TimeSeries

Michael McCormack

March 12, 2018

Step 1: Get the data.

Download from website https://cdn.rawgit.com/mikejt33/DataViz/246c2026/data/flights.csv.gz Easiest to unzip locally then read in the data as a csv file (hint: read.table() is typically faster than read.csv)

```
flights <- read.table('flights.csv', header = TRUE, sep = ',')
```

Step 2: Prepare the data.

*Are there any null values?

Time series data needs to be over a regular time interval. Calculate the average departure delay time and/or average arrival delay time for each day of 2017.

```
arr_delay <- flights %>%
             arrange(FL_DATE) %>%
             group_by(FL_DATE) %>%
             summarise(mean(ARR_DELAY, na.rm = TRUE))
names(arr_delay) <- c('FL_DATE', 'AVE_DELAY')

dep_delay <- flights %>%
             arrange(FL_DATE) %>%
             group_by(FL_DATE) %>%
             summarise(mean(DEP_DELAY, na.rm = TRUE))
names(dep_delay) <- c('FL_DATE', 'AVE_DELAY')
```

If you like, compare average delay times for different carriers or different airports by creating multiple time series.

Step 3: Create a ts object of the data.

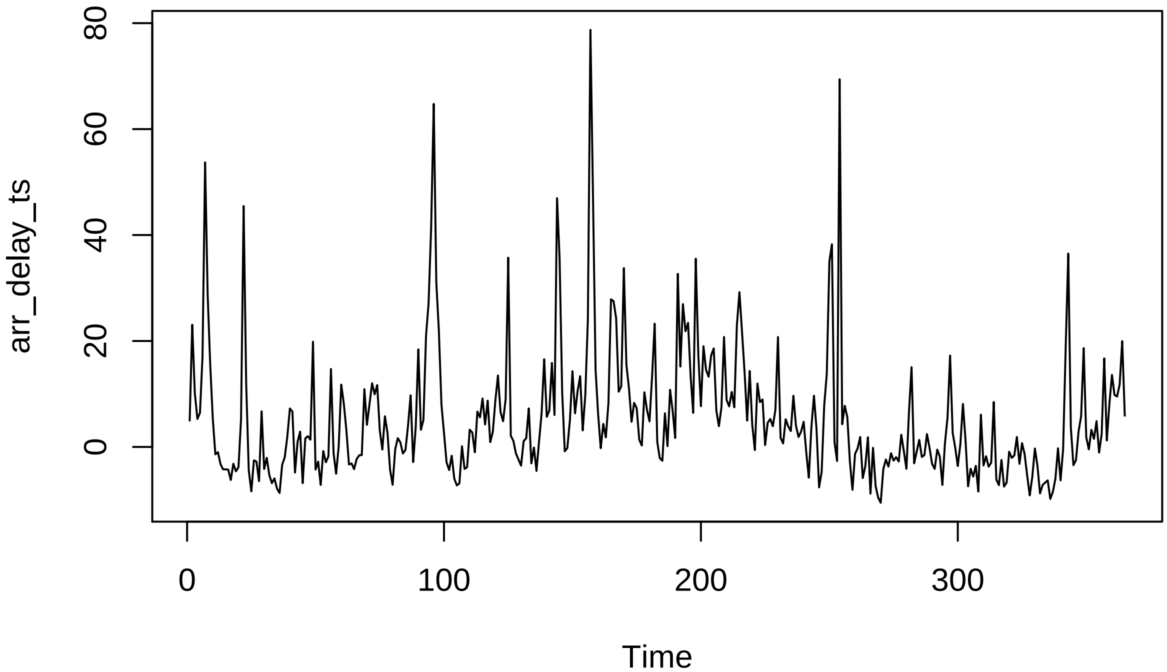
Refer to the slides for tips on how to do this.

```
arr_delay_ts <- ts(arr_delay$AVE_DELAY)
dep_delay_ts <- ts(dep_delay$AVE_DELAY)
```

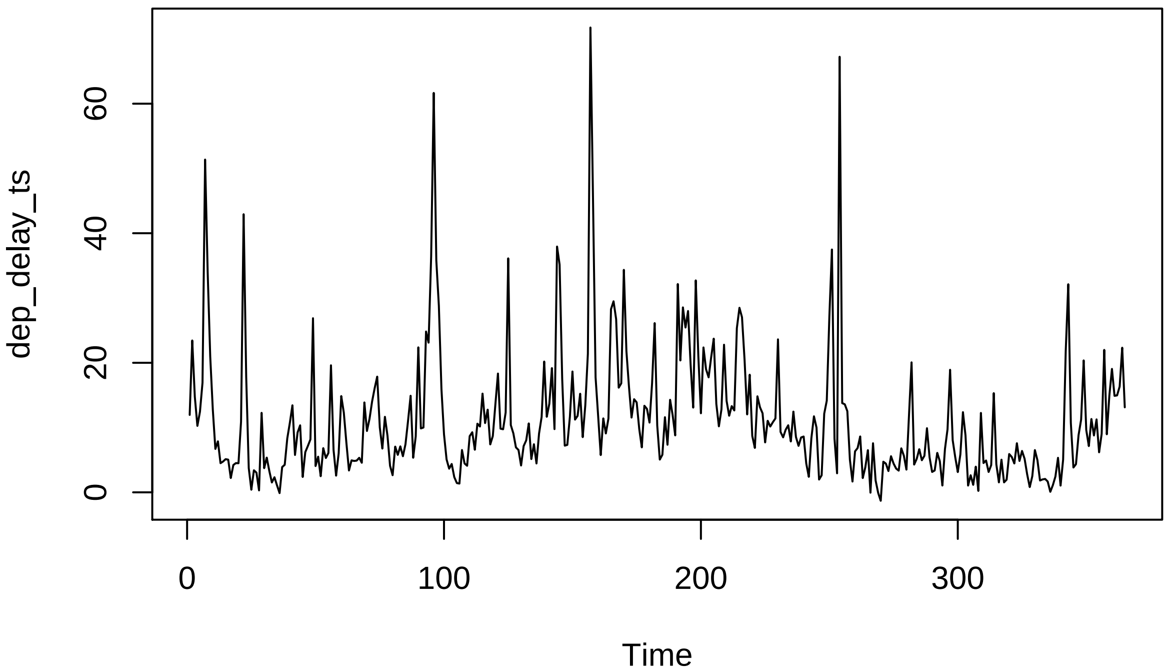
Step 4: Plot the time series using base package and ggplot (advanced).

