TimeSeries Skeleton

Richard Decal

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)

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

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.

```
dep_delay_noNA <- flights$DEP_DELAY[!is.na(flights$DEP_DELAY)]
(mean(dep_delay_noNA))
```

```
## [1] 11.10297
```

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.

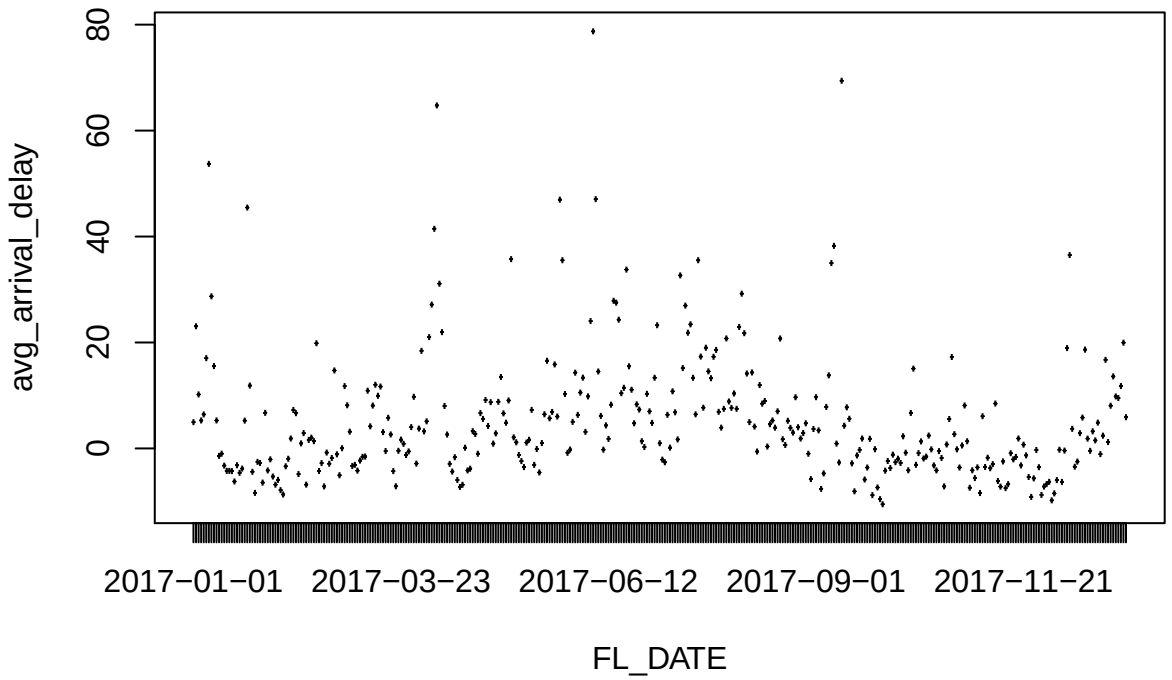
```
flights_ts <- as.ts(flights)
```

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

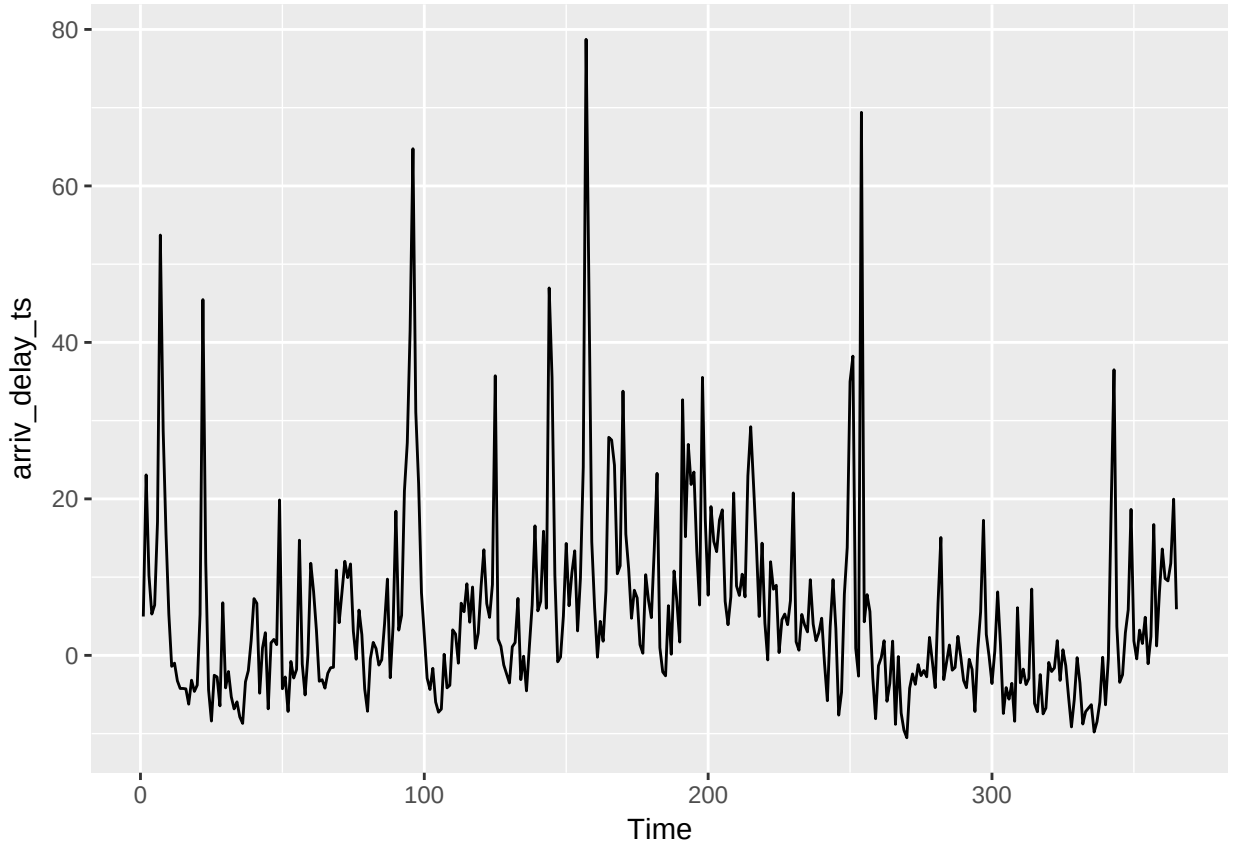
Create a basic visualization of the time series.

```
# before plot ts, we need to aggregate. sort by flight date, groupby, aggregate with mean
arriv_delay <- flights %>%
    arrange(FL_DATE) %>%
    group_by(FL_DATE) %>%
    summarise(mean(ARR_DELAY, na.rm = TRUE))
