STA4003 Project

Xinyi Liang 119010172

2023-05-09

First, we prepare the packages used in this assignment.

```
require(data.table)
```

```
## Loading required package: data.table
```

```
require(fpp3)
```

```
## Loading required package: fpp3

## -- Attaching packages --------------------------------------------- fpp3 0.5 --

## v tibble      3.1.6     v tsibble     1.1.3
## v dplyr       1.1.0     v tsibbledata 0.4.1
## v tidyr       1.1.4     v feasts      0.3.0
## v lubridate   1.8.0     v fable       0.3.2
## v ggplot2     3.3.6     v fabletools  0.3.2

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x dplyr::between()     masks data.table::between()
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x dplyr::first()       masks data.table::first()
## x lubridate::hour()    masks data.table::hour()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x lubridate::isoweek() masks data.table::isoweek()
## x tsibble::key()       masks data.table::key()
## x dplyr::lag()         masks stats::lag()
## x dplyr::last()        masks data.table::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x lubridate::second()  masks data.table::second()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
## x lubridate::wday()    masks data.table::wday()
## x lubridate::week()    masks data.table::week()
## x lubridate::yday()    masks data.table::yday()
## x lubridate::year()    masks data.table::year()
```

Process the data

read the data

```
Grading <- FALSE

if (Grading==TRUE){
  load("data2015.RData")
  load("data2016.RData")
  load("data2017.RData")
  load("data2018.RData")
  load("data2019.RData")
  Train1 <- data2015
  Train2 <- data2016
  Train3 <- data2017
  Train4 <- data2018
  Test1 <- data2019
} else {
  load("data2014.RData")
  load("data2015.RData")
  load("data2016.RData")
  load("data2017.RData")
  load("data2018.RData")
  Train1 <- data2014
  Train2 <- data2015
  Train3 <- data2016
  Train4 <- data2017
  Test1 <- data2018
}
```

Pre-process the data

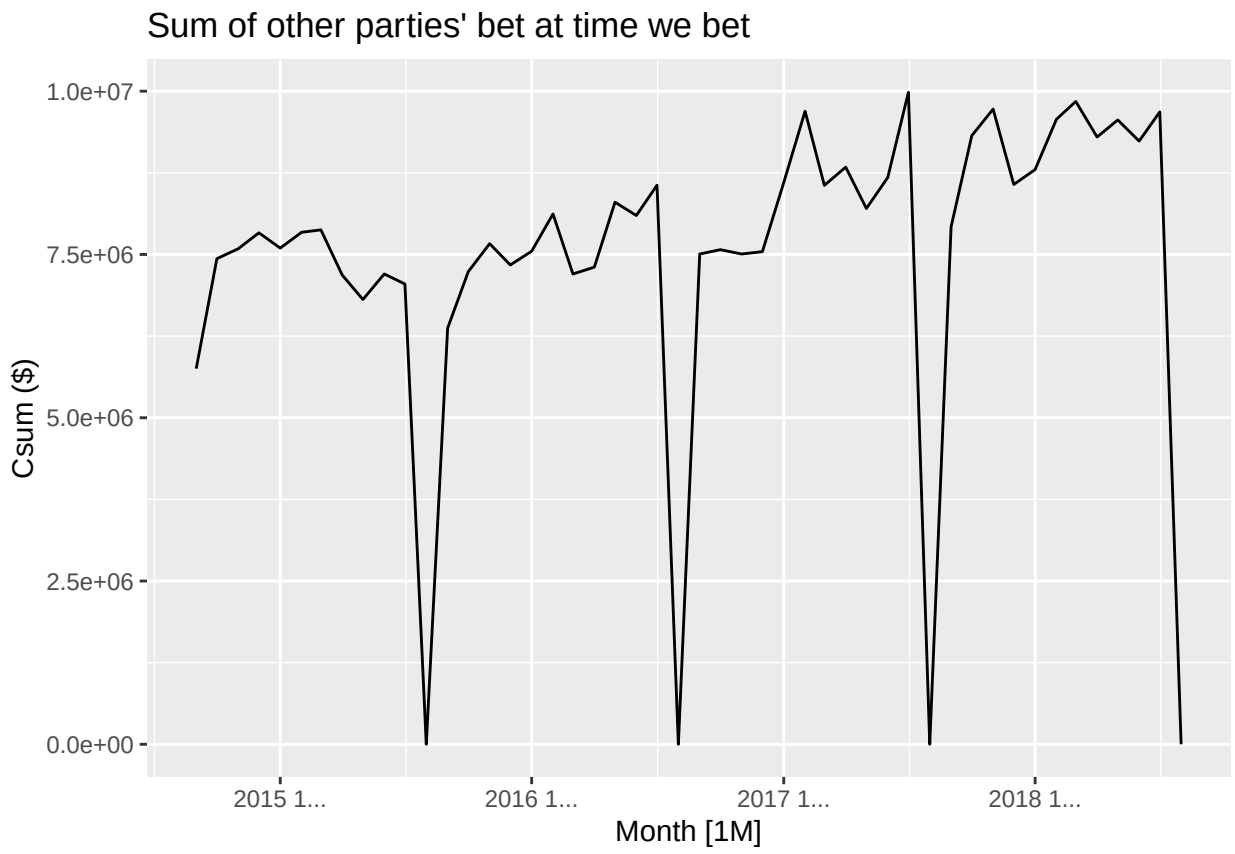
Looking into the dataset, we can see two facts about horse racing: 1. not everyday there is a horse racing; 2. there are more than one race on each day and the number of races can be different on each day. We can see that the horse racing data is irregularly-spaced time series. Therefore, to easily build a time series model, we have to transform the data into a regularly-spaced format. The data will be a monthly time series. All variables' values will be averaged across month.

```
Train1["Month"] <- as.Date(as.character(Train1$WIN_TIME.y),format = "%Y%m%d")
train1 <- Train1 %>%
  mutate(Month = yearmonth(Month)) %>%
  tsibble(index = Month, key = ID) %>%
  summarise(across(WIN_POOL.x:WIN_MODEL_14.y, mean)) %>%
  mutate(Csum = WIN_POOL.y - WIN_POOL.x)
# add August into data to bridge the gap in time
train1 <- train1 %>%
  add_row(Month = yearmonth("2015 Aug"))
train1[is.na(train1)] <- 0

Train2["Month"] <- as.Date(as.character(Train2$WIN_TIME.y),format = "%Y%m%d")
train2 <- Train2 %>%
```