Create a basic visualization of the time ser

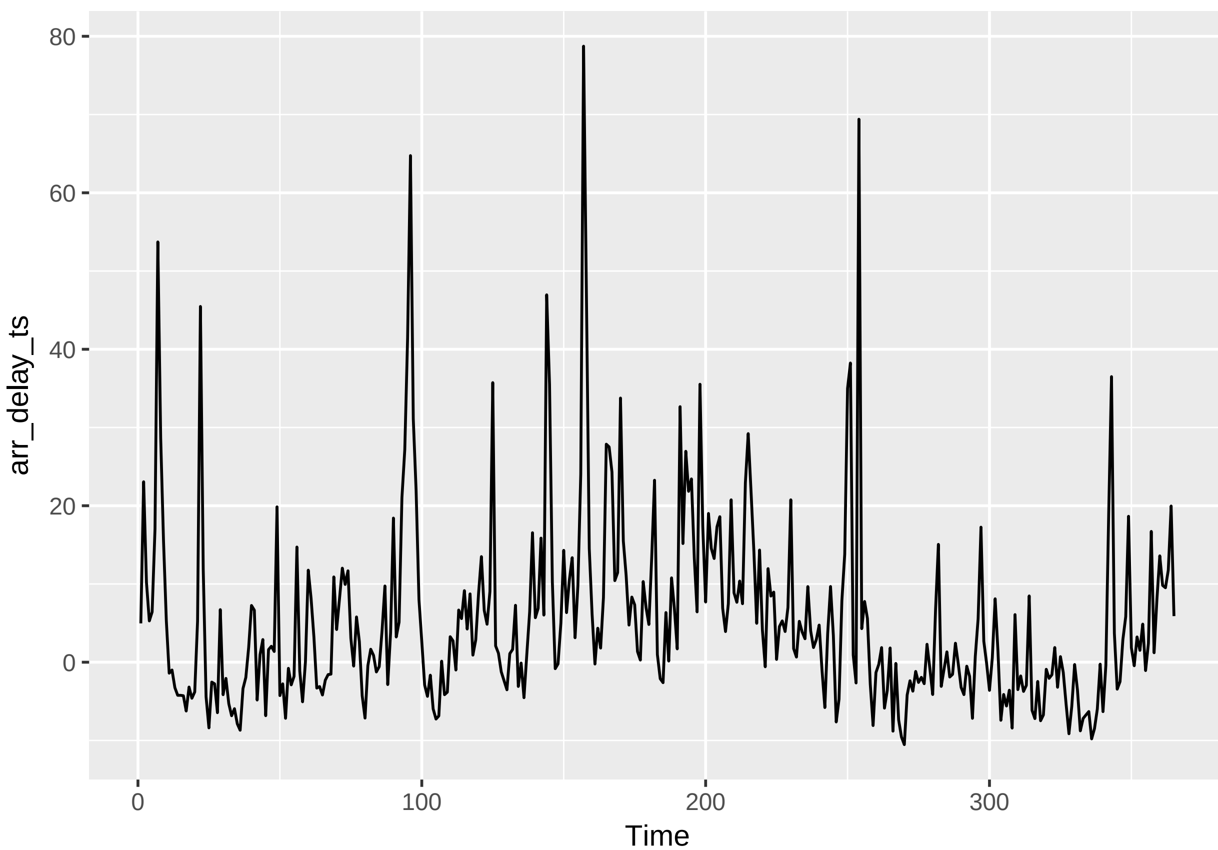
```
plot(arr_delay_ts)
```




```r
plot(dep_delay_ts)
```




```r
# Advanced Portion, requires ggfortify
autoplot(arr_delay_ts)
```

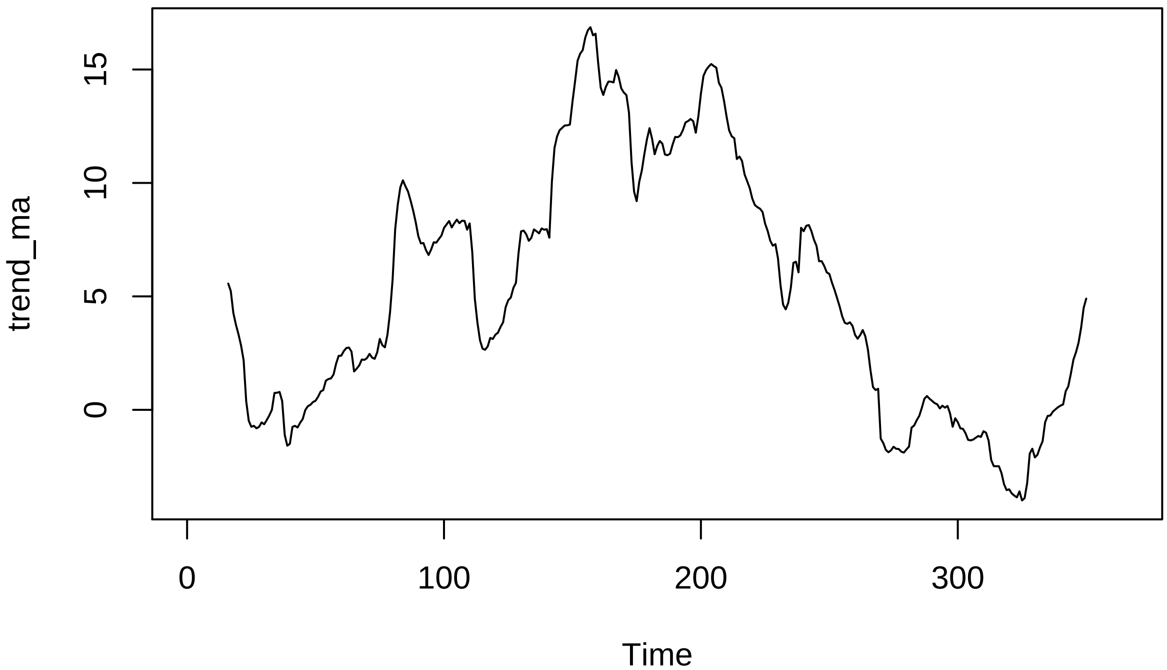


Step 5: Smooth the data to reduce noise and identify trends.

