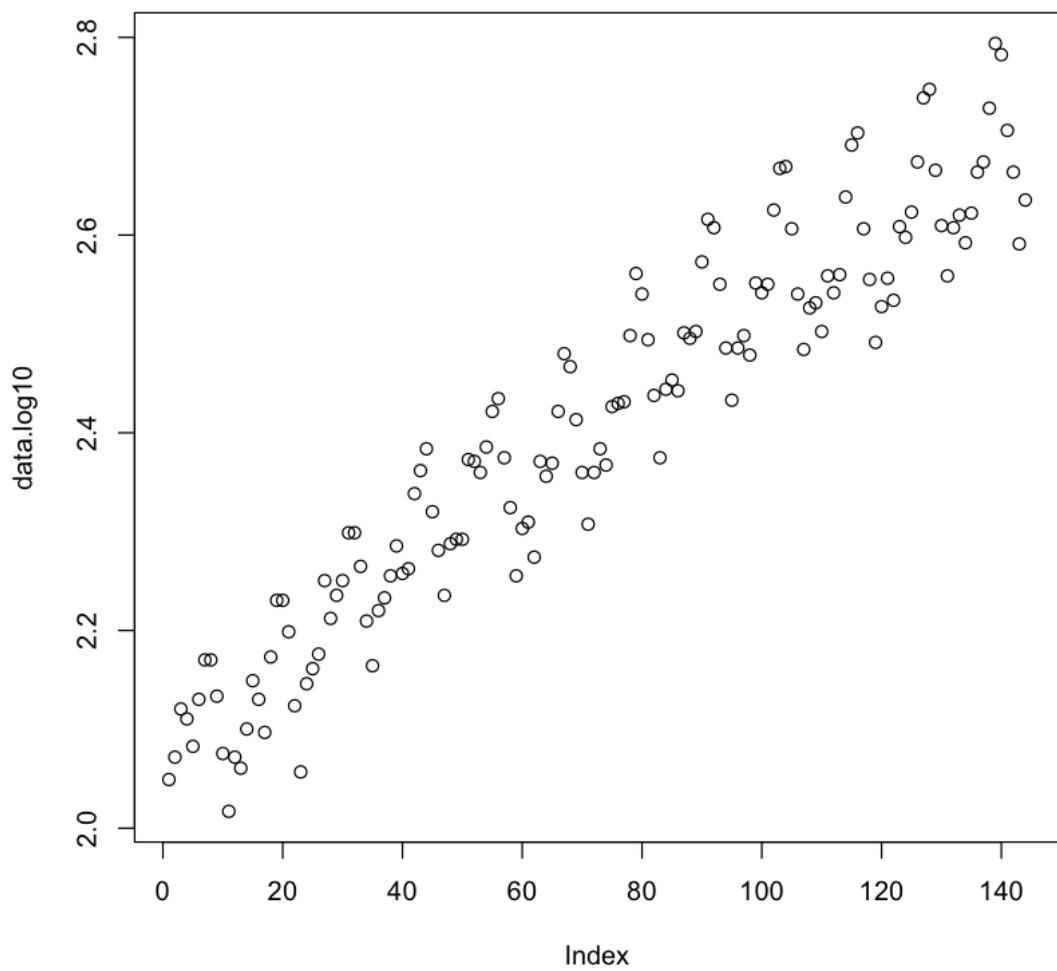
```
'ts'  
1. 1949 2. 1  
1. 1960 2. 12  
12  
'numeric'
```

