Bali_sport

2025-05-29

Introduction

Bali is the top tourism destination in Indonesia. Bali's Ngurah Rai international airports accommodate 38.7% of Indonesia's international visitor arrivals in 2019, and has risen to 45.4% in 2024{1}. The foreign visitors with continuing tourism development has introduce new lifestyle changes in Bali regarding diets, wellness, culture, and fitness.

Data Preparation

Package and file load

```
## Data processing
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

File load

For this analysis, I'm gonna use the google trends' relative search volume (RSV) data on key search terms. RSV represent the popularity level of a search terms in a given period. RSV ranges from 0 to 100, with 100 representing the peak popularity (based on quantity of search terms entry), and 0 represent the lowest popularity. For this analysis, I will investigate 6 search terms limited to the Bali region: - Gym - Boxing - Running - Surfing - Yoga - Meditation I will also use monthly data from 2010 to 2019. I choose this period to avoid external (omitted) effects of google trends' data collection adjustment in January 2011 and the Covid-19 pandemic in 2020.

Intial Checking

```
glimpse(df)
```

```
## Rows: 108
## Columns: 8
## $ Month              <chr> "2011-01", "2011-02", "2011-03", "2011-04", "2011-0~
## $ boxing...Bali.     <int> 0, 73, 0, 0, 0, 60, 0, 36, 52, 39, 56, 69, 37, 0, 4~
## $ gym...Bali.        <int> 21, 0, 21, 20, 21, 19, 23, 25, 21, 28, 31, 32, 36, ~
## $ Yoga...Bali.       <int> 38, 44, 47, 52, 42, 34, 30, 34, 25, 36, 36, 33, 45,~
## $ Surfing...Bali.    <int> 66, 71, 56, 60, 49, 77, 57, 62, 64, 45, 45, 45, 38,~
## $ Running...Bali.    <int> 24, 20, 18, 0, 17, 23, 25, 22, 22, 24, 23, 24, 36, ~
## $ Meditation...Bali. <int> 0, 0, 0, 0, 0, 0, 0, 87, 0, 0, 61, 0, 64, 96, 77, 9~
## $ Diving...Bali.     <int> 52, 66, 69, 84, 75, 70, 66, 63, 66, 63, 54, 53, 59,~
```

The 'Week' column types should be date instead of characters.

```
head(df, 5)
```

```
##     Month boxing...Bali. gym...Bali. Yoga...Bali. Surfing...Bali.
## 1 2011-01              0          21           38              66
## 2 2011-02             73           0           44              71
## 3 2011-03              0          21           47              56
## 4 2011-04              0          20           52              60
## 5 2011-05              0          21           42              49
##   Running...Bali. Meditation...Bali. Diving...Bali.
## 1              24                  0             52
## 2              20                  0             66
## 3              18                  0             69
## 4               0                  0             84
## 5              17                  0             75
```

The column name is too long, I need to rename it for simplicity.

```
df <- df %>%
  rename(
    gym = gym...Bali.,
    yoga = Yoga...Bali.,
    boxing = boxing...Bali.,
    surfing = Surfing...Bali.,
    running = Running...Bali.,
    meditation = Meditation...Bali.,
    diving = Diving...Bali.
  )
```

```
glimpse(df)
```

```
## Rows: 108
## Columns: 8
## $ Month      <chr> "2011-01", "2011-02", "2011-03", "2011-04", "2011-05", "201~
```

```
## $ boxing     <int> 0, 73, 0, 0, 0, 60, 0, 36, 52, 39, 56, 69, 37, 0, 46, 46, 8~
## $ gym        <int> 21, 0, 21, 20, 21, 19, 23, 25, 21, 28, 31, 32, 36, 40, 35, ~
## $ yoga       <int> 38, 44, 47, 52, 42, 34, 30, 34, 25, 36, 36, 33, 45, 49, 49,~
## $ surfing    <int> 66, 71, 56, 60, 49, 77, 57, 62, 64, 45, 45, 45, 38, 49, 57,~
## $ running    <int> 24, 20, 18, 0, 17, 23, 25, 22, 22, 24, 23, 24, 36, 29, 24, ~
## $ meditation <int> 0, 0, 0, 0, 0, 0, 0, 87, 0, 0, 61, 0, 64, 96, 77, 94, 66, 5~
## $ diving     <int> 52, 66, 69, 84, 75, 70, 66, 63, 66, 63, 54, 53, 59, 67, 67,~
```

Data Cleaning

Format Correction

Transform the 'Month' column into date (m/d/y).

```
df$Month <- as.Date(paste(df$Month, "-01", sep=""))
```

```
glimpse(df)
```

```
## Rows: 108
## Columns: 8
## $ Month      <date> 2011-01-01, 2011-02-01, 2011-03-01, 2011-04-01, 2011-05-01~
## $ boxing     <int> 0, 73, 0, 0, 0, 60, 0, 36, 52, 39, 56, 69, 37, 0, 46, 46, 8~
## $ gym        <int> 21, 0, 21, 20, 21, 19, 23, 25, 21, 28, 31, 32, 36, 40, 35, ~
## $ yoga       <int> 38, 44, 47, 52, 42, 34, 30, 34, 25, 36, 36, 33, 45, 49, 49,~
## $ surfing    <int> 66, 71, 56, 60, 49, 77, 57, 62, 64, 45, 45, 45, 38, 49, 57,~
## $ running    <int> 24, 20, 18, 0, 17, 23, 25, 22, 22, 24, 23, 24, 36, 29, 24, ~
## $ meditation <int> 0, 0, 0, 0, 0, 0, 0, 87, 0, 0, 61, 0, 64, 96, 77, 94, 66, 5~
## $ diving     <int> 52, 66, 69, 84, 75, 70, 66, 63, 66, 63, 54, 53, 59, 67, 67,~
```

Duplicate Check

Check duplicate rows

```
sum(duplicated(df))
```

```
## [1] 0
```

Check for repeated time observations ('Month').

```
## Check repeated coords
month_counts <- df %>%
  group_by(Month) %>%
  summarise(n=n())

duplicate_months <- month_counts %>%
  filter(n > 1)

print(duplicate_months)
```

```
## # A tibble: 0 x 2
## # i 2 variables: Month <date>, n <int>
```

Findings: - There's no duplicated rows nor time entry.

NA's Check

```
sum(is.na(df))
```

```
## [1] 0
```

There's no NA entry.

Empty string value

Prepare to check all columns

```
all_cols <- colnames(df)
```

```
print("Columns with empty string value")
```

```
## [1] "Columns with empty string value"
```

```
for (col in all_cols) {
  n_empty <- sum(df[[col]]=="", na.rm = TRUE)
  if (n_empty > 0) {
  print(paste0("Number and percentages of empty string value in ", col))
  print(n_empty)
  print((n_empty/(nrow(df)))*100)
  }
}
```

There's no columns with missing value.

EDA

```
## For plotting
library(ggplot2)
```

Data Preparation

```
num_cols <- names(df)[sapply(df, is.numeric)]
print("Column who are numeric types:")
```

```
## [1] "Column who are numeric types:"
```

```
print(num_cols)
```

```
## [1] "boxing"     "gym"        "yoga"       "surfing"    "running"
## [6] "meditation" "diving"
```

Exploratory analysis

Date analysis

```
summary(df$Month)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2011-01-01" "2013-03-24" "2015-06-16" "2015-06-16" "2017-09-08" "2019-12-01"
```

Checks numerical quartile analysis

```
summary(df[num_cols])
```

Checking

```
##      boxing            gym              yoga           surfing
##  Min.   :  0.00   Min.   :  0.00   Min.   : 20.00   Min.   : 22.00
##  1st Qu.: 38.00   1st Qu.: 34.00   1st Qu.: 52.00   1st Qu.: 40.00
##  Median : 46.00   Median : 45.00   Median : 65.00   Median : 48.00
##  Mean   : 47.13   Mean   : 46.52   Mean   : 63.79   Mean   : 50.83
##  3rd Qu.: 55.00   3rd Qu.: 62.00   3rd Qu.: 76.25   3rd Qu.: 59.00
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00
##     running         meditation         diving
##  Min.   :  0.00   Min.   :  0.00   Min.   : 16.00
##  1st Qu.: 37.75   1st Qu.: 48.75   1st Qu.: 47.75
##  Median : 46.00   Median : 56.00   Median : 55.00
##  Mean   : 49.25   Mean   : 52.72   Mean   : 57.41
##  3rd Qu.: 61.00   3rd Qu.: 64.00   3rd Qu.: 67.25
##  Max.   :100.00   Max.   :100.00   Max.   :100.00
```

For the following analysis, we will needs to transform the variable to ensure normal distribution. As transformation can't handle 0, then I will transform data points equal to 0 as 0.1 and 0.5 to all data points less than 1 but not 0.

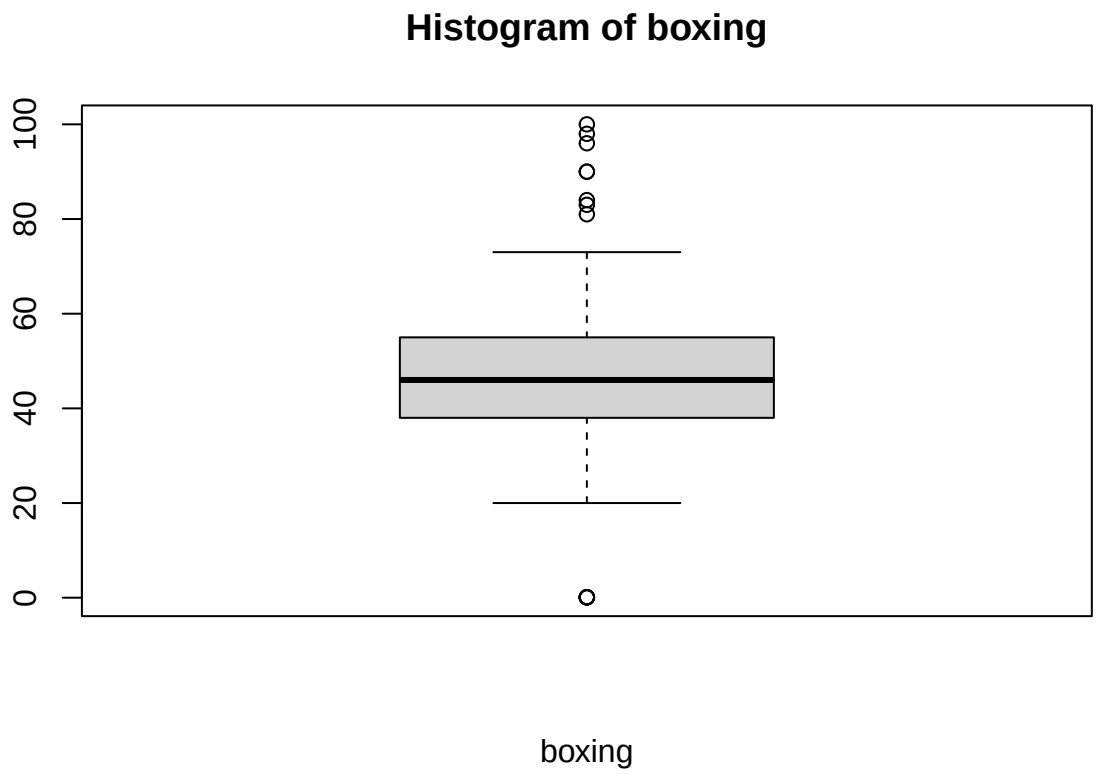
```
df[num_cols][df[num_cols]< 1 & df[num_cols] > 0] = 0.5