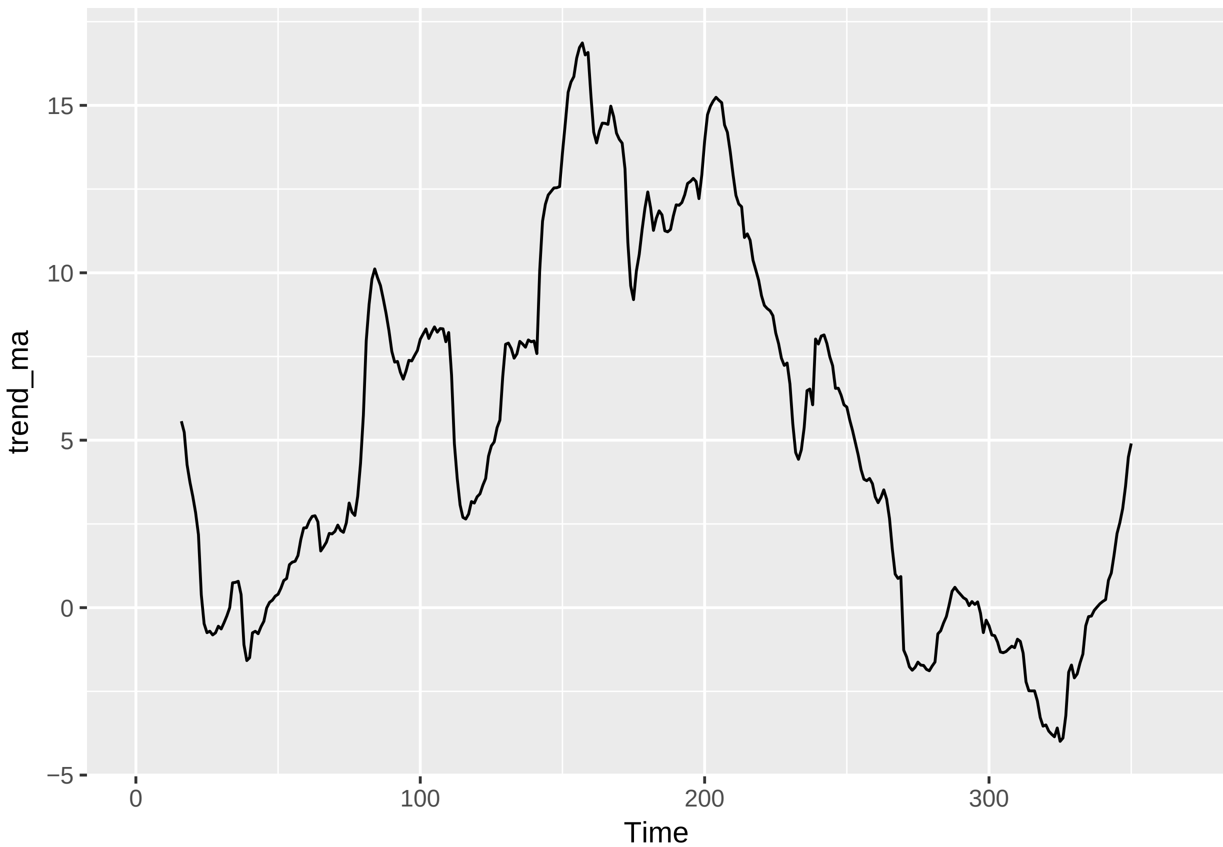
Create your own simple moving average for monthly data. Plot the smoothed data using base package. Plot both the originial and the smoothed data ggplot (advanced).

Hints
* good StackOverflow reference for moving average in R: https://stackoverflow.com/questions/743812/calculating-moving-average * watch out for functions that may have been masked by other packages * ggplot: may need to convert data to long format to plot mutliple series

```
# simple moving average for monthly data, n = is neighborhood size i.e. the number of point in each loc
moving_ave <- function(x,n){stats::filter(x,rep(1/n,n), sides=2)}
trend_ma <- moving_ave(arr_delay_ts,31)

# plot smoothed data using base package
plot(trend_ma)
```




```
# plot using ggplot
# autoplot, requires ggfortify
autoplot(trend_ma, ts.color = 'blue')
```