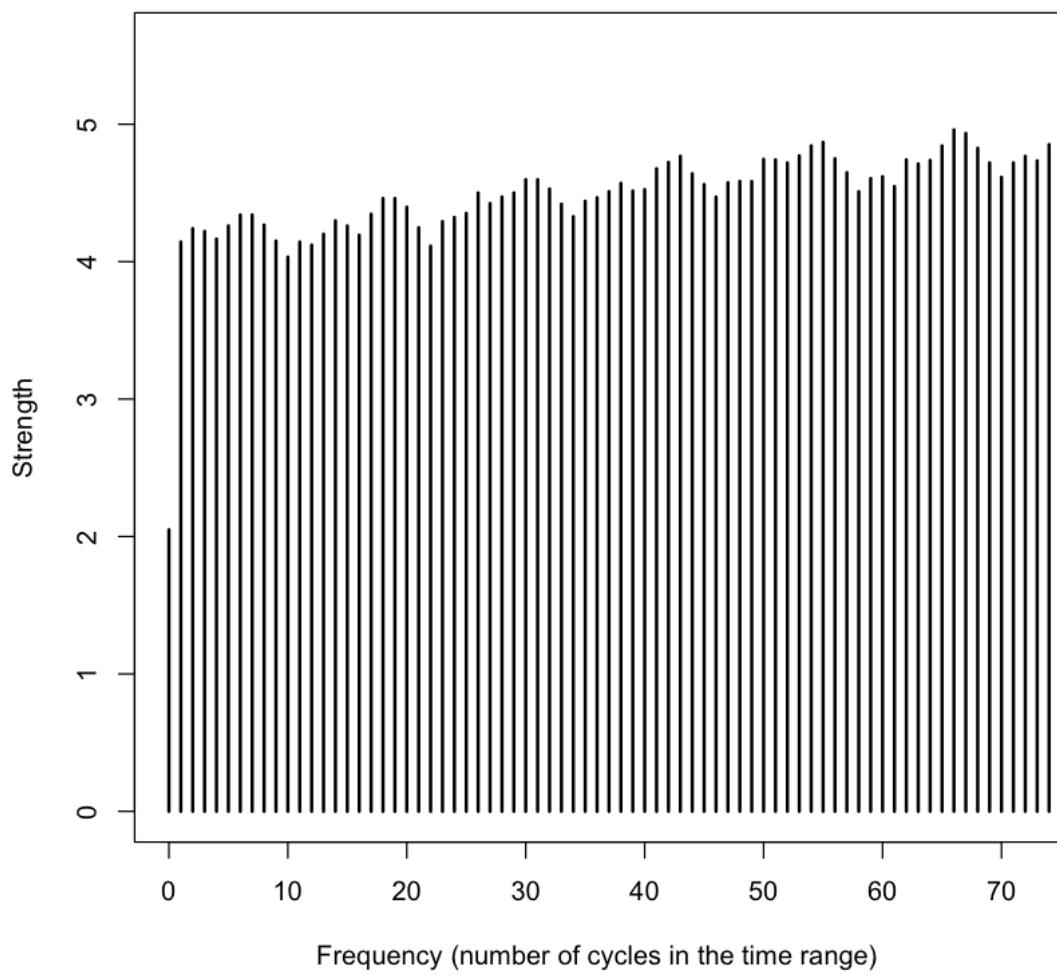


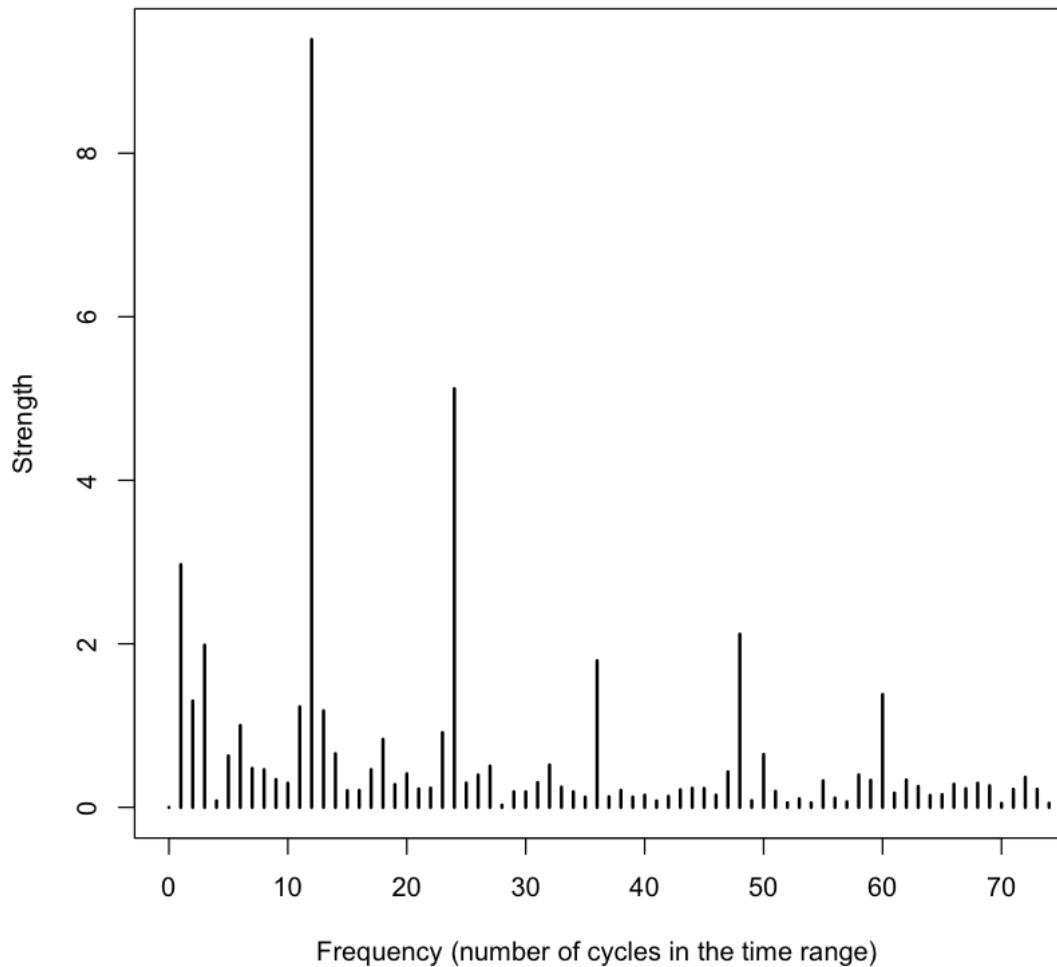




```
[8]: air.passengers.data <- AP[1:length(AP)]
data.log10 <- log10(air.passengers.data)
plot(data.log10)
# plot.frequency.spectrum(data.log10)
t.var <- 0:(length(data.log10)-1)
trend <- lm(data.log10 ~ t.var)
data.log10.detrended <- trend$residuals
ft.data.log10.detrended <- fft(data.log10.detrended);
plot.frequency.spectrum(ft.data.log10.detrended)
```







```
[12]: t <- seq(1,10, 0.2)
head(t,5)
format(round(sum(t*t), 10), nsmall=10)
t.fft <- Mod(fft(t))
# format(round(sum(t.fft * t.fft) / length(t), 10), nsmall = 10)
# approximately equal
sum(t*t) == sum(t.fft * t.fft) / length(t)
a <- sum(abs(t)^2)
b <- sum(abs(t.fft)^2 )*(1/length(t.fft))
a <= b
```

1.1 2. 1.2 3. 1.4 4. 1.6 5. 1.8

'1715.8000000000'

'1715.8000000000'

```

FALSE
TRUE
[9]: set.seed(1113)
data.original <- runif(10)

left.side <- sum(abs(data.original)^2)
print(paste("Left side of equation: ", left.side))

data.transformed <- fft(data.original)
N <- length(data.transformed)
right.side <- (1/N) * sum(abs(data.transformed)^2)
print(paste("Right side of equation: ", right.side))

```

```

[1] "Left side of equation: 1.93709403869464"
[1] "Right side of equation: 1.93709403869464"