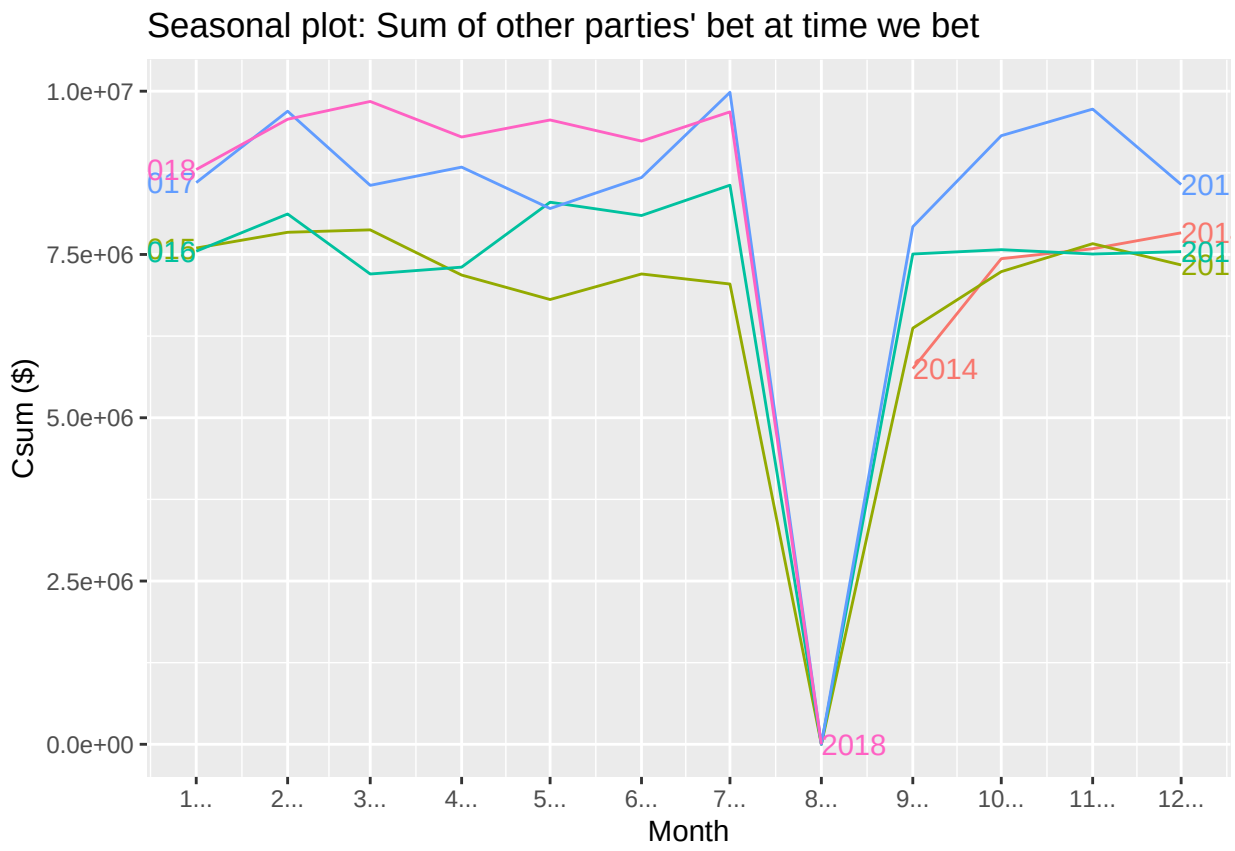
```
  mutate(Month = yearmonth(Month)) %>%
  tsibble(index = Month, key = ID) %>%
  summarise(across(WIN_POOL.x:WIN_MODEL_14.y, mean)) %>%
  mutate(Csum = WIN_POOL.y - WIN_POOL.x)
train2 <- train2 %>%
  add_row(Month = yearmonth("2016 Aug"))
train2[is.na(train2)] <- 0

Train3["Month"] <- as.Date(as.character(Train3$WIN_TIME.y),format = "%Y%m%d")
train3 <- Train3 %>%
  mutate(Month = yearmonth(Month)) %>%
  tsibble(index = Month, key = ID) %>%
  summarise(across(WIN_POOL.x:WIN_MODEL_14.y, mean)) %>%
  mutate(Csum = WIN_POOL.y - WIN_POOL.x)
train3 <- train3 %>%
  add_row(Month = yearmonth("2017 Aug"))
train3[is.na(train3)] <- 0

Train4["Month"] <- as.Date(as.character(Train4$WIN_TIME.y),format = "%Y%m%d")
train4 <- Train4 %>%
  mutate(Month = yearmonth(Month)) %>%
  tsibble(index = Month, key = ID) %>%
  summarise(across(WIN_POOL.x:WIN_MODEL_14.y, mean)) %>%
  mutate(Csum = WIN_POOL.y - WIN_POOL.x)
train4 <- train4 %>%
  add_row(Month = yearmonth("2018 Aug"))
train4[is.na(train4)] <- 0

train = bind_rows(train1, train2, train3, train4)
```