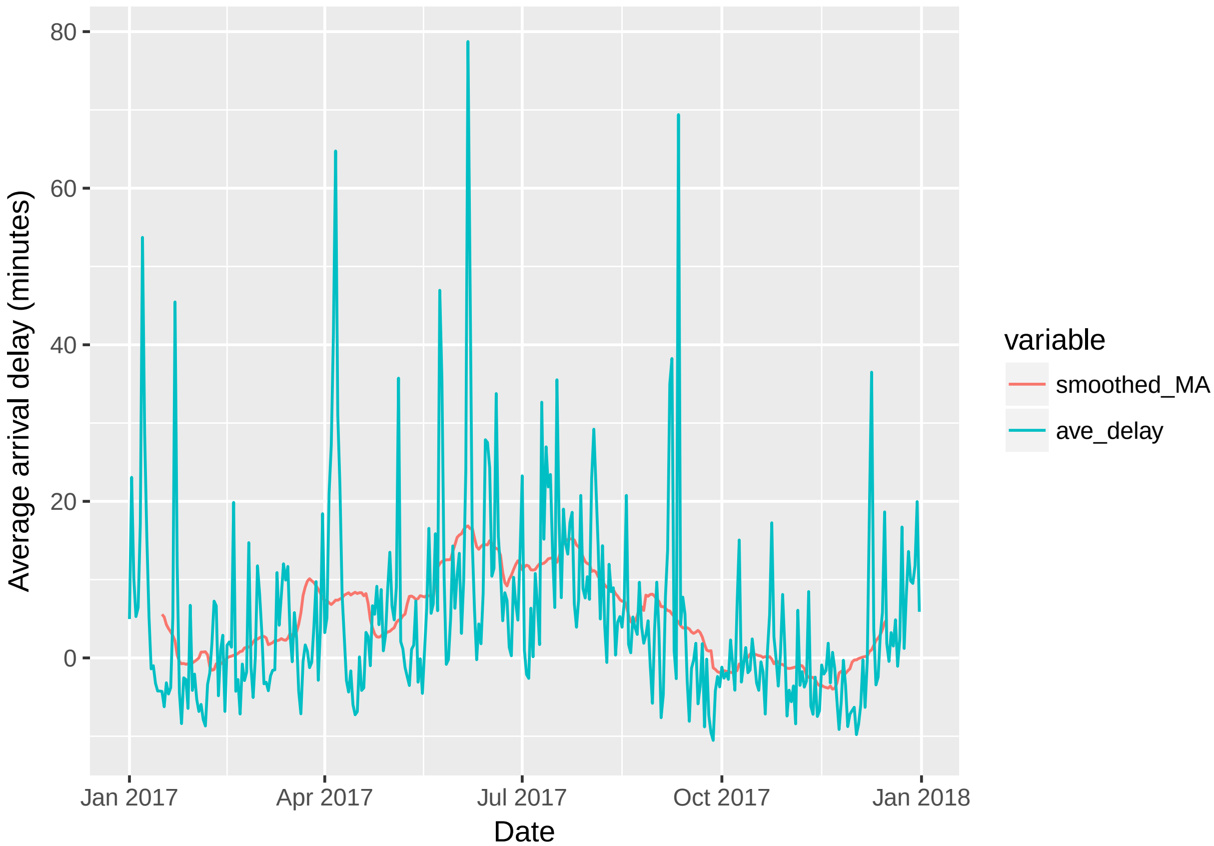

```
# creating data frame with dates to improve readability of plots
dates <- seq(as.Date('2017-01-01'), as.Date('2017-12-31'), by = 'day') # create sequence of dates
```

```r
plot_ts <- data.frame(smoothed_MA = trend_ma, ave_delay=as.numeric(arr_delay_ts),date=dates)

# melt into long format so both the origininal and smoothed data can be easily plotted on the same axes
plot_ts_melted <- melt(plot_ts, id='date')
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```r
ggplot(plot_ts_melted) + geom_line(aes(x = date, y  = value, col = variable)) + labs(x = 'Date', y = 'A
```

```
## Warning: Removed 30 rows containing missing values (geom_path).
```



Questions
1. How does the neighborhood size, i.e. the number of points in each localized subset, affect the amount of smoothing?
2. What happened to endpoints of the smoothed data?

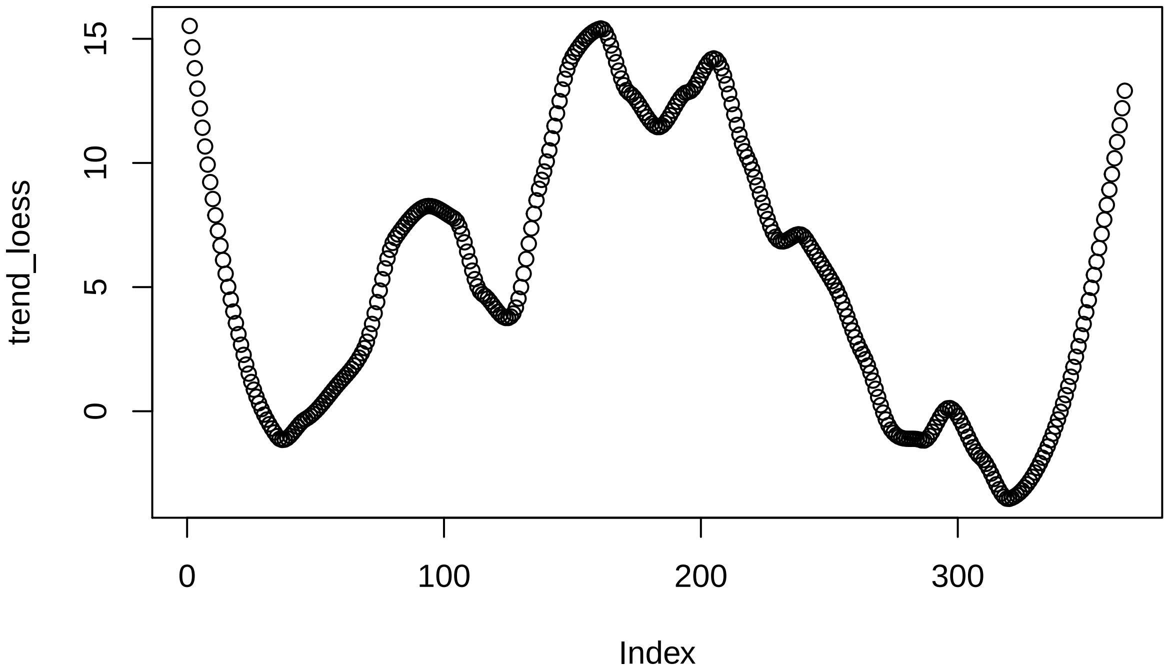
Answers 1. Increasing window size increases the amount of smoothing. 2. The first and last (n-1)/2 data points are lost, where n is odd and denotes the neighborhood size. The moving average at those points in time don't have access to a full localized subset of size n and are set as NA's.

Advanced: Smooth the same data using Local Regression (loess). Plot smoothed data using base package. Plot all three series (original, smoothed by MA, and smoothed by loess) using ggplot (advanced).