df[num_cols][df[num_cols]== 0] = 0.1

## Convert to numeric
df[num_cols] <- sapply(df[num_cols], as.numeric)
```

```r
summary(df[num_cols])
```

Data transformation

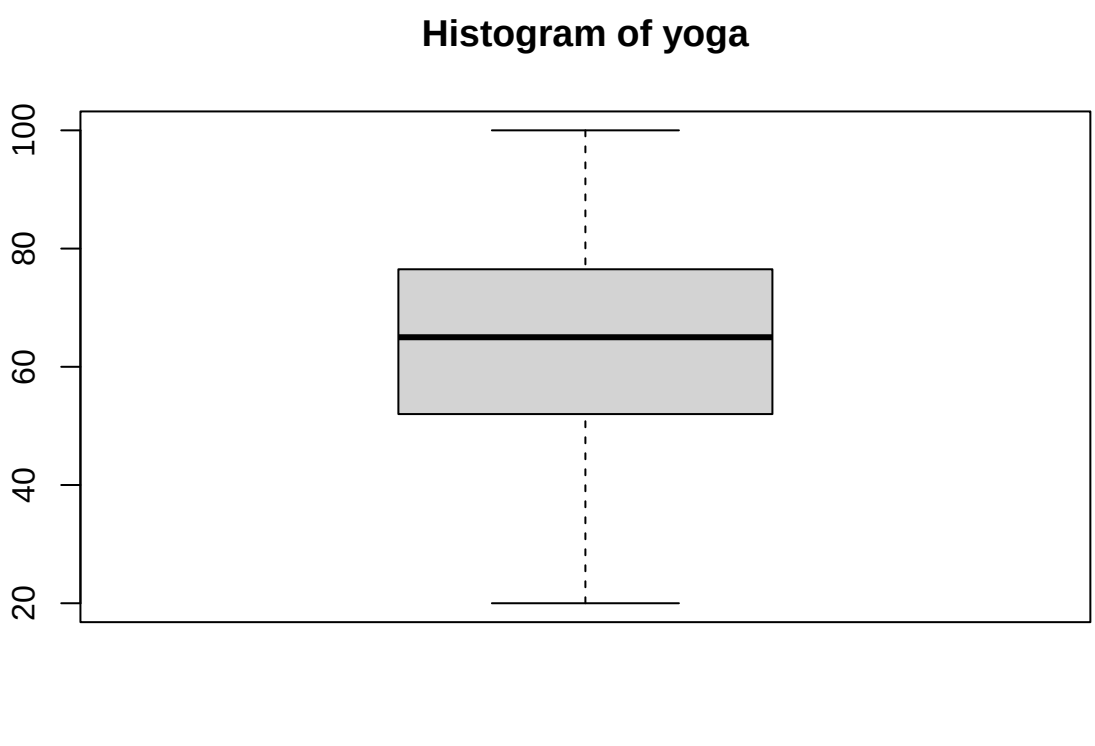
```
##      boxing            gym              yoga           surfing
##  Min.   :  0.10   Min.   :  0.10   Min.   : 20.00   Min.   : 22.00
##  1st Qu.: 38.00   1st Qu.: 34.00   1st Qu.: 52.00   1st Qu.: 40.00
##  Median : 46.00   Median : 45.00   Median : 65.00   Median : 48.00
##  Mean   : 47.14   Mean   : 46.52   Mean   : 63.79   Mean   : 50.83
##  3rd Qu.: 55.00   3rd Qu.: 62.00   3rd Qu.: 76.25   3rd Qu.: 59.00
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00
##     running         meditation         diving
##  Min.   :  0.10   Min.   :  0.10   Min.   : 16.00
##  1st Qu.: 37.75   1st Qu.: 48.75   1st Qu.: 47.75
##  Median : 46.00   Median : 56.00   Median : 55.00
##  Mean   : 49.25   Mean   : 52.73   Mean   : 57.41
##  3rd Qu.: 61.00   3rd Qu.: 64.00   3rd Qu.: 67.25
##  Max.   :100.00   Max.   :100.00   Max.   :100.00
```

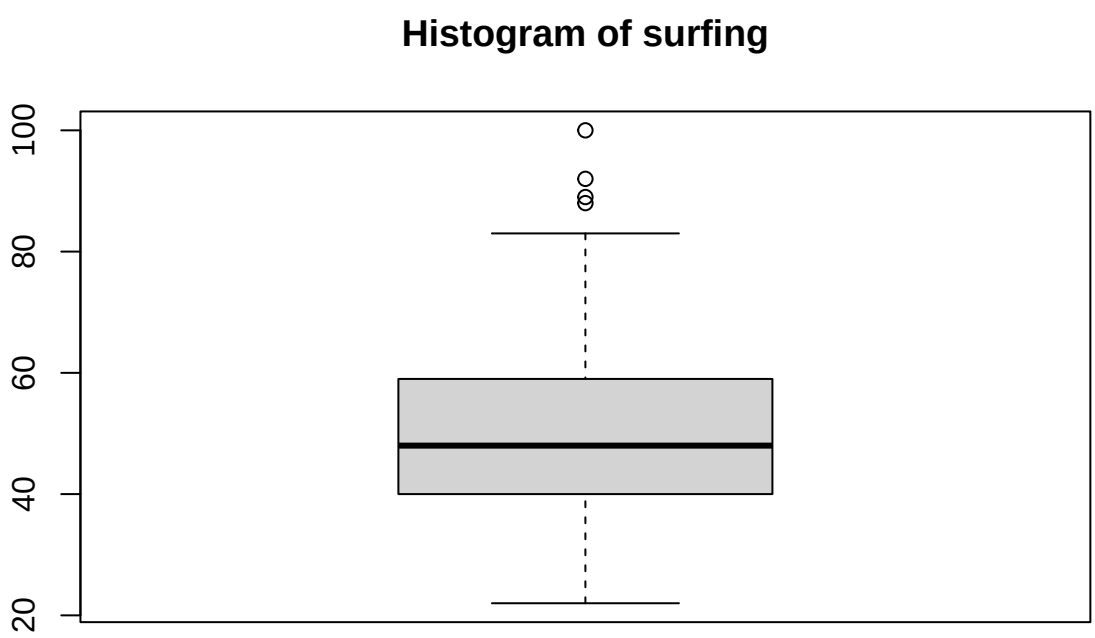
```r
for (col in num_cols){
  boxplot(df[[col]], main = paste("Histogram of", col), xlab = col)
}
```

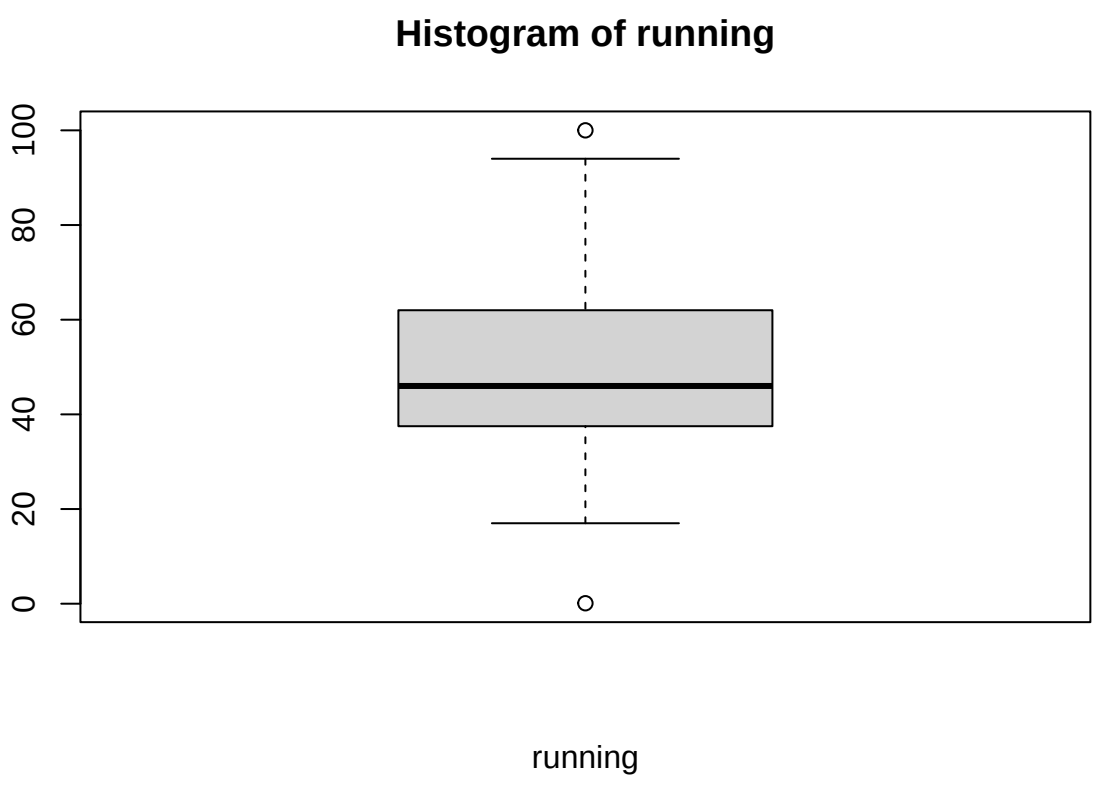

Histogram of boxing
0
20
40
60
80
100
boxing

Boxplot

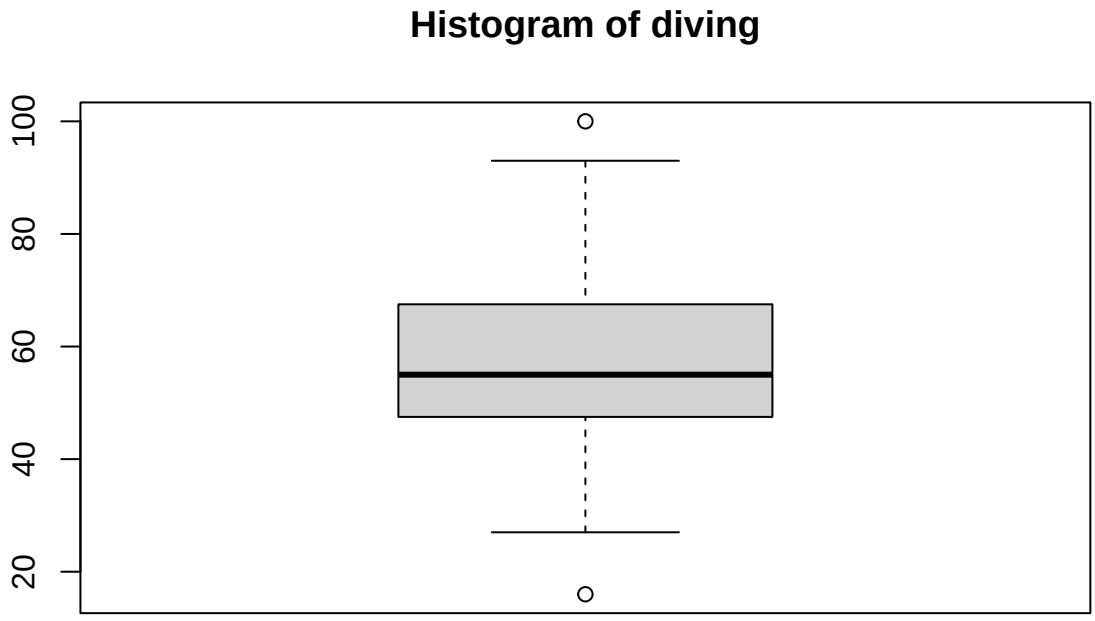

Histogram of gym
0
20
40
60
80
100
gym


Histogram of yoga
100
80
60
40
20
yoga


Histogram of surfing
100
80
60
40
20
surfing


Histogram of running
0
20
40
60
80
100
running


Histogram of meditation
0
20
40
60
80
100
meditation