```

```

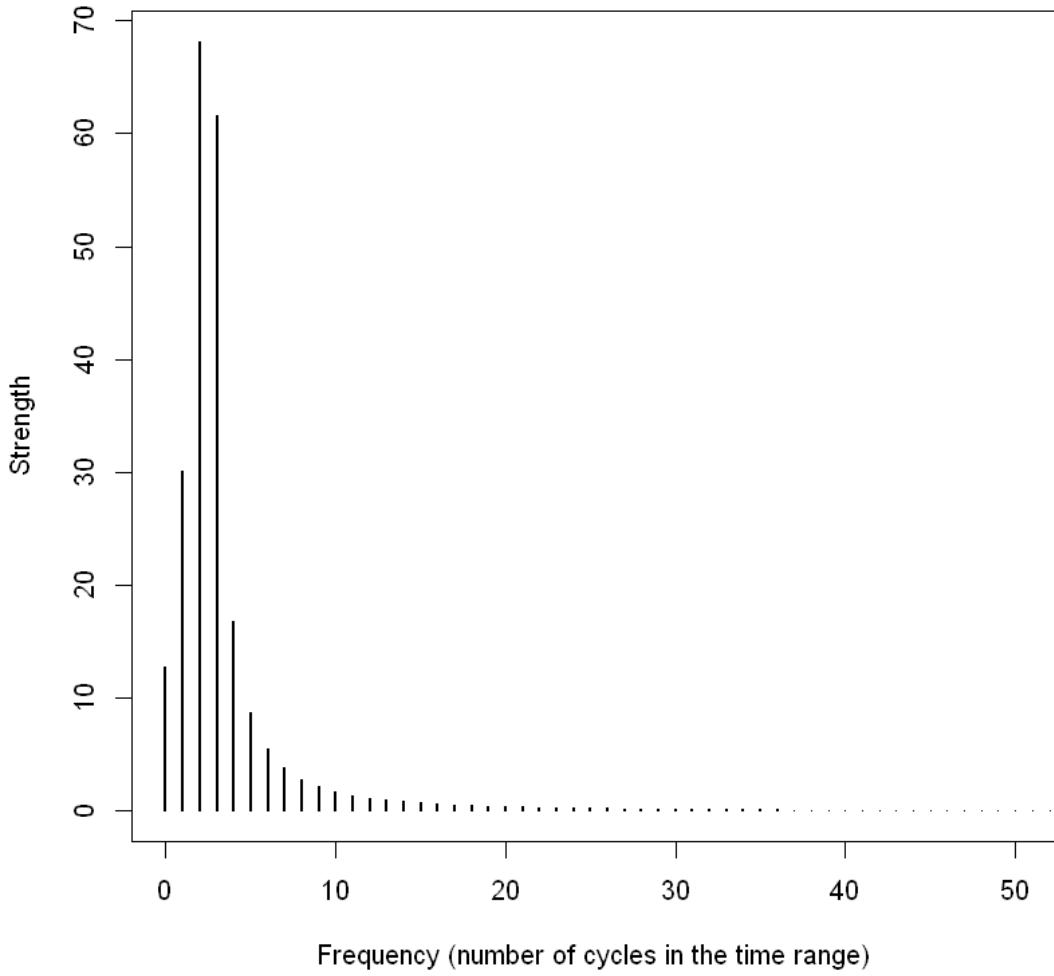
[3]: plot.frequency.spectrum <- function(X.k, xlims=c(0,length(X.k)/2)) {
  #horizontal axis from zero to Nyquist frequency
  #we plot complex modules of data vs. index numbers
  plot.data <- cbind(0:(length(X.k)-1), Mod(X.k))

  # TODO: why this scaling is necessary?
  plot.data[2:length(X.k),2] <- 2*plot.data[2:length(X.k),2]

  plot(plot.data, t="h", lwd=2, main="", xlab="Frequency (number of cycles in_
  →the time range)",
       ylab="Strength", xlim=xlims,
       ylim=c(0,max(Mod(plot.data[,2]))))
}

tt <- seq(from=0, to=1, by=0.01)
#time in sec
freq <- 2.5      #wholecycles per second
mydata<-sin(2*pi*freq* tt)
ft.mydata <- fft(mydata)
plot.frequency.spectrum(ft.mydata)