Hint
* loess() requires all predictors to be numerical so dates cannot be used

Try different values for the span argument and see how it affects the amount of smoothing.

```
# create index variable
arr_delay$index <- 1:nrow(arr_delay)
# get the loess model, span is the proportion of data used in each localized subset
loess_model <- loess(AVE_DELAY ~ index, data=arr_delay, span=0.25)
# smooth the data by using the predict() function
trend_loess <- predict(loess_model)

# plot the smoothed data using base package
plot(trend_loess)
```

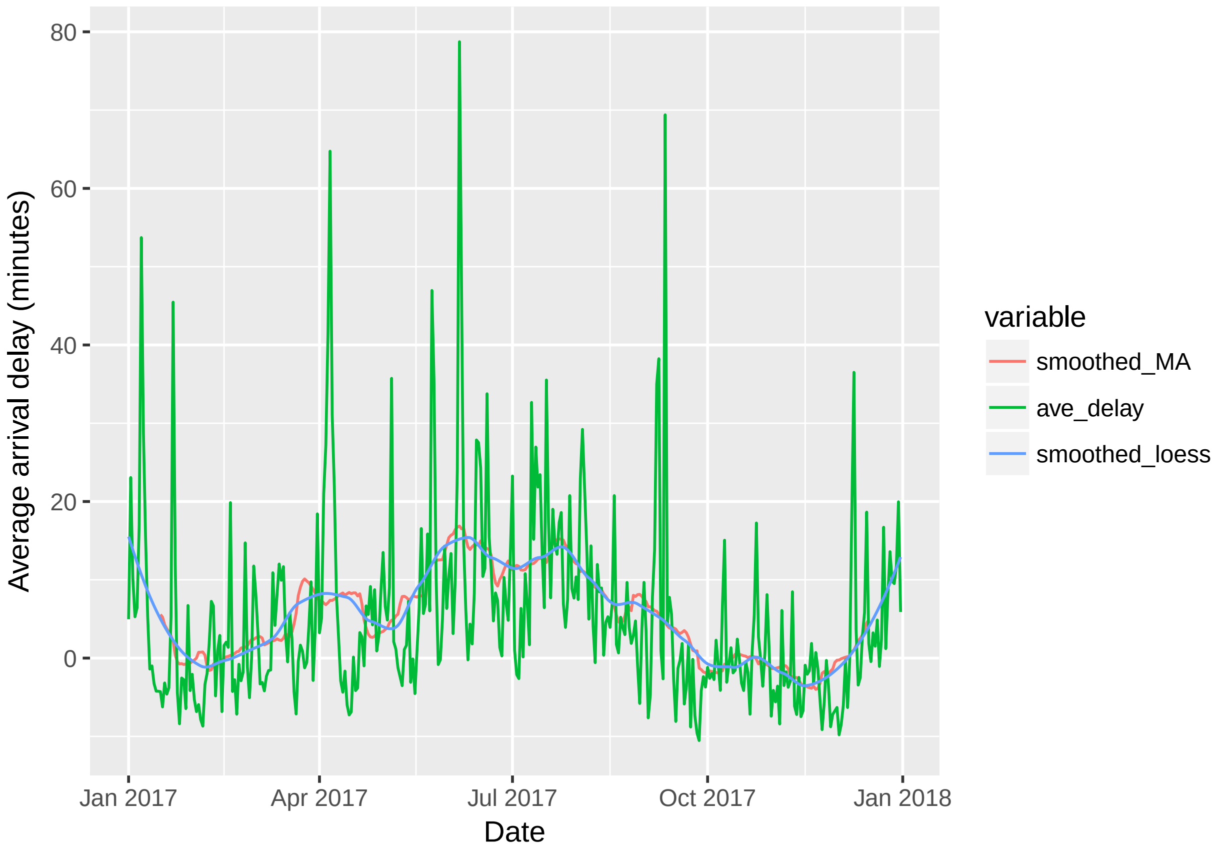



```
# plot using ggplot

# creating data frame with dates to improve readability of plots
plot_ts <- cbind(plot_ts, smoothed_loess = trend_loess)

# melt into long format so both the origininal and smoothed data can be plotted on the same axes
plot_ts_melted <- melt(plot_ts, id='date')
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```
ggplot(plot_ts_melted) + geom_line(aes(x = date, y  = value, col = variable)) + labs(x = 'Date', y = 'A
```

```
## Warning: Removed 30 rows containing missing values (geom_path).
```