```
for (col in num_cols){

  var <- df[[col]]
  #Calculate IQE, Upper, and Lower Bound
  iqr <- IQR(var)
  q1 <- quantile(var, 0.25)
  q3 <- quantile(var, 0.75)

  # Calculate Upper and lower bound
  lower_bound <- q1 - (1.5*iqr)
  upper_bound <- q3 + (1.5*iqr)

  # Count character that is outisde the bounds
  outliers <- which(var < lower_bound | var > upper_bound)

  # Count percentages
  print(col)
  res = (length(outliers) / length (df[[col]]))*100
  print(res)
}
```

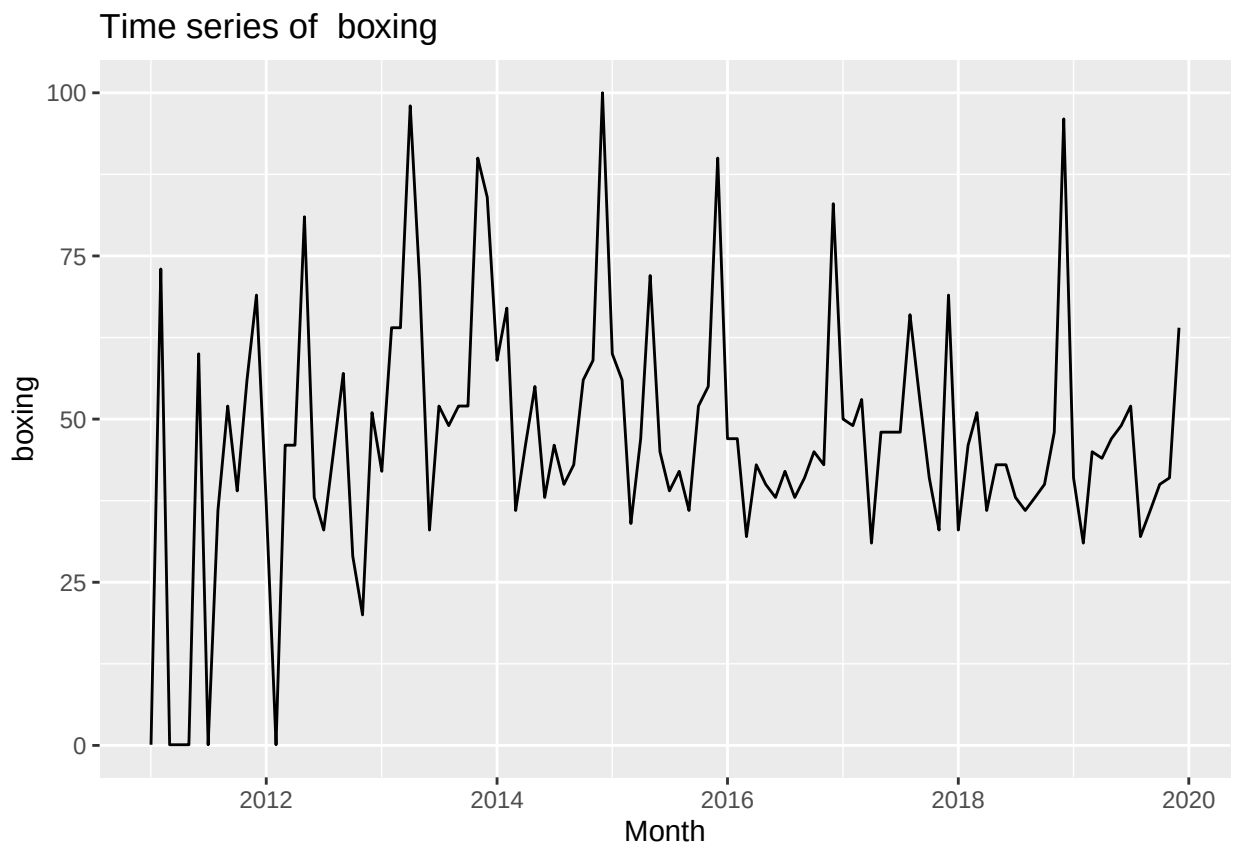
```
## [1] "boxing"
## [1] 12.96296
## [1] "gym"
## [1] 0
## [1] "yoga"
```

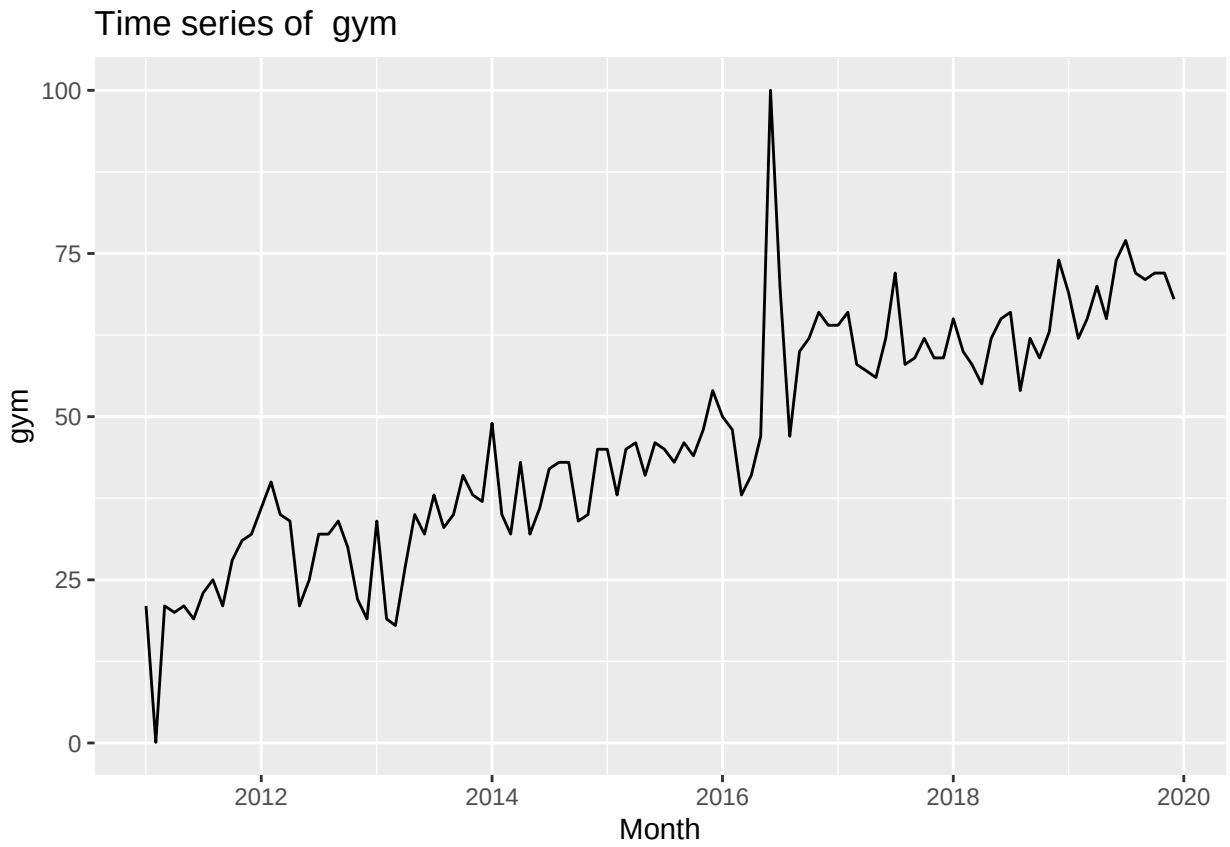
```
## [1] 0
## [1] "surfing"
## [1] 3.703704
## [1] "running"
## [1] 1.851852
## [1] "meditation"
## [1] 16.66667
## [1] "diving"
## [1] 1.851852
```

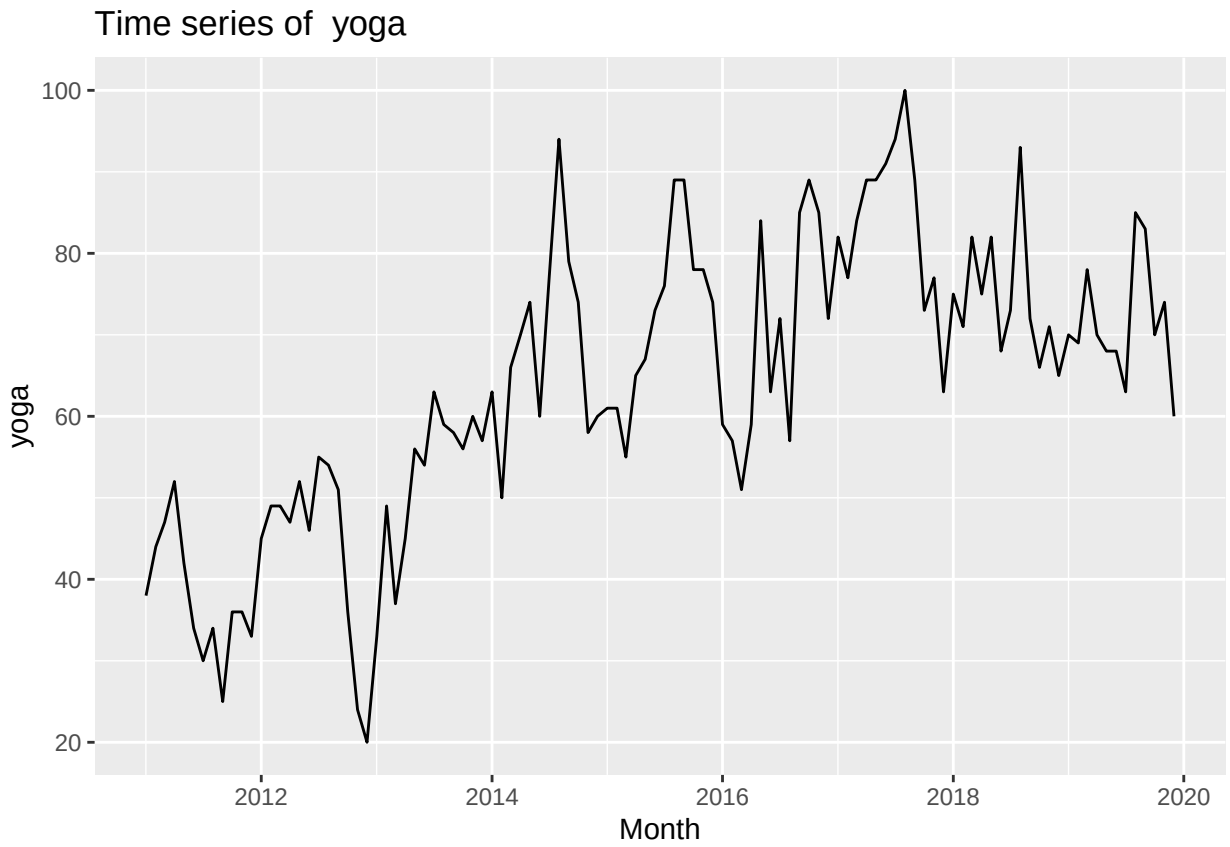
Findings: - 'boxing' variable have more than 5% of outliers.