colnames(arriv_delay) <- c('FL_DATE', 'avg_arrival_delay')

plot(arriv_delay)
```




```
# get rid of plot of time against itself by selecting col
# this is why it's important to arrange() first, since we a re throwing out that
# data and relying on the index.
arriv_delay_ts <- as.ts(arriv_delay$avg_arrival_delay)
autoplot(arriv_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

```
#TODO
```

Questions
1. How does the neighborhood size, i.e. the number of points in each localized subset, affect the amount of smoothing?

Larger window size means more smoothing.

2. What happened to endpoints of the smoothed data?

Points at the extreme ends of the timeseries will be undefined.

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.

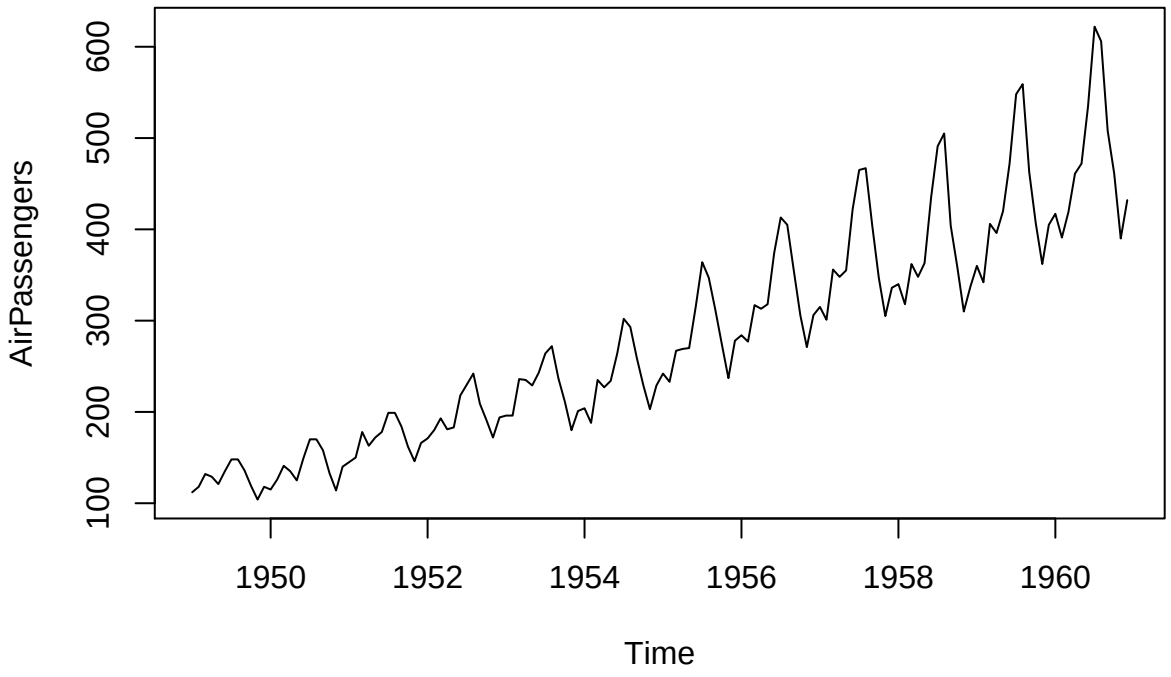
Dive in Deeper to TimeSeries

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

```
plot(AirPassengers)
```



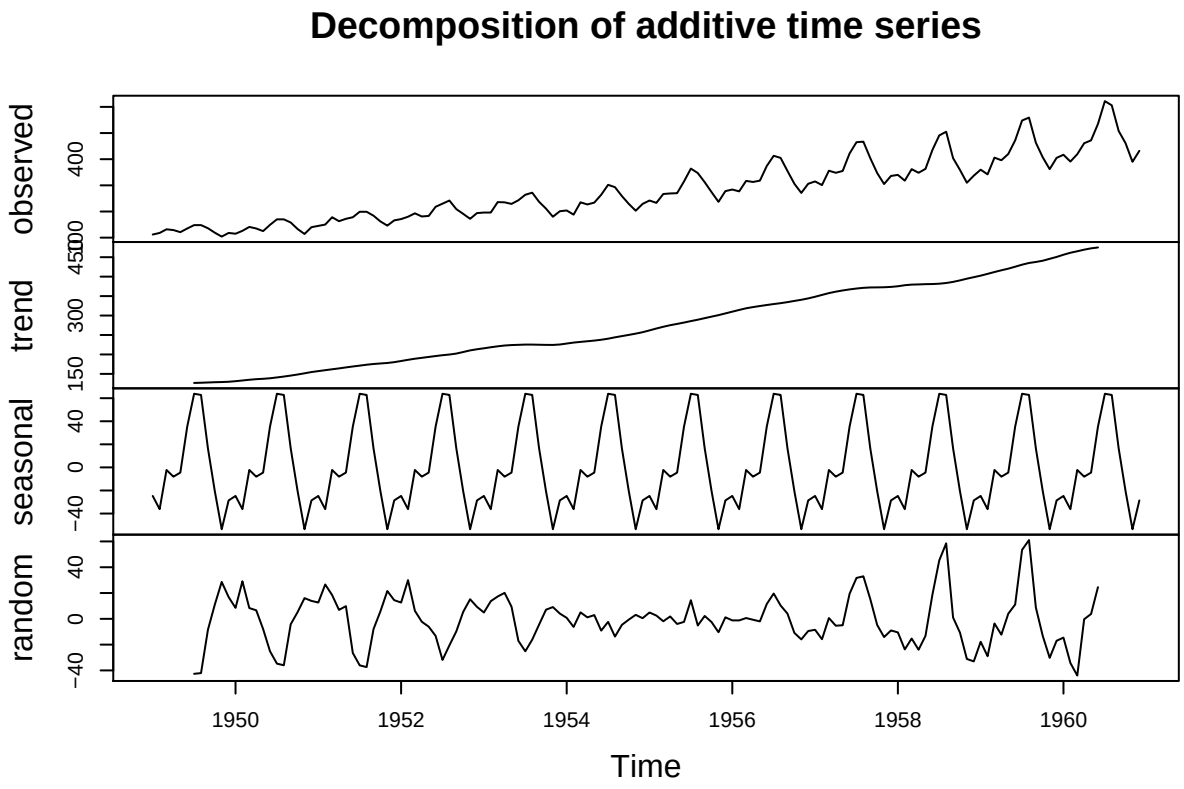
Step 7: Compute the Moving Average of this data using forecast package and vizualize this

```
air_pass_ma <- forecast::ma(AirPassengers, order=1)
```

Step 9: Create a decomposition of the data by month

– Hint (Frequency = 12)

```
x <- as.ts(AirPassengers, frequency=12)
decomposition <- decompose(x)
plot(decomposition)
```



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

```
trend_subtracted <- x - decomposition$trend
plot(trend_subtracted)
```