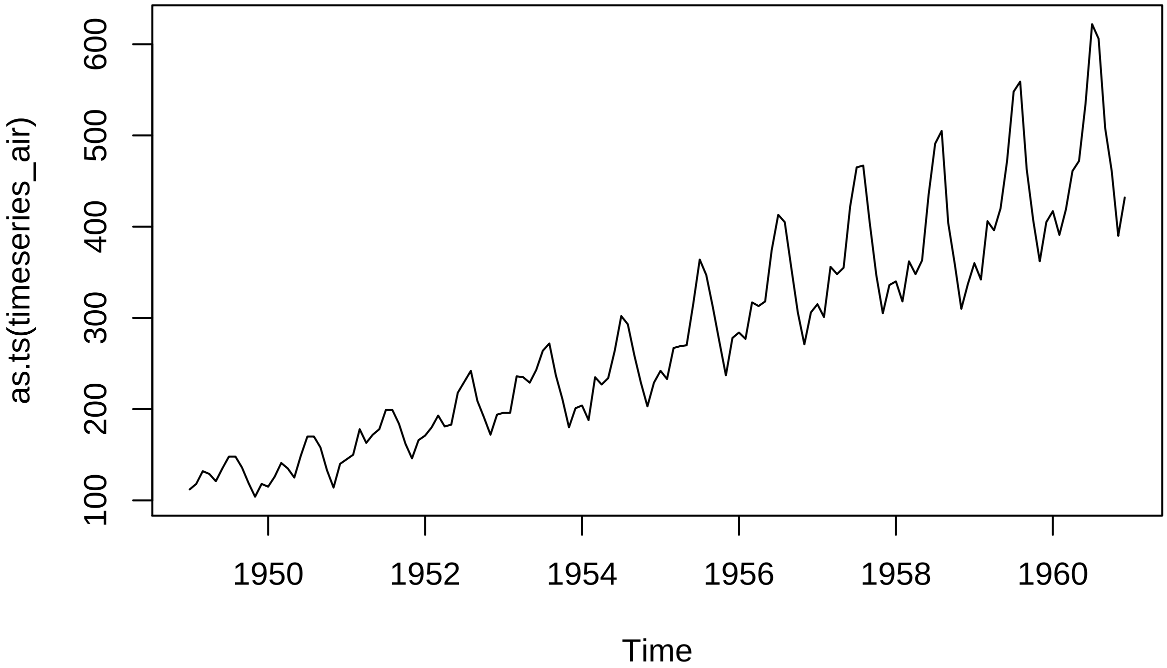
Dive in Deeper to TimeSeries

For this portion of our lab we will be using data from the AirPassengers Dataset

```
data(AirPassengers)
```

Step 6: Make an inital TimeSeries Visual of the data

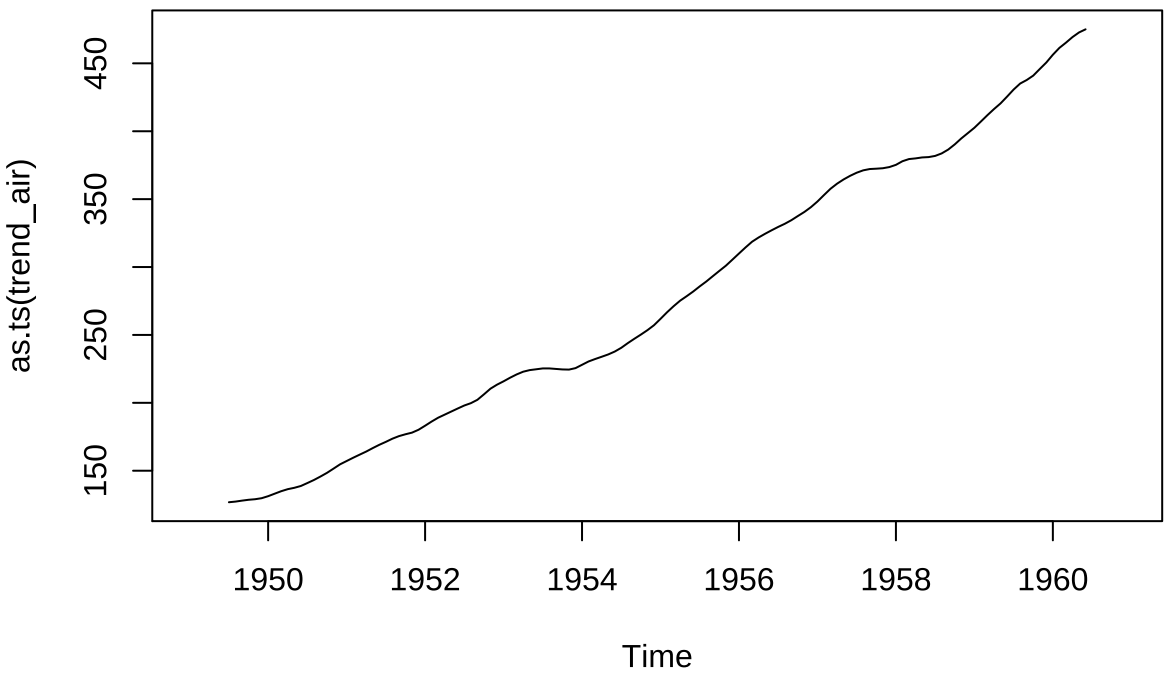
```
timeseries_air = AirPassengers
plot(as.ts(timeseries_air))
```



##

Step 7: Compute the Moving Average of this data using forecast package and vizualize this

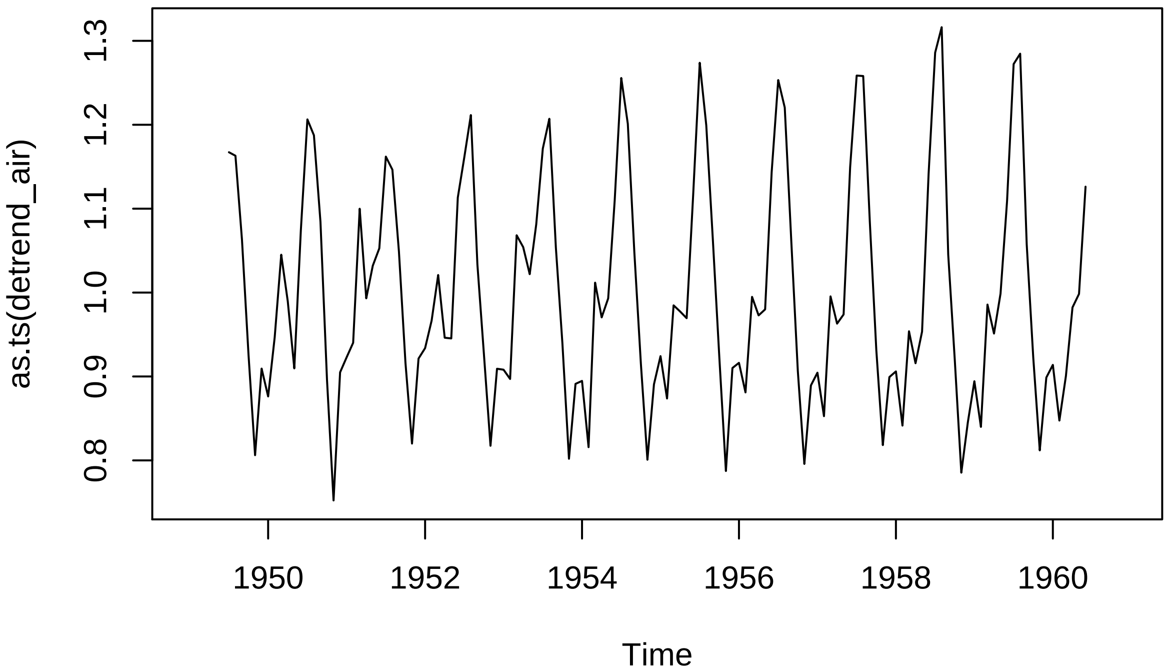
```
trend_air = ma(timeseries_air, order = 12, centre = T)
#lines(trend_air)
plot(as.ts(trend_air))
```