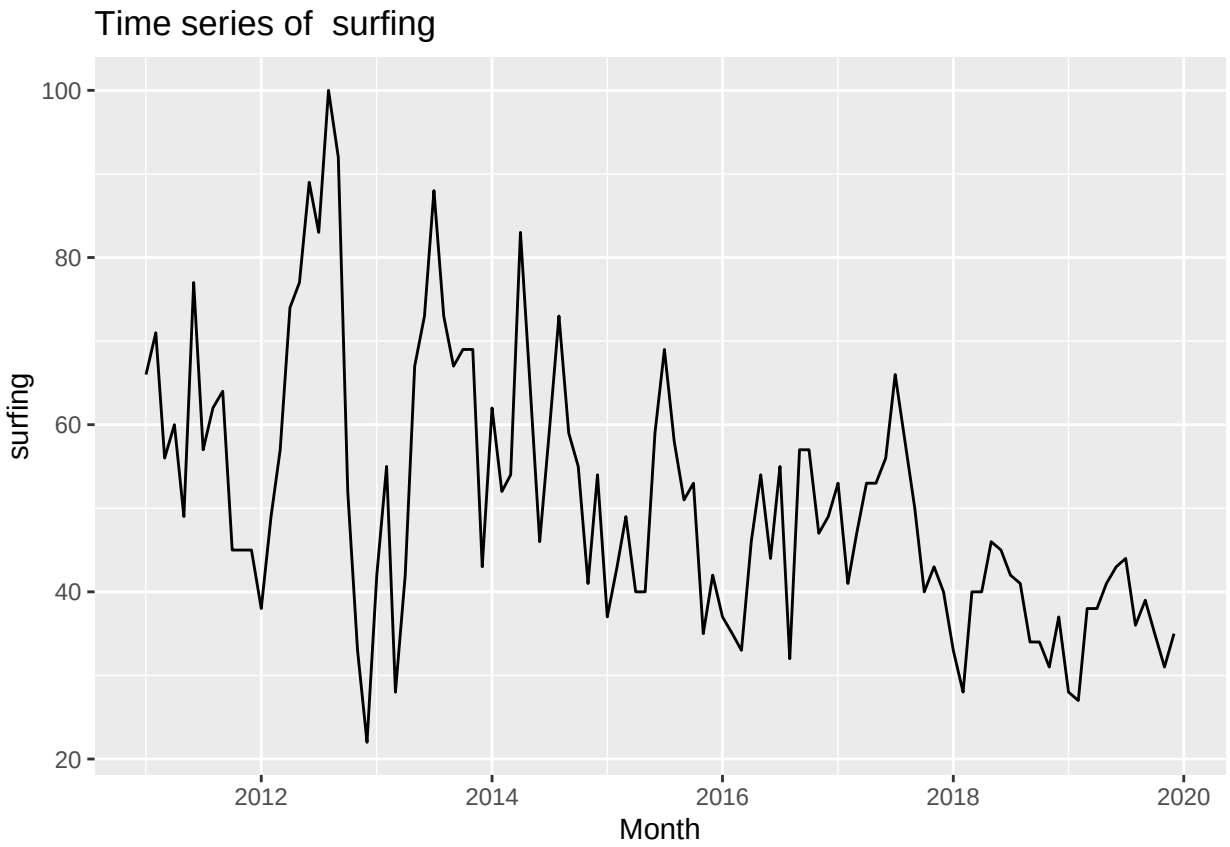
Time series line

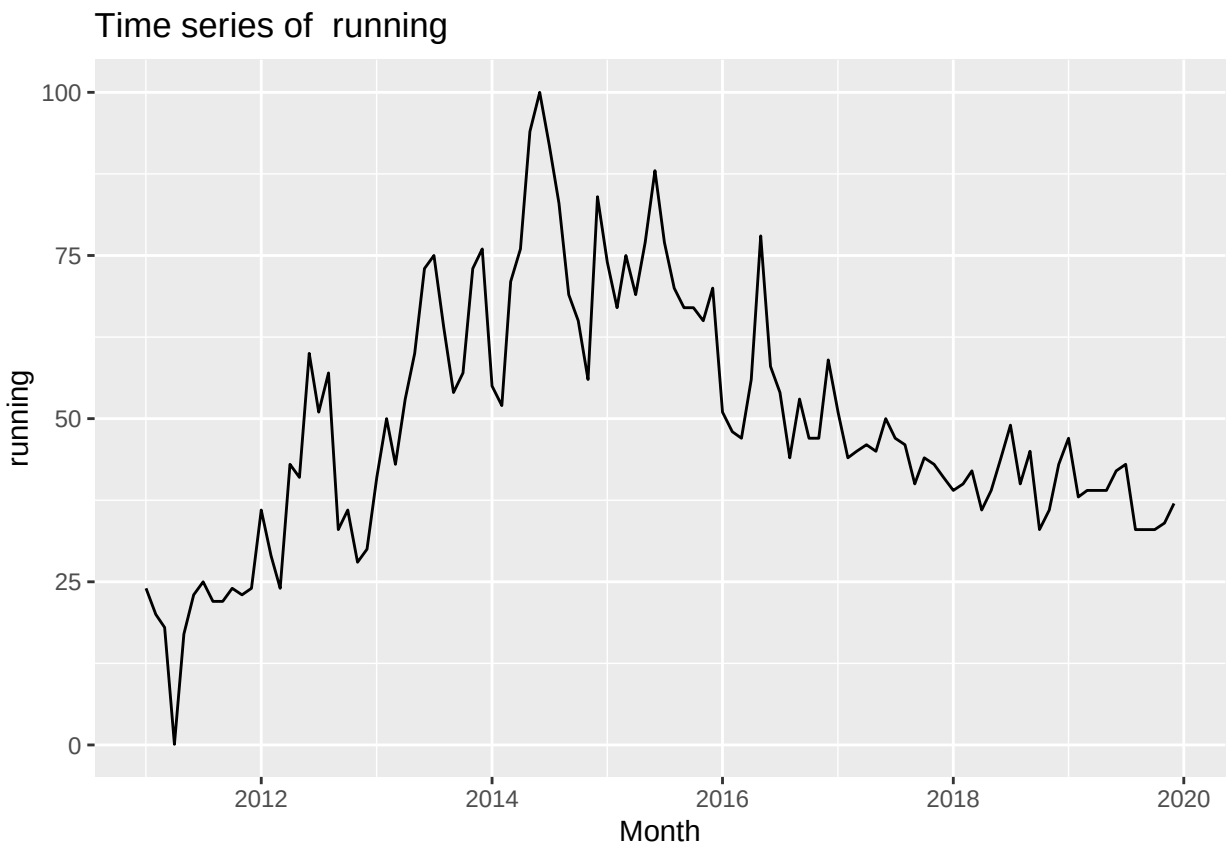
```
for (col in num_cols){
  print(ggplot(data = df) +
  geom_line(mapping = aes(x = Month, y = .data[[col]])) +
  labs(title = paste("Time series of ", col))
  )
}
```

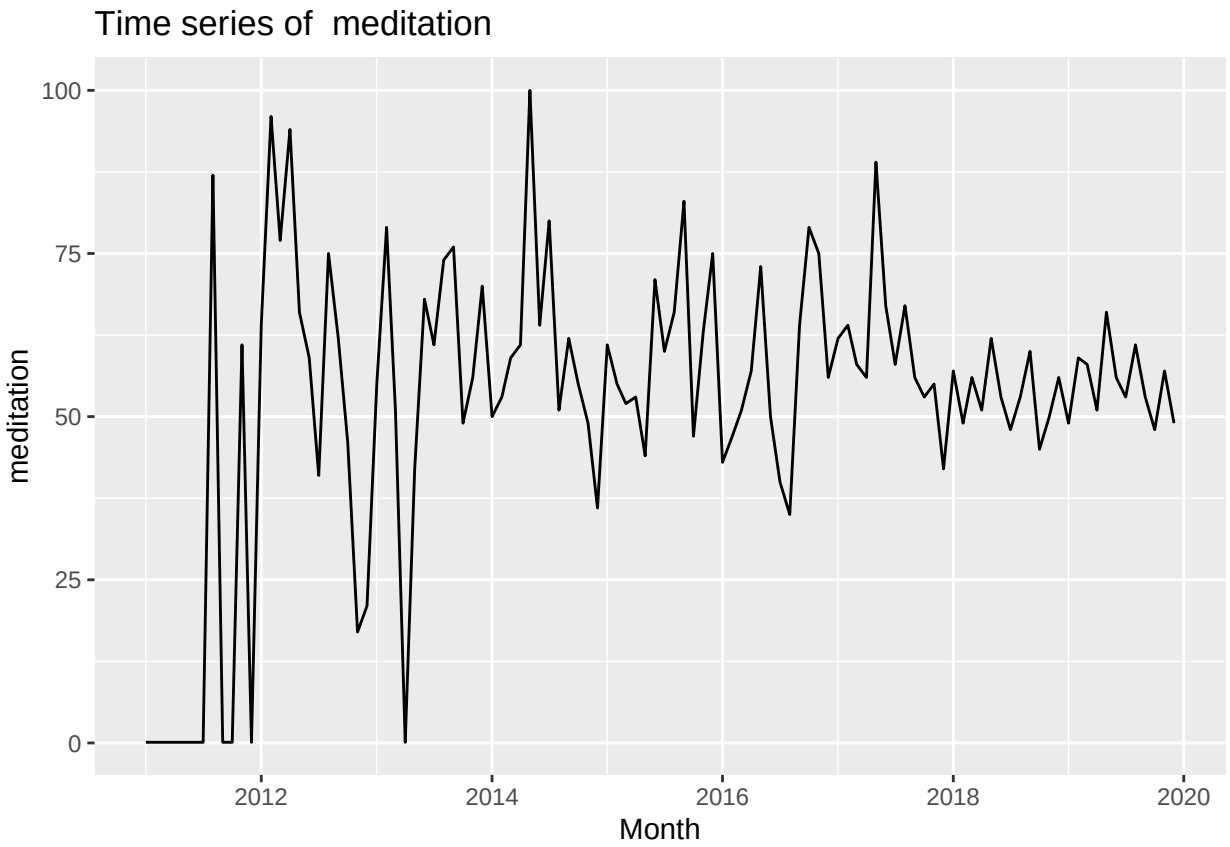


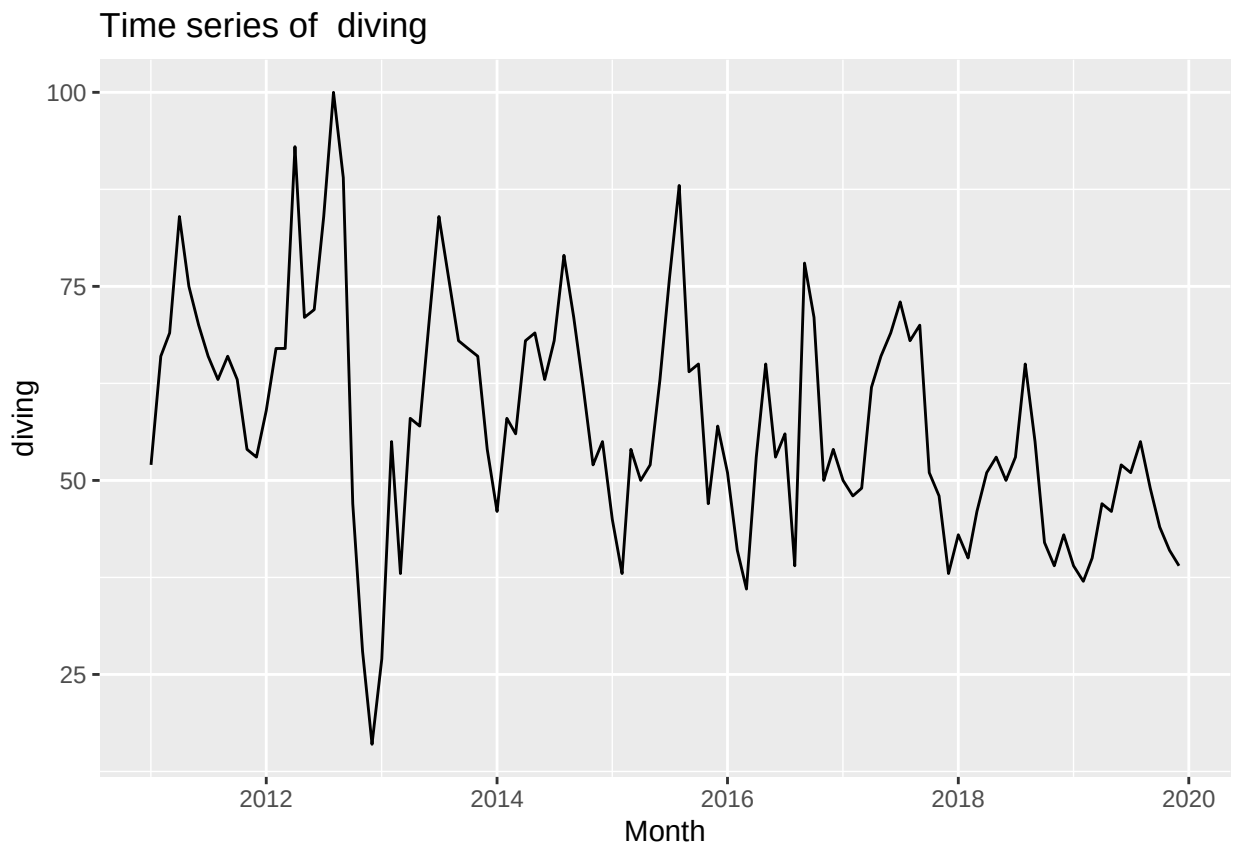

Time series of gym
100
75
50
25
0
gym
2012
2014
2016
2018
2020
Month


Time series of yoga
100
80
60
40
20
yoga
2012
2014
2016
2018
2020
Month


Time series of surfing
surfing
100
80
60
40
20
2012
2014
2016
2018
2020
Month


Time series of running
100
75
50
25
0
running
2012
2014
2016
2018
2020
Month


Time series of meditation
meditation
100
75
50
25
0
2012
2014
2016
2018
2020
Month



Trend Analysis This analysis is focused on the trend aspect of secular and seasonal trend. - Secular trend: long term movement that is constant despite the seasonal trend fluctuations. - seasonal trend : Recurring predictable fluctuations pattern based on a particular 'season' in a year. In this analysis, the seasonal trend will be focused on a month based seasonal trend. I choose to analyze monthly seasonal trend because I hypothesized RSV result in Bali is not affected by climate season (3 or 6 months interval) due to the tropical climate but rather by holiday seasons on a particular month (such as new year and spring break). ## Trend validation

Secular trend validation

To test for secular trend, I performed the seasonal Mann-Kendall test on all of the search. Significant result signify that the search terms has a secular trend for the given period. I will also further run a univariate linear regression on the terms with significant secular terms to find the monthly rate of secular trend changes according to the time slope coefficient.