Time Series Graphics

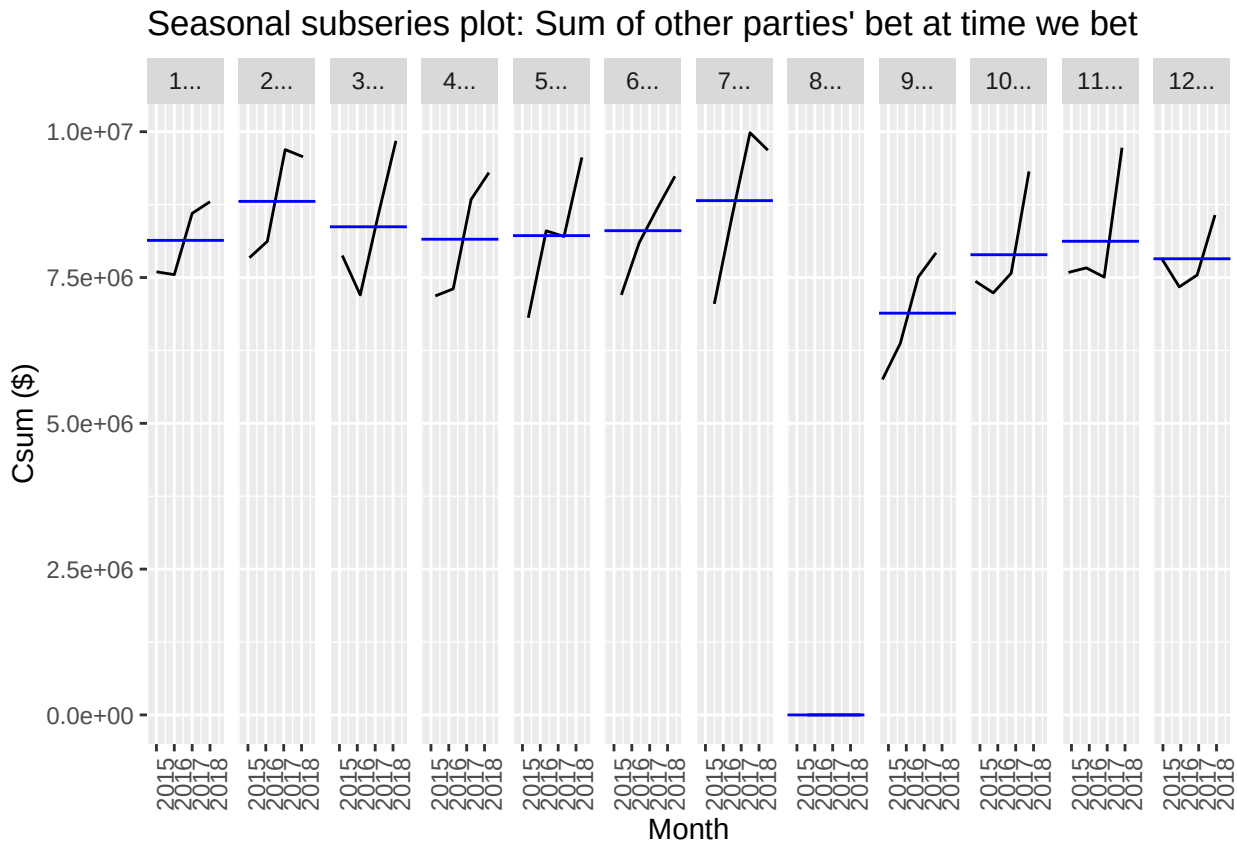
```
# Time plot
train %>%
  autoplot(Csum) +
  labs(y = "Csum ($)", title = "Sum of other parties' bet at time we bet")
```




```
# Seasonal plot
train %>%
  gg_season(Csum, labels = "both") +
  labs(y = "Csum ($)", title = "Seasonal plot: Sum of other parties' bet at time we bet")
```

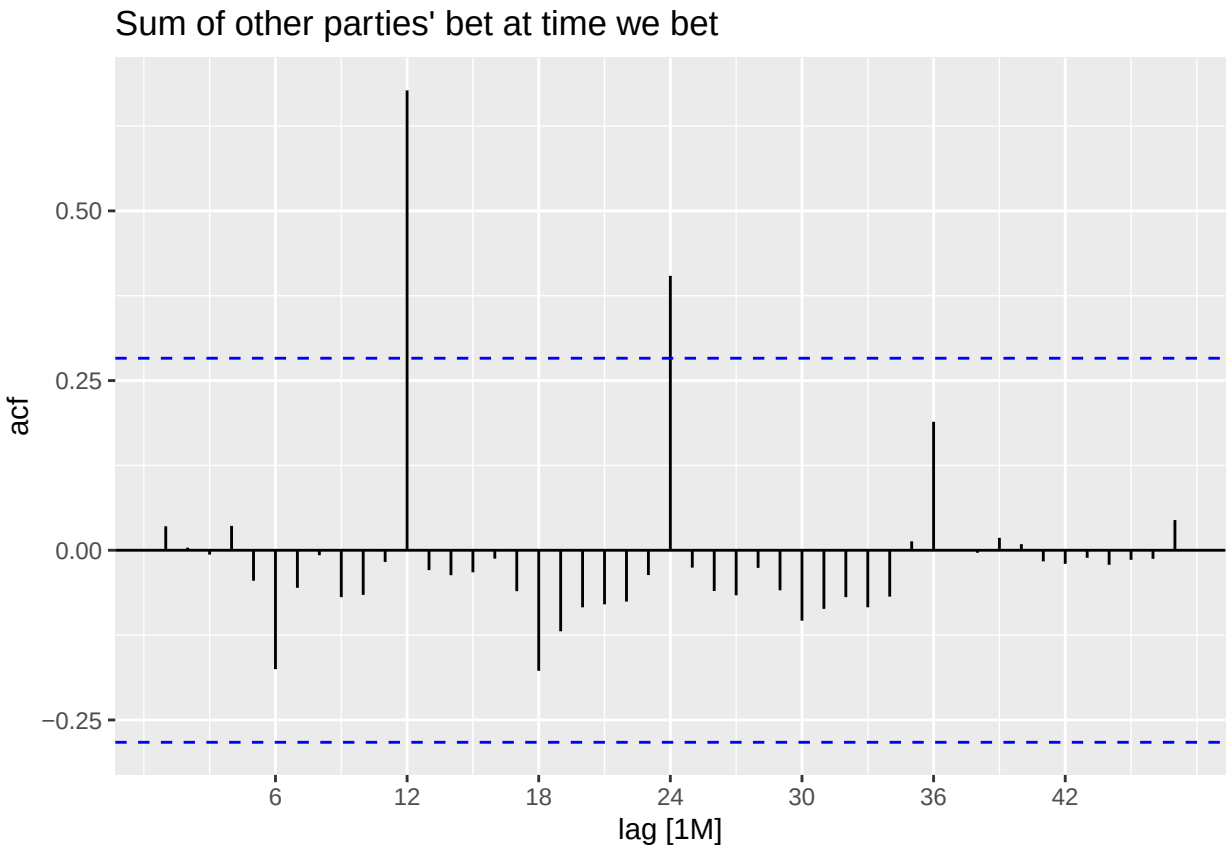



```
# Seasonal subseries plot
train %>%
  gg_subseries(Csum) +
  labs(y = "Csum ($)", title = "Seasonal subseries plot: Sum of other parties' bet at time we bet")
```



From the time plot, the monthly data shows seasonality and trend. We further check by seasonal and seasonal subseries plot. Please attention that there is actually no data in August every year. For the completeness of time series, I add August data into dataset by myself. For other months, there is a general increasing trend by years. From 2015 to 2016, the bet sum sometimes decreases too.

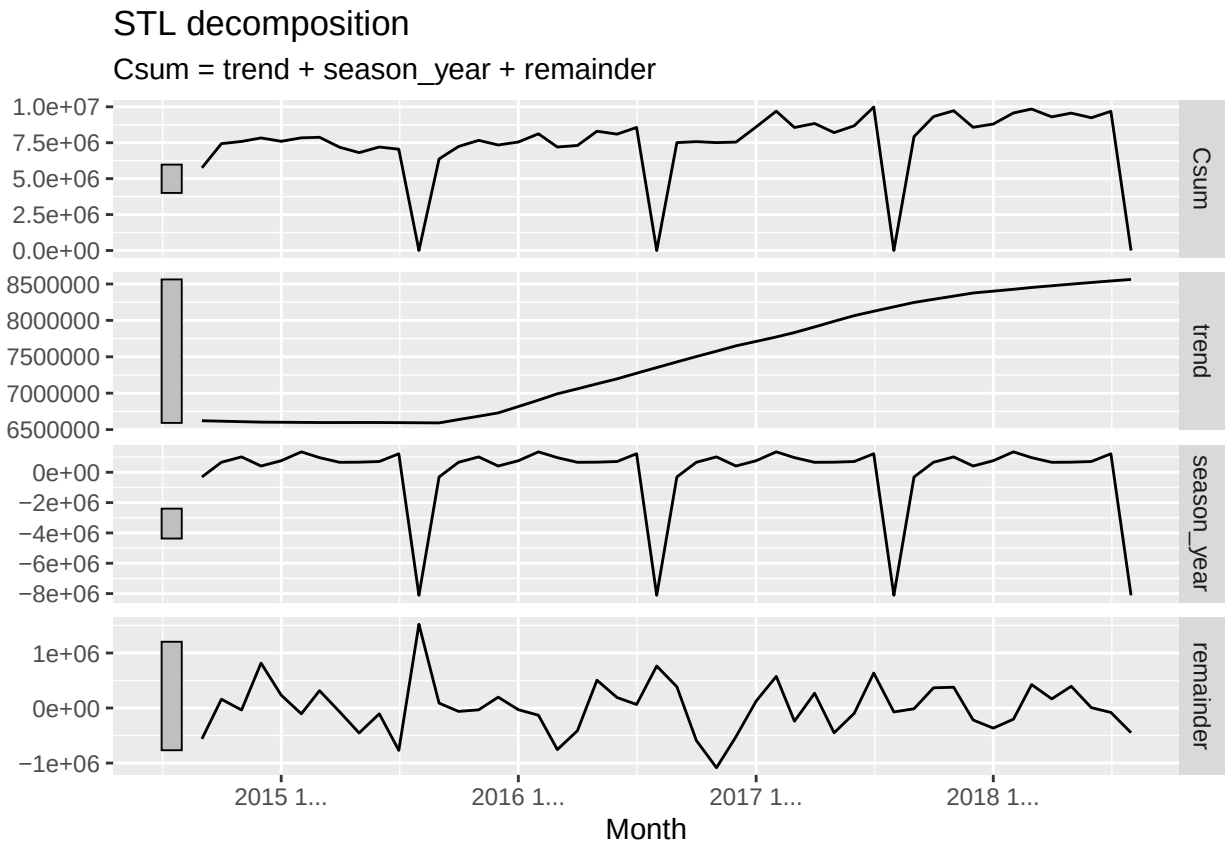
```
# Autocorrelation
train %>%
  ACF(Csum, lag_max = 48) %>%
  autoplot() + labs(title="Sum of other parties' bet at time we bet")
```



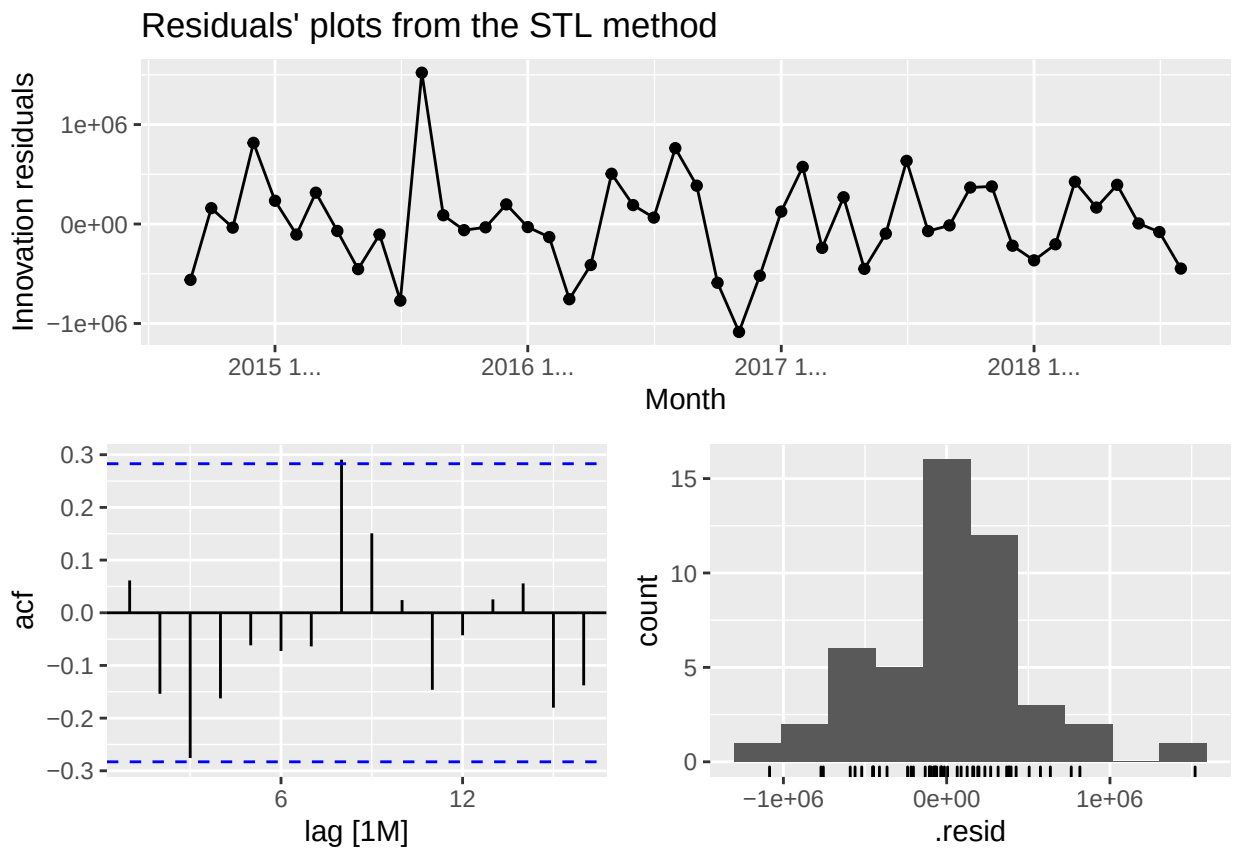
When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal period) than for other lags, so the lag is 12. And the ACF of a trended time series tends to have positive values that slowly decrease as the lags increase.

Time series decomposition