##

Step 8: Remove the Trend from the data and Visualize this

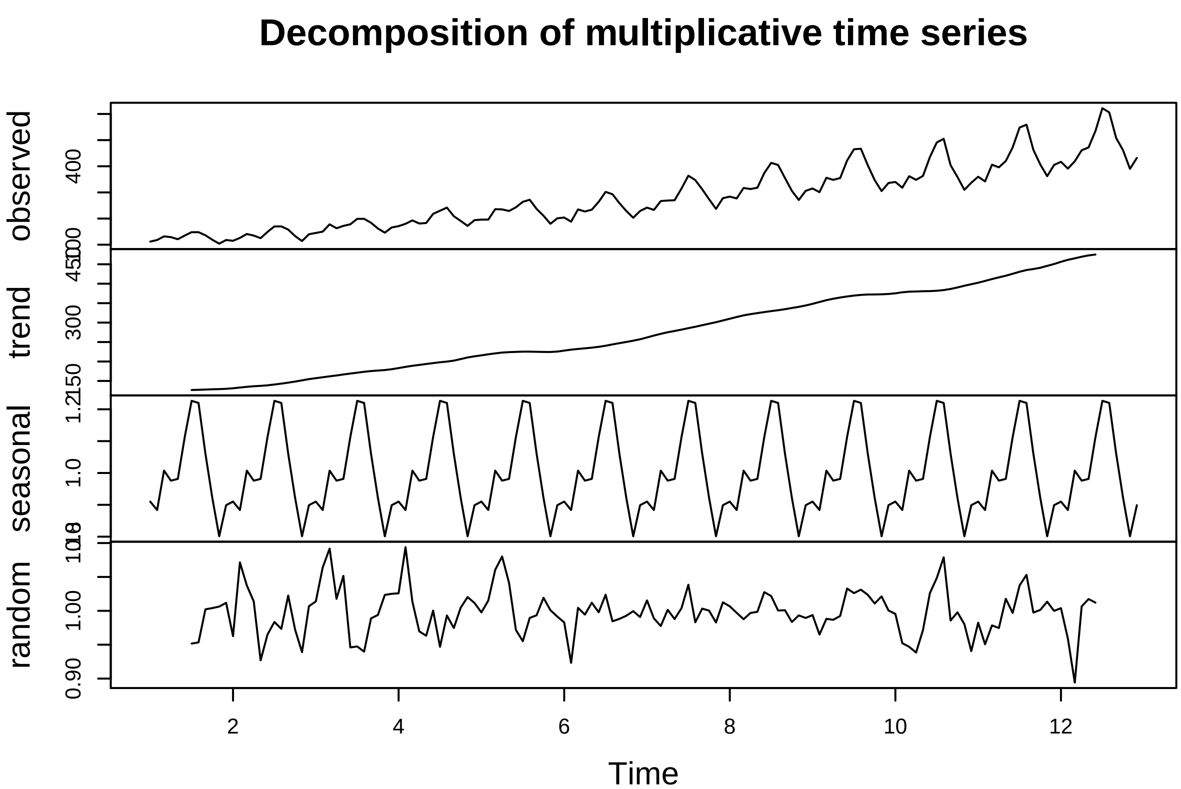
```
detrend_air = timeseries_air / trend_air
plot(as.ts(detrend_air))
```



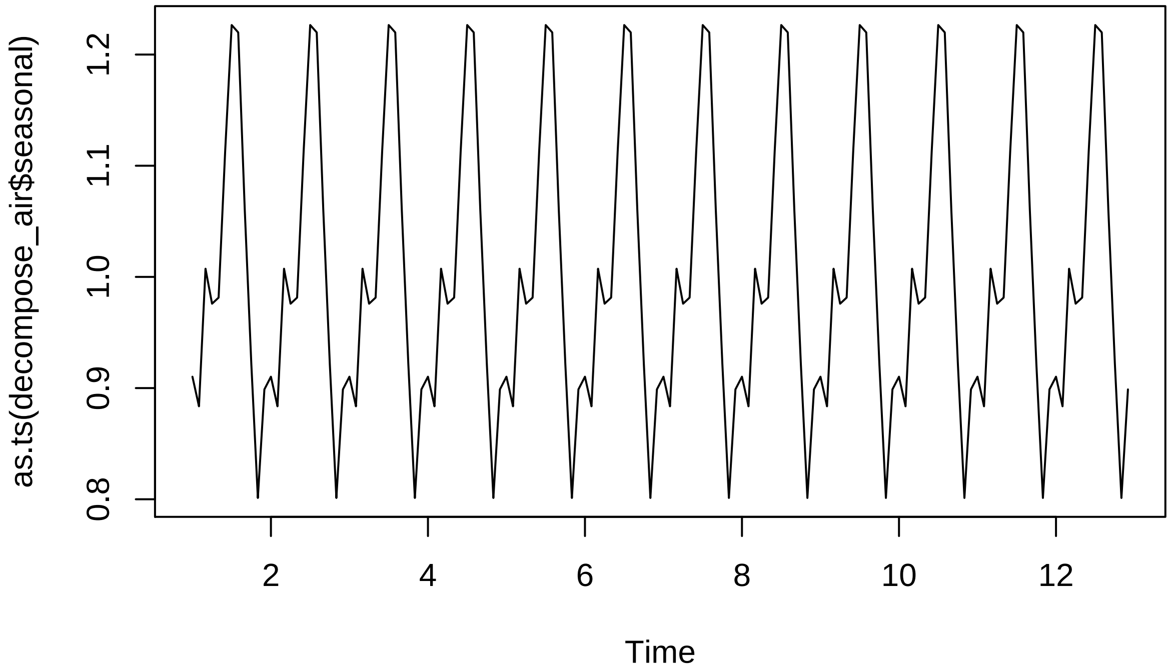
##

Step 9: Create a decomposition of the data by month
– Hint (Frequency = 12)