```
library(Kendall)
```

```
## Warning: package 'Kendall' was built under R version 4.3.3
```

There is a positive secular trend as the p-value is less than 0.5 and tau >0.

```
## Established dataframe
columns <- c("Terms","SMK_P_value","SMK_tau", "slope")
df_sk <- data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(df_sk) <- columns
```

```
## Run S-MK test
for (col in num_cols){
  temp_ts <- ts(df[[col]], frequency = 12)
  smk_results <- SeasonalMannKendall(temp_ts)
  sl_model <- lm(df[[col]] ~ df$Month)
  if (smk_results$sl < 0.05){
    sv <- as.numeric(sl_model$coefficients[2])
  } else{
    sv <- "-"}
  ## Input to temp dataframe
  temp <- data.frame(Terms = c(col),
                     SMK_P_value = c(smk_results$sl),
                     SMK_tau = c(smk_results$tau),
                     slope = c(sv))
  ## Input to new dataframe
  df_sk <- rbind(df_sk, temp)
}
df_sk
```

```
##        Terms  SMK_P_value     SMK_tau                 slope
## 1     boxing 2.147805e-01 -0.09614827                     -
## 2        gym 0.000000e+00  0.84188804   0.0166234275371659
## 3       yoga 1.663247e-11  0.52045001   0.0132409707514005
## 4    surfing 3.212985e-12 -0.53786581 -0.00894025582993535
## 5    running 4.692946e-01 -0.05581578                     -
## 6 meditation 6.506315e-01  0.03500903                     -
## 7     diving 2.901512e-09 -0.45761866 -0.00651600531064838
```

TBATS

To analyze monthly seasonal trend I run an exponential smoothing state-space model with Box-Cox transformation, autoregressive-moving average errors, trend, and seasonal components (TBATS): - First I check whether if the search terms have a seasonality effect by comparing the AIC value of the model with monthly seasonal incorporated and non seasonal effect. I determined that a search terms have a seasonal trend if the seasonal trend model have lower AIC value. - Second, I use the TBATS to decompose the seasonal effect of each months to find months with the highest and lowest seasonal effect value. - Third, I calculate the yearly amplitude by calculating the difference of the peak and lowest popular month to find the fitness popularity volatility.

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
for (col in num_cols){
  col_ts <- ts(df[[col]], frequency = 12)
  col_vc <- df[[col]]
```

```
  #tbats
  tbats_model_ts <- tbats(col_ts, seasonal.periods=12)
  tbats_model_1 <- tbats(col_vc)

  print(paste0("Seasonality of ", col))
  print(tbats_model_1$seasonal.periods)
  print("Seasonality model result")
  print(tbats_model_ts$seasonal.periods)
  print("AIC value with seasonality (time series)")
  print(tbats_model_ts$AIC)
  print("AIC value without seasonality (numerical vector)")
  print(tbats_model_1$AIC)
}
```

```
## [1] "Seasonality of boxing"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 1127.186
## [1] "AIC value without seasonality (numerical vector)"
## [1] 1138.515
## [1] "Seasonality of gym"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 954.9824
## [1] "AIC value without seasonality (numerical vector)"
## [1] 958.3901
## [1] "Seasonality of yoga"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 985.2029
## [1] "AIC value without seasonality (numerical vector)"
## [1] 995.9681
## [1] "Seasonality of surfing"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 1015.911
## [1] "AIC value without seasonality (numerical vector)"
## [1] 1033.242
## [1] "Seasonality of running"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 964.8112
## [1] "AIC value without seasonality (numerical vector)"
```

```
## [1] 984.228
## [1] "Seasonality of meditation"
## NULL
## [1] "Seasonality model result"
## NULL
## [1] "AIC value with seasonality (time series)"
## [1] 1134.604
## [1] "AIC value without seasonality (numerical vector)"
## [1] 1134.604
## [1] "Seasonality of diving"
## NULL
## [1] "Seasonality model result"
## [1] 12
## [1] "AIC value with seasonality (time series)"
## [1] 1011.198
## [1] "AIC value without seasonality (numerical vector)"
## [1] 1031.212
```

Only meditation that is found not having seasonal variations.

```
remove_char <- "meditation"

num_cols <- num_cols[!(num_cols %in% remove_char)]
```

LOESS

```
library(stats)
```

```
## Established the min max dataframe
columns <- c("Terms","Max_value","Min_value","Max_month", "Min_month")
df_rsv <- data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(df_rsv) <- columns

## Run and extract LOESS value
for (col in num_cols){
  col_ts <- ts(df[[col]], frequency = 12)
  decomp <- stl(col_ts, s.window="periodic", t.window=13)
  seasonal <- decomp$time.series[, "seasonal"]
  ## Convert to time series and keep the first row
  df_s <- as.data.frame(seasonal)
  df_s <- df_s[1:12, , drop=FALSE]
  df_s$month <- month.abb[as.numeric(rownames(df_s))]
  ## Add LOESS value to min-max
  temp <- data.frame(Terms = c(col),
                     Max_value = c(max(df_s$x)),
                     Min_value = c(min(df_s$x)),
                     Max_month = c(df_s$month[df_s$x == max(df_s$x)]),
                     Min_month = c(df_s$month[df_s$x == min(df_s$x)]))
  df_rsv <- rbind(df_rsv, temp)
}
df_rsv$Amplitude <- df_rsv$Max_value - df_rsv$Min_value
df_rsv
```

```
##     Terms Max_value  Min_value Max_month Min_month Amplitude
## 1  boxing 30.846618  -8.247642       Dec       Jul 39.094259
## 2     gym  5.136306  -4.532308       Jul       Mar  9.668615
## 3    yoga 10.062676  -8.875181       Aug       Dec 18.937856
## 4 surfing 11.767629  -8.641345       Jul       Dec 20.408974
## 5 running 10.442638  -5.675944       Jun       Feb 16.118583
## 6  diving 12.997124 -12.952824       Aug       Jan 25.949949
```

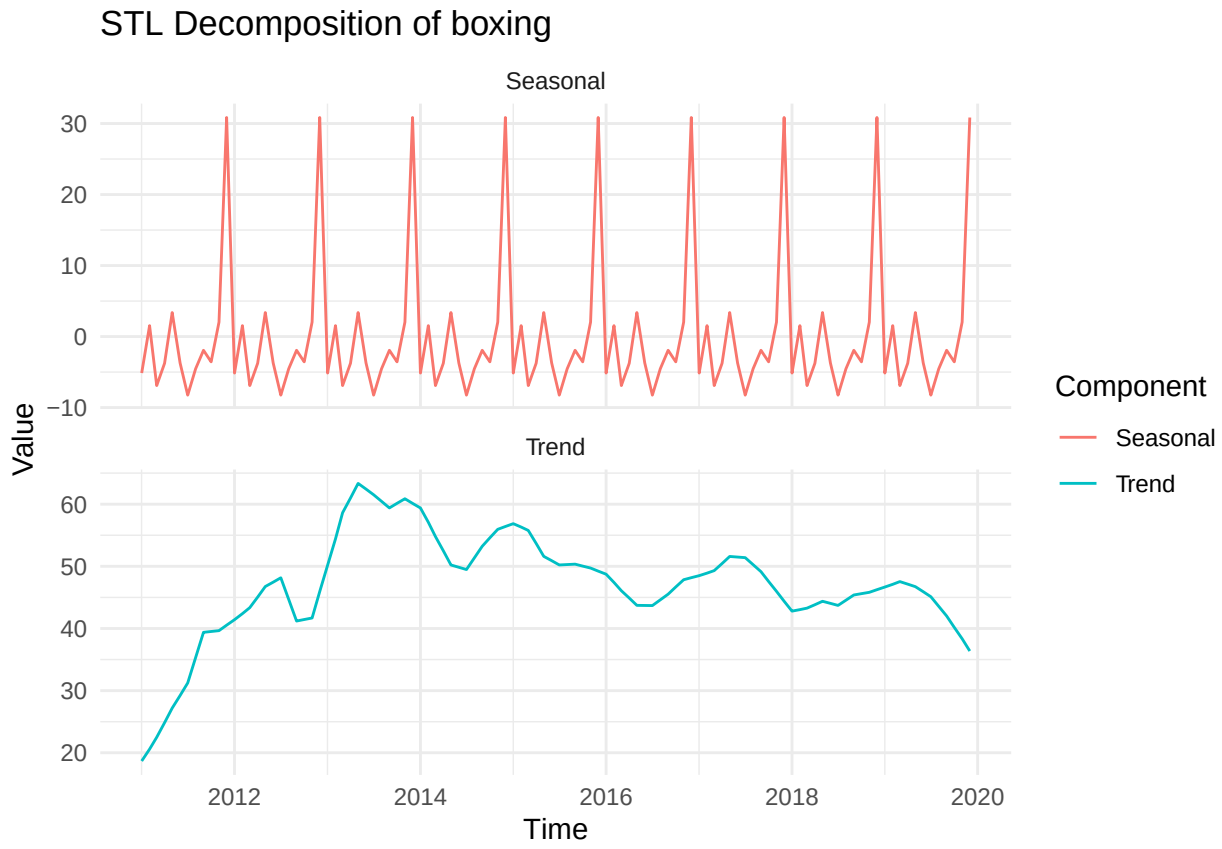
```
df_merge <- merge(df_sk,df_rsv,by="Terms", all.x=TRUE)
df_merge[is.na(df_merge)] <- "-"
df_merge
```

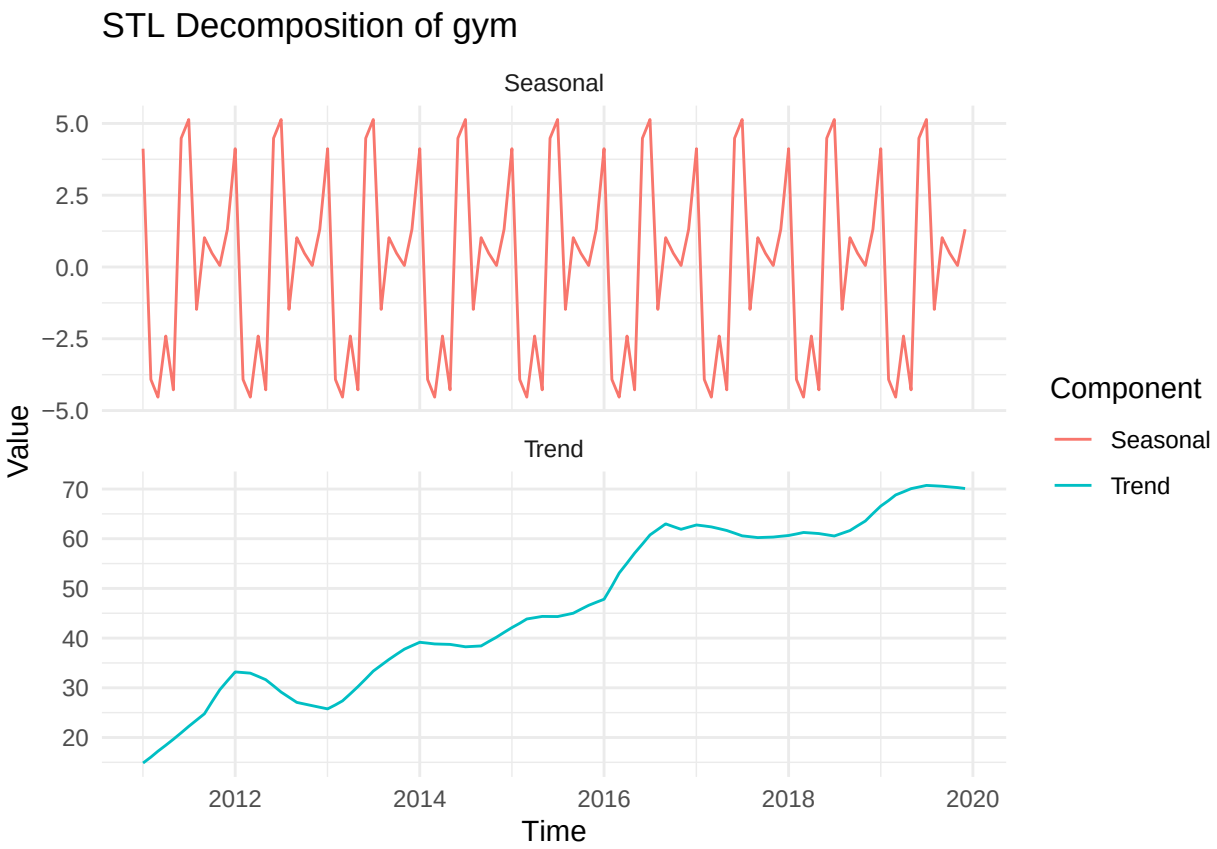
```
##        Terms  SMK_P_value     SMK_tau                slope        Max_value
## 1     boxing 2.147805e-01 -0.09614827                    - 30.8466177010131
## 2     diving 2.901512e-09 -0.45761866 -0.00651600531064838 12.9971243901335
## 3        gym 0.000000e+00  0.84188804   0.0166234275371659 5.13630639674017
## 4 meditation 6.506315e-01  0.03500903                    -                -
## 5    running 4.692946e-01 -0.05581578                    - 10.4426384393948
## 6    surfing 3.212985e-12 -0.53786581 -0.00894025582993535  11.767629441726
## 7       yoga 1.663247e-11  0.52045001   0.0132409707514005 10.0626759536567
##           Min_value Max_month Min_month        Amplitude
## 1 -8.24764168889022       Dec       Jul 39.0942593899033
## 2 -12.9528242696992       Aug       Jan 25.9499486598327
## 3 -4.53230818656997       Jul       Mar 9.66861458331014
## 4                 -         -         -                -
## 5 -5.67594441404319       Jun       Feb 16.1185828534379
## 6 -8.64134459386534       Jul       Dec 20.4089740355914
## 7 -8.87518052775784       Aug       Dec 18.9378564814146
```

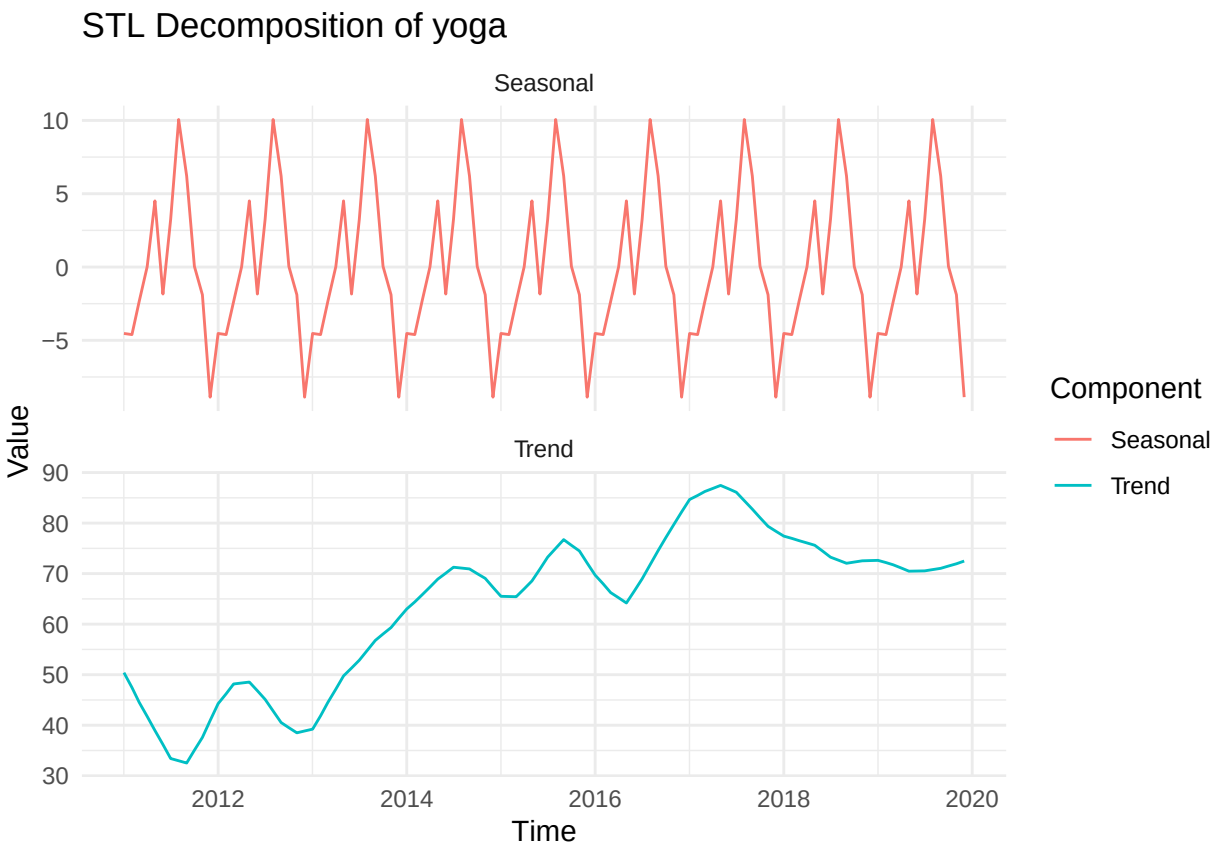
Seasonal Plot

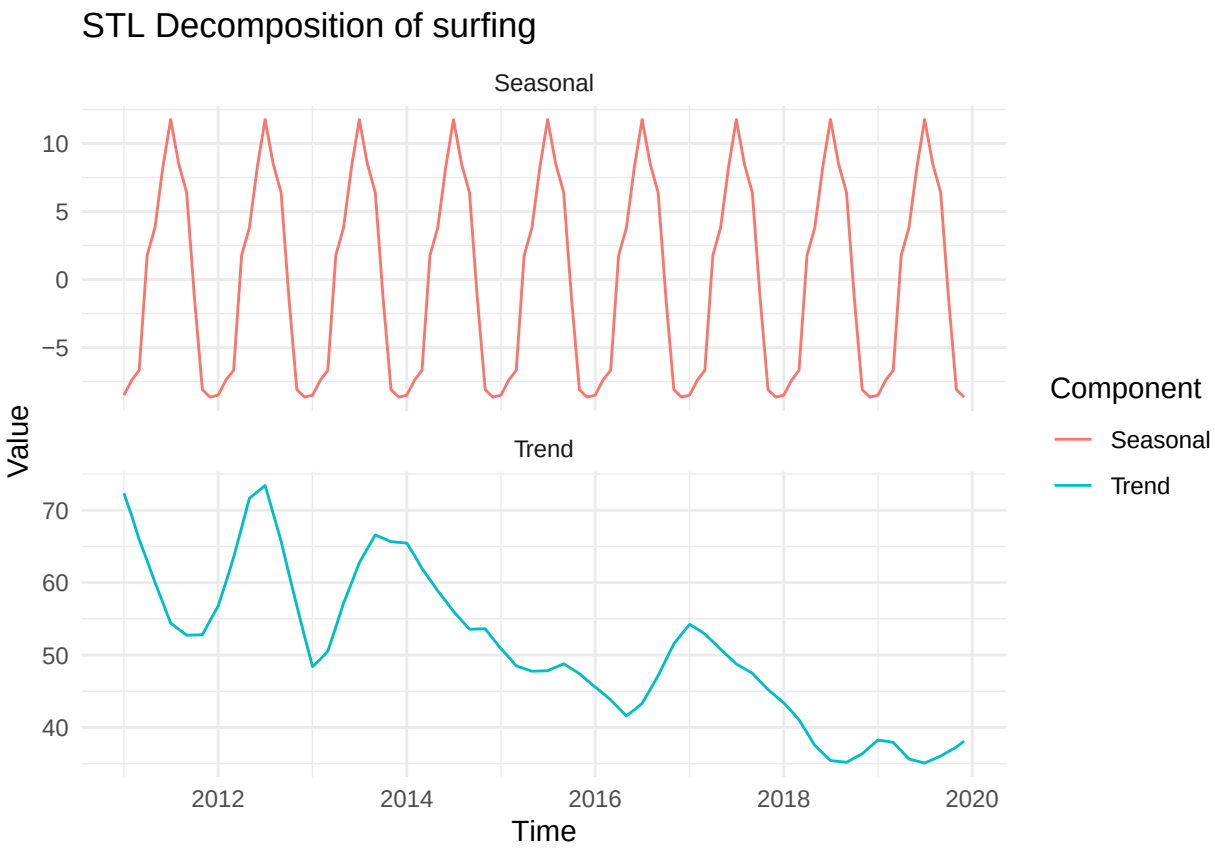
```
for (col in num_cols){
  ## Decomp
  col_ts <- ts(df[[col]], frequency = 12)
  decomp <- stl(col_ts, s.window="periodic", t.window=13)

  ## Extract the Secular and Seasonality
  trend <- decomp$time.series[, "trend"]
  seasonal <- decomp$time.series[, "seasonal"]

  ## Make into dataframe
  ts_df <- data.frame(
    Time = df$Month,
    Trend = as.numeric(trend),
    Seasonal = as.numeric(seasonal)
  ) %>% pivot_longer(cols = -Time, names_to = "Component", values_to = "Value")

  ## Plot
  plot_1 <- ggplot(ts_df, aes(x = Time, y = Value, color = Component)) +
    geom_line() +
    facet_wrap(~ Component, scales = "free_y", ncol = 1) +
```

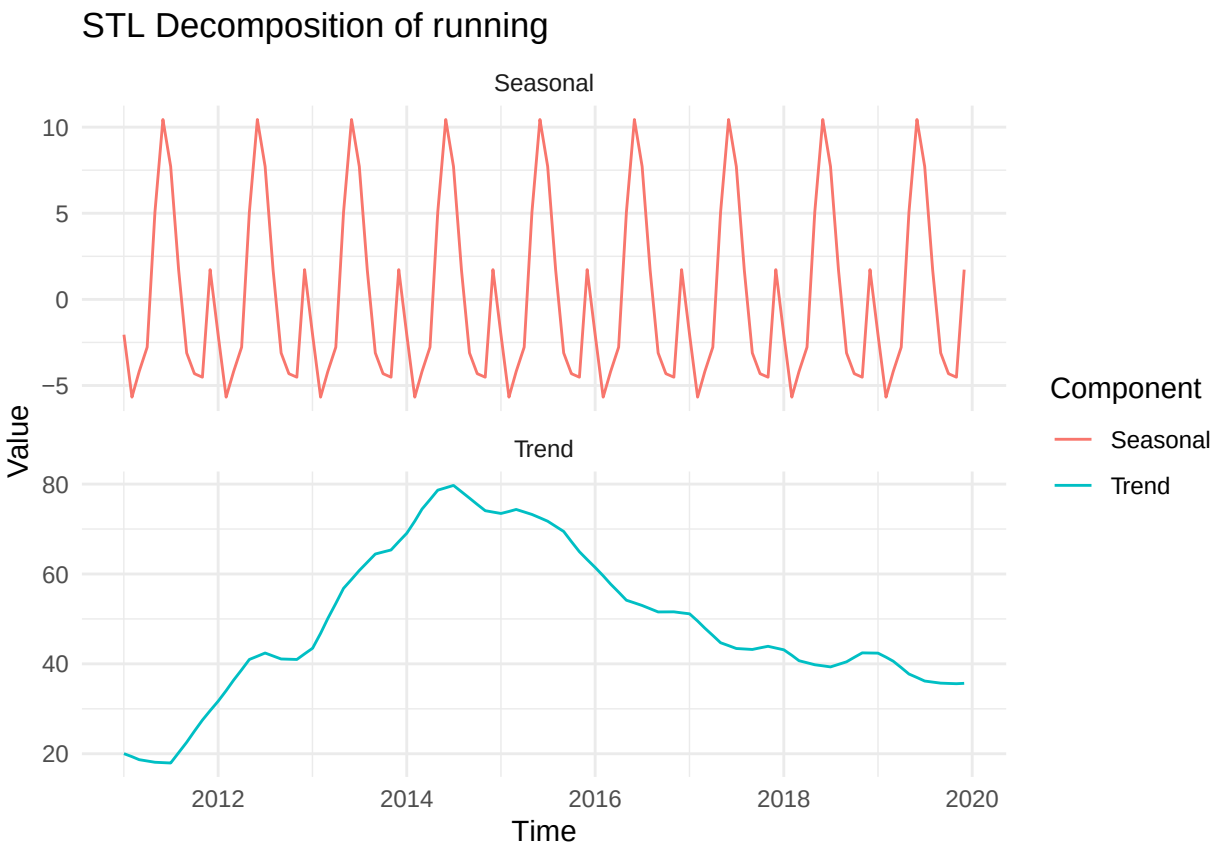
```
    theme_minimal() +
    labs(title = paste("STL Decomposition of", col))
  print(plot_1)

  ## Save plot for publication
  ggsave(plot_1,
       filename = paste(col, "trend & seasonality.png"),
       device = "png",
       height = 4, width = 8, units = "in")
}
```

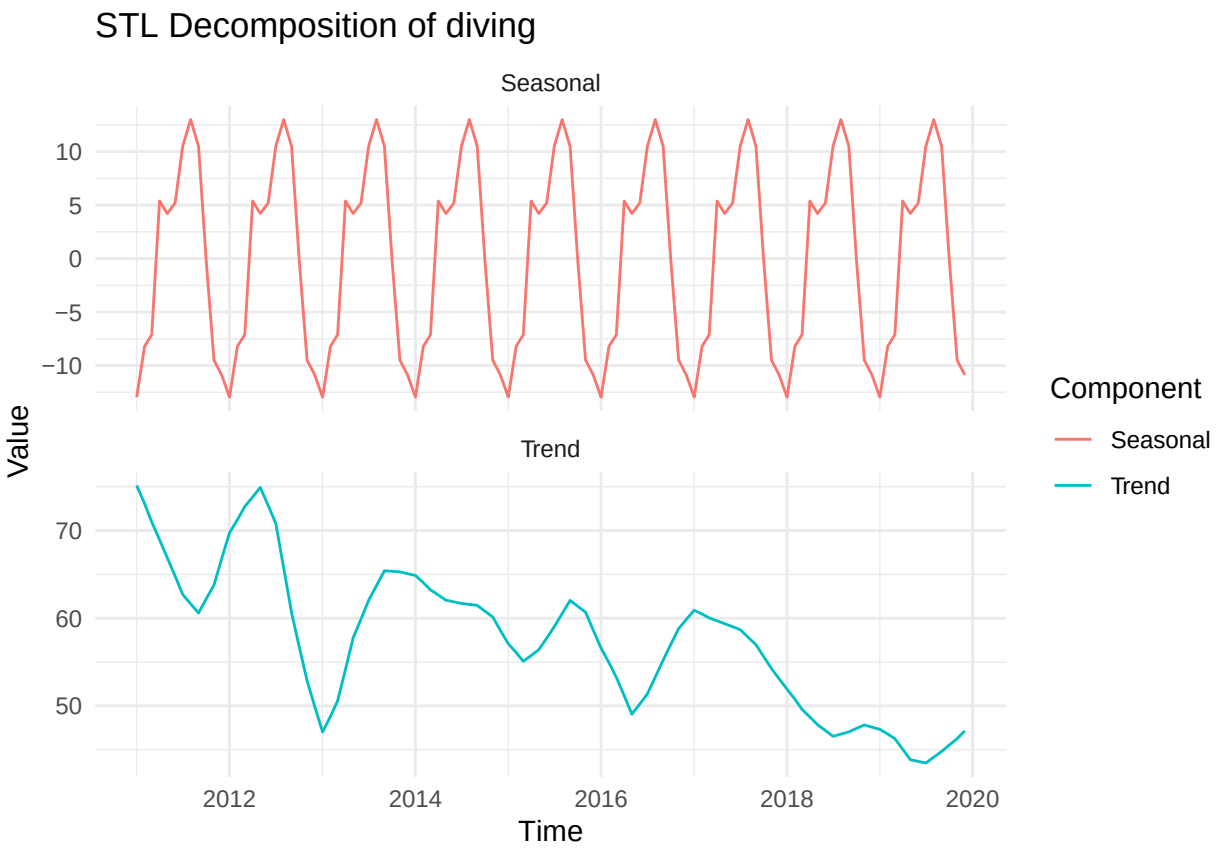



STL Decomposition of gym
Seasonal
5.0
2.5
0.0
−2.5
−5.0
Trend
70
60
50
40
30
20
2012
2014
2016
2018
2020
Time
Value
Component
Seasonal
Trend


STL Decomposition of yoga
Seasonal
10
5
0
−5
Trend
90
80
70
60
50
40
30
Value
2012
2014
2016
2018
2020
Time
Component
Seasonal
Trend


STL Decomposition of surfing
Seasonal
10
5
0
−5
Trend
70
60
50
40
Value
2012
2014
2016
2018
2020
Time
Component
Seasonal
Trend


STL Decomposition of running
Seasonal
Trend
Value
Time
10
5
0
−5
80
60
40
20
2012
2014
2016
2018
2020
Component
Seasonal
Trend


STL Decomposition of diving
Seasonal
10
5
0
−5
−10
Trend
70
60
50
Value
2012
2014
2016
2018
2020
Time
Component
Seasonal
Trend