```



```
[14]: my.data <- read.csv("multiTimeline.csv", stringsAsFactors = FALSE)
names(my.data)[1] <- 'M'
names(my.data)[2] <- 'N'
for (i in 1:nrow(my.data)){
  my.data$M[i] <- paste(my.data$M[i], '01', sep='-')
}
my.data$M <- as.Date(my.data$M, format = "%Y-%m-%d")
# xts(my.data$N, as.Date(my.data$M, format='%Y-%m'))
head(my.data,5)
tail(my.data, 5)
ggplot(data = my.data, aes(M, N)) + geom_line()
data.fft <- fft(my.data$N)
plot.frequency.spectrum(data.fft)
```

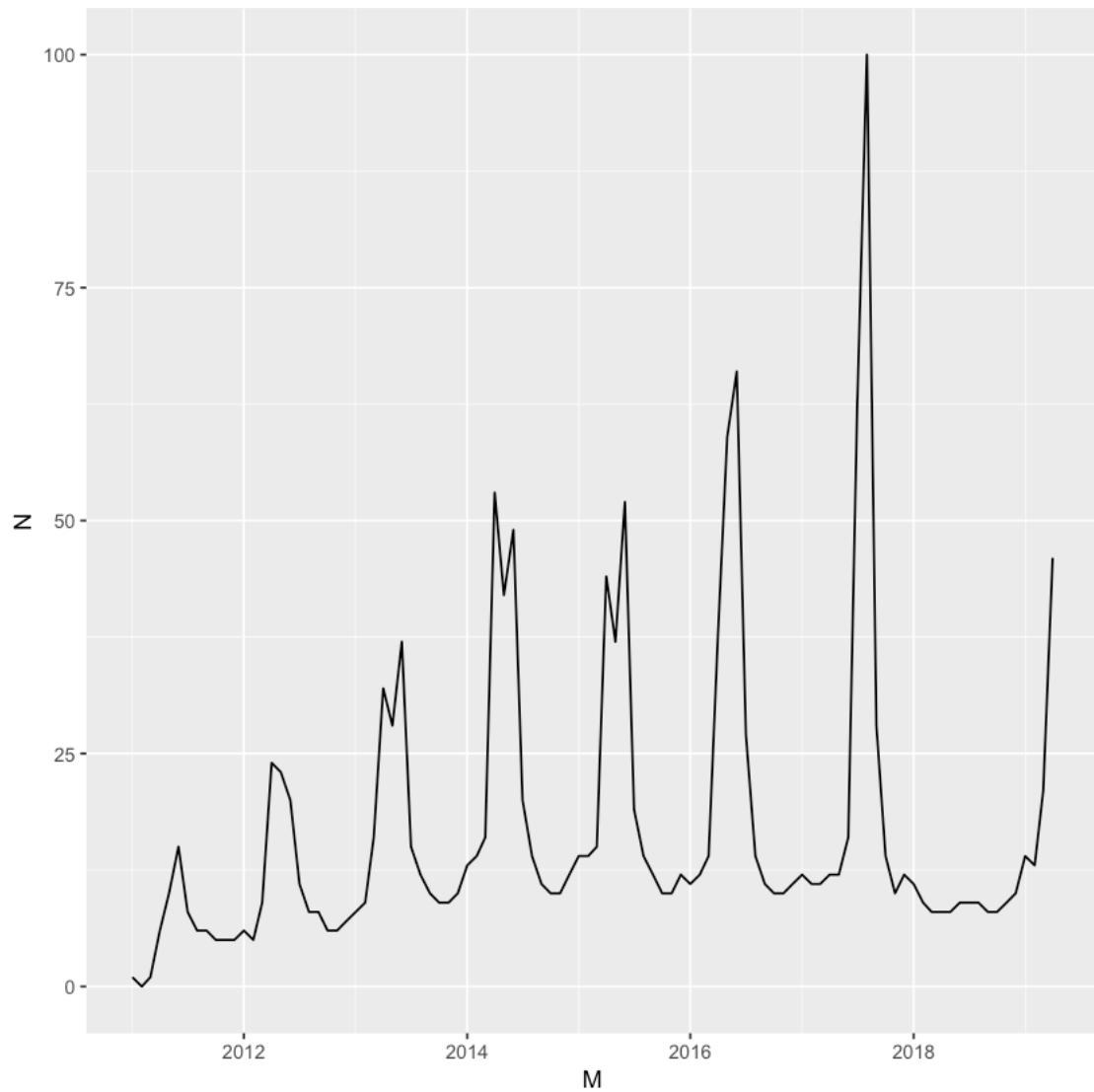
```

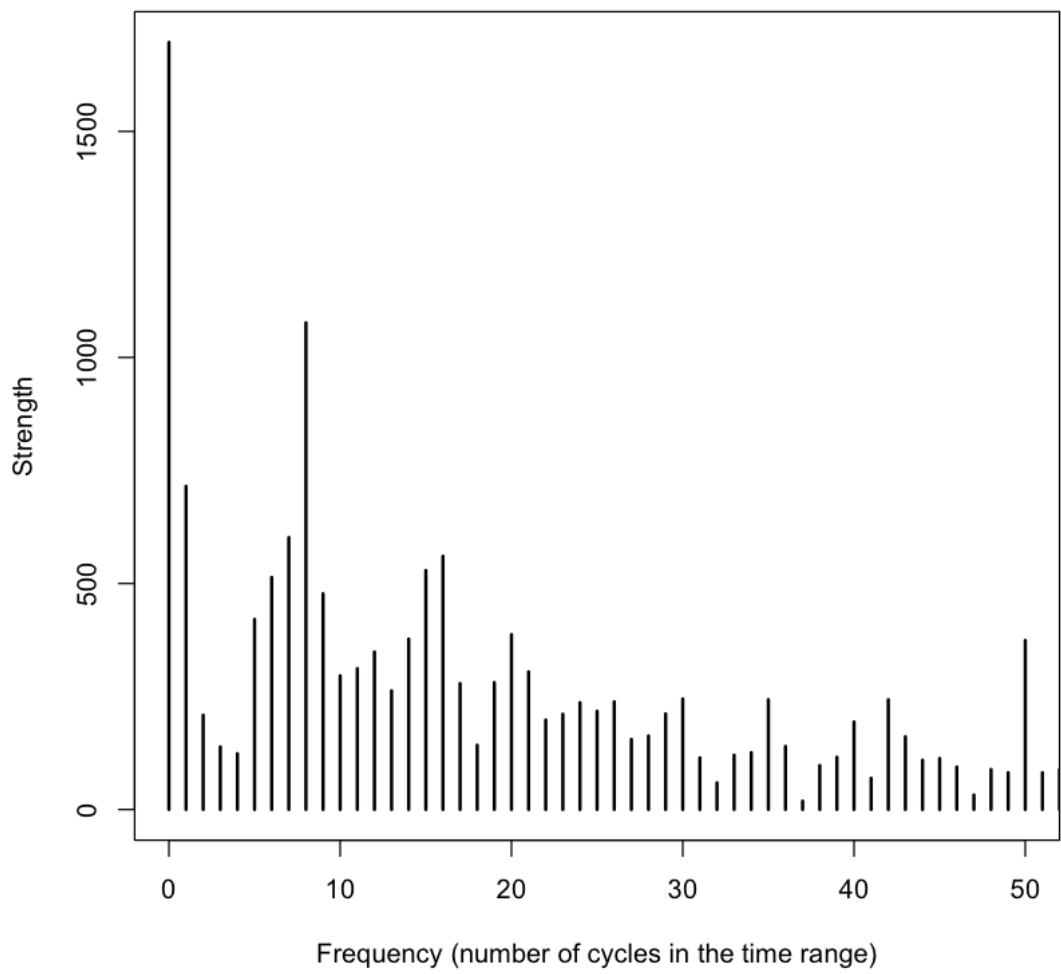
data.fft <- fft(my.data$N - mean(my.data$N))
plot.frequency.spectrum(data.fft)

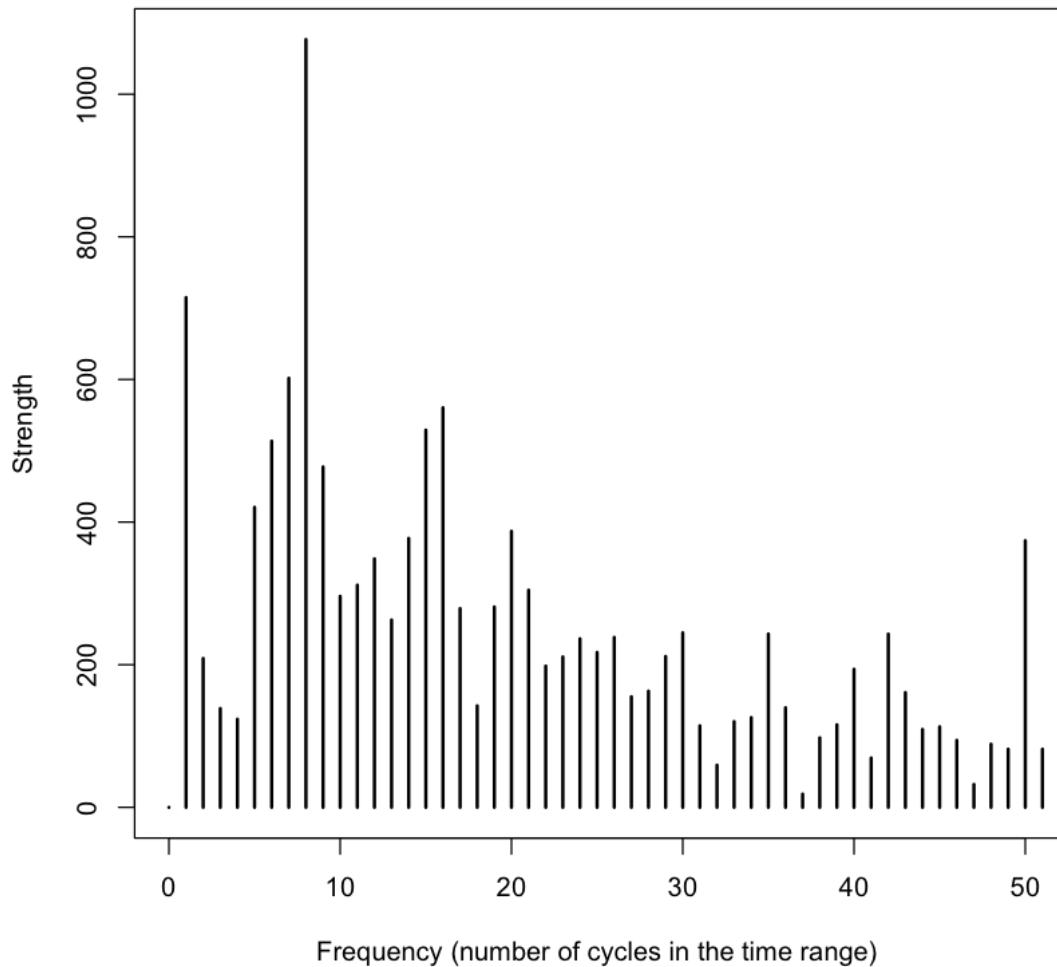
```

M	N
2011-01-01	1
2011-02-01	0
2011-03-01	1
2011-04-01	6
2011-05-01	10

M	N
96	2018-12-01
97	2019-01-01
98	2019-02-01
99	2019-03-01
100	2019-04-01







[ ]:

```
[343]: # This will give NA(s) in some locales; setting the C locale
## as in the commented lines will overcome this on most systems.
## lct <- Sys.getlocale("LC_TIME"); Sys.setlocale("LC_TIME", "C")
x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
z <- as.Date(x, "%d%b%Y")
## Sys.setlocale("LC_TIME", lct)
x
z
```

1. '1960-01-01' 2. '1960-02-01' 3. '1960-03-01' 4. '1960-04-13'  
 1. 1960-01-01 2. 1960-02-01 3. 1960-03-01 4. 1960-04-13

[ ]: