Course Outline: Time Series Data in R

Harrison Brown

2022-05-04

Contents

Welcome

Welcome to the course outline for *Time Series Data in R*! This course offers methods and workflows for analyzing and interpreting time series data, an overview of when, why, and how to use time series data, and various utilities and packages in R that are beneficial to analysts.

By the end of this course, students will have the skills to:

- Interpret and understand time series plots
- Import ts data to create and manipulate `ts` objects from the `stats` package
- Understand why time series data is fundamentally different than non-ts data.
- Analyze time series data with plots
- ?Intro to Wavelet analysis?

Chapter 1

Introduction to time series data

1.1 What is a time series

- Sampled at equi-spaced points in time

1.2 Stationary vs Non-Stationary series

Non-stationary time series are defined by:

- Time-dependent Mean
- Time-dependent Variance
- Time-dependent Autocorrelation/Covariance

1.3 Dickey-Fuller Test of Stationarity

Chapter 2

Creating and Manipulating Time Series

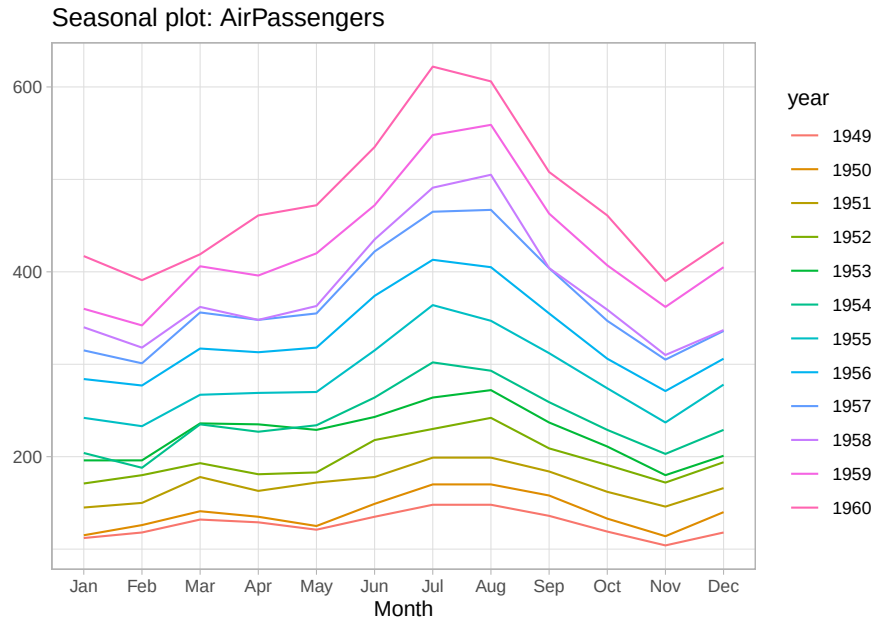
2.1 `ts` Class

2.2 Creating a `ts.plot()`

2.2.1 Interpreting Plots

```
ggseasonplot(x = AirPassengers)
```

2.2.2 Seasonality Plot



2.2.3 Polar Seasonality Plot

2.3 Trends and Seasons

2.3.1 Decomposition

2.3.2 De-trending Data

Chapter 3

Rolling and Expanding Windows

3.1 Rolling Window

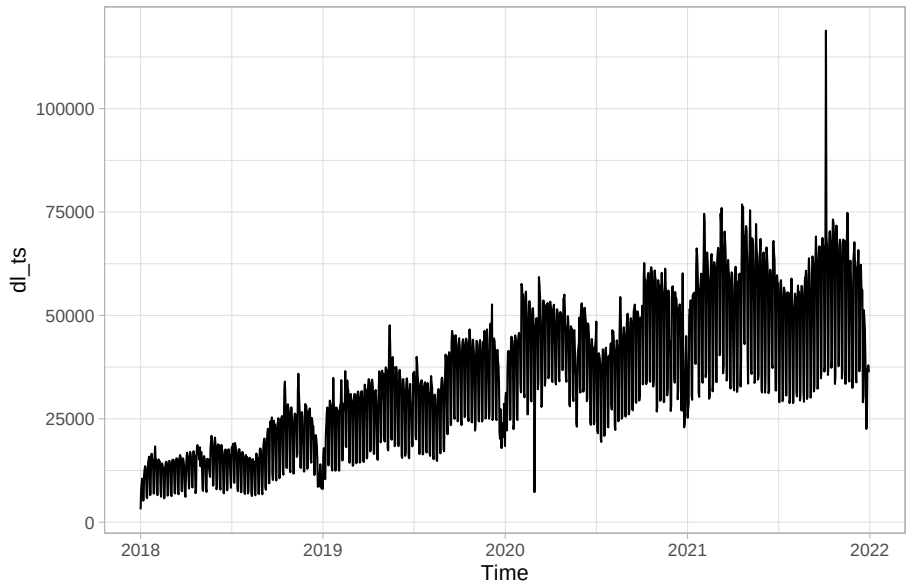
- Moving lower and upper bound

3.1.1 Data

```
library(tsbox)

dl_dplyr <- cran_data %>%
  filter(date >= as.Date("2018-01-01")) %>%
  select(-package)

# tsbox::ts_ts() parses the Date column in a much easier way, making the
# conversion process easy to interpret
dl_ts <- dl_dplyr %>%
  tsbox::ts_ts()
```

```
## [time]: 'date' [value]: 'count'
```

```
autoplot(dl_ts)
```



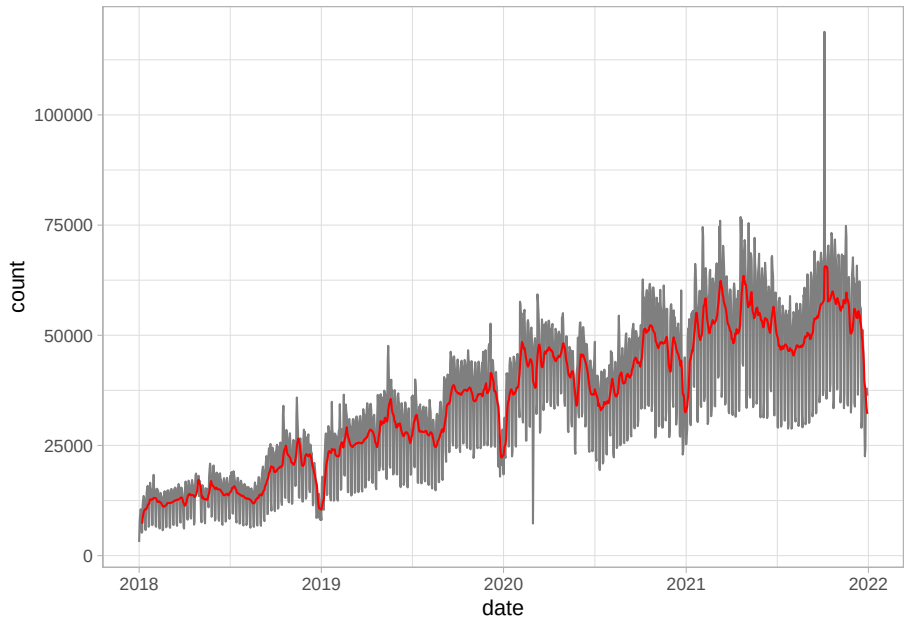
3.1.2 Calculating a Rolling Window

```
dl_dplyr_rolling <- dl_dplyr %>%
  tq_mutate(
    select = count,
    mutate_fun = rollapplyr,
    FUN = mean,
    width = 7,
    col_rename = "weekly_avg"
  ) %>%
  tq_mutate(
    select = count,
    mutate_fun = rollapplyr,
    FUN = mean,
    width = 30,
    col_rename = "monthly_avg"
  )

weekly_ts <- dl_dplyr_rolling %>%
  select(date, weekly_avg) %>%
  ts_ts()
```

```
## [time]: 'date' [value]: 'weekly_avg'
```

```
ggplot() +
  geom_line(data = dl_dplyr, mapping = aes(x = date, y = count), color = "grey50") +
  geom_line(data = dl_dplyr_rolling, mapping = aes(x = date, y = weekly_avg), color = "red")
```

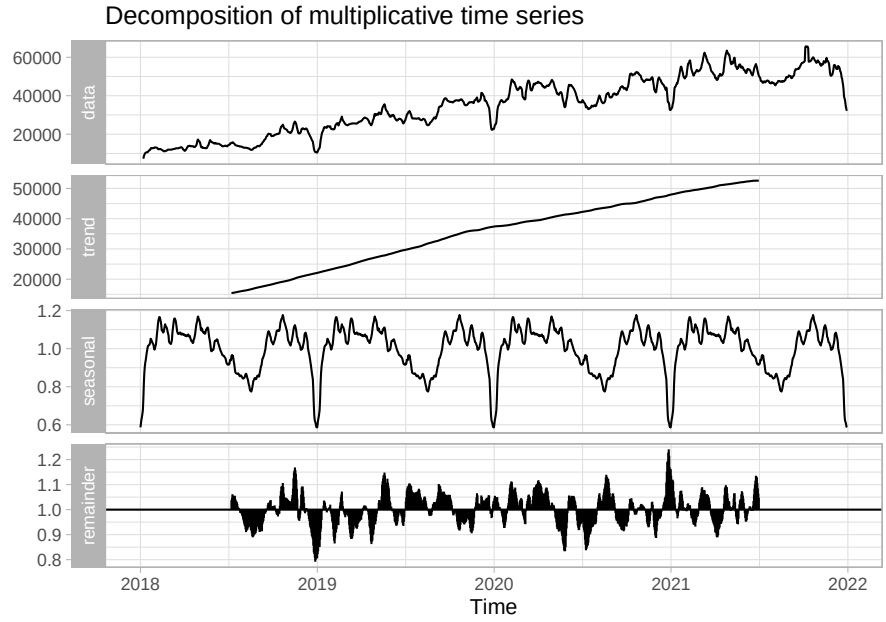
```
## Warning: Removed 6 row(s) containing missing values (geom_path).
```




```
dl_rolling_ts <- dl_dplyr_rolling %>%
  select(
    date, weekly_avg
  ) %>%
  ts_ts()
```

```
## [time]: 'date' [value]: 'weekly_avg'
```

```
dl_rolling_ts %>%
  decompose(type = "multiplicative") %>%
  autoplot()
```


Decomposition of multiplicative time series
data
trend
seasonal
remainder
60000
40000
20000
50000
40000
30000
20000
1.2
1.0
0.8
0.6
1.2
1.1
1.0
0.9
0.8
2018
2019
2020
2021
2022
Time

Chapter 4

Expanding Window

- Fixed lower bound
- Moving upper bound

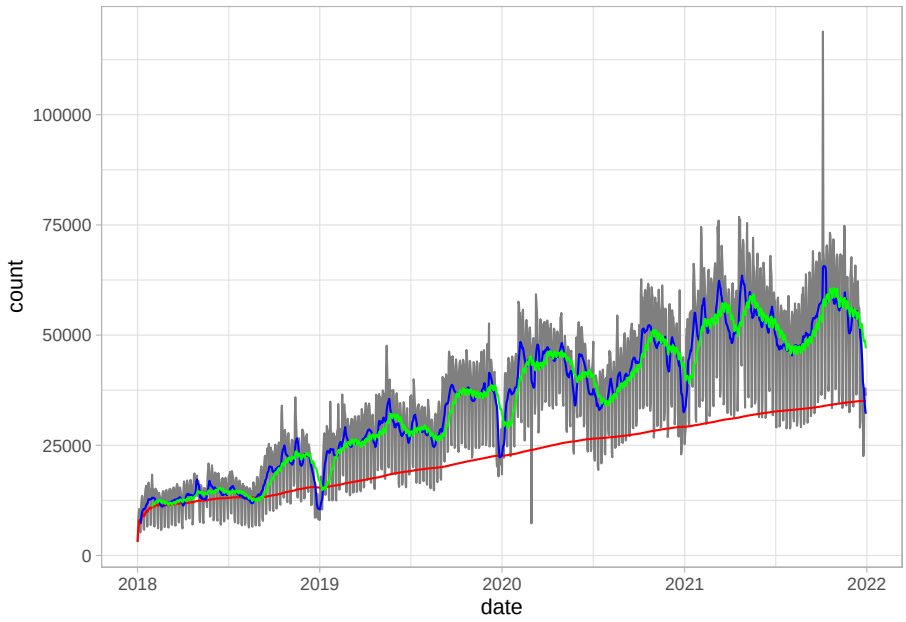
```
dl_expand <- dl_dplyr %>%
  mutate(
    mean_expand = cummean(count),
    cum_sum = cumsum(count)
  )

expand_ts <- dl_expand %>%
  select(date, mean_expand) %>%
  ts_ts()
```

```
## [time]: 'date' [value]: 'mean_expand'
```

```
ggplot() +
  geom_line(data = dl_dplyr, aes(x = date, y = count), color = "gray50") +
  geom_line(data = dl_expand, aes(x = date, y = mean_expand), color = "red") +
  geom_line(data = dl_dplyr_rolling, aes(x = date, y = weekly_avg), color = "blue") +
  geom_line(data = dl_dplyr_rolling, aes(x = date, y = monthly_avg), color = "green") +
  theme_light()
```

```
## Warning: Removed 6 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 29 row(s) containing missing values (geom_path).
```


100000
75000
50000
25000
0
count
2018
2019
2020
2021
2022
date

Chapter 5

Introduction to Forecasting in R

5.1 Methods for Forecasting

5.1.1 Exponential Smoothing