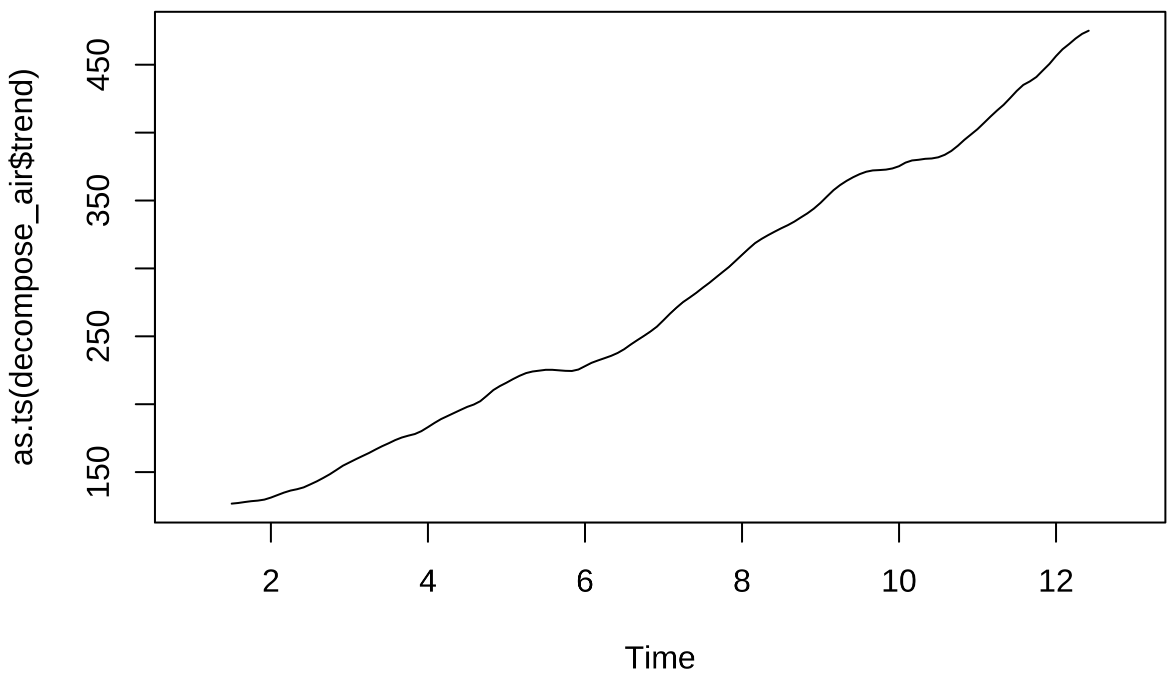
```
ts_air = ts(timeseries_air, frequency = 12)
decompose_air = decompose(ts_air, "multiplicative")

plot(decompose_air)
```



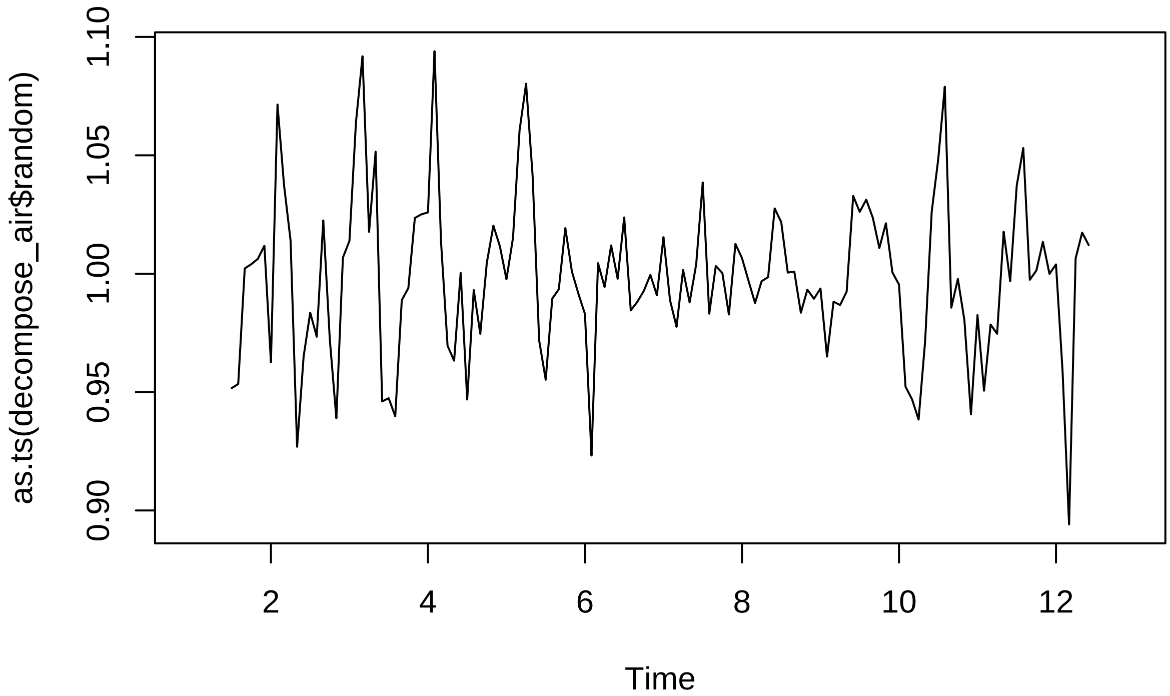
```r
### Advanced answers
plot(as.ts(decompose_air$seasonal))
```




```r
plot(as.ts(decompose_air$trend))
```




```r
plot(as.ts(decompose_air$random))
```



as.ts(decompose_air$random)
0.90
0.95
1.00
1.05
1.10
2
4
6
8
10
12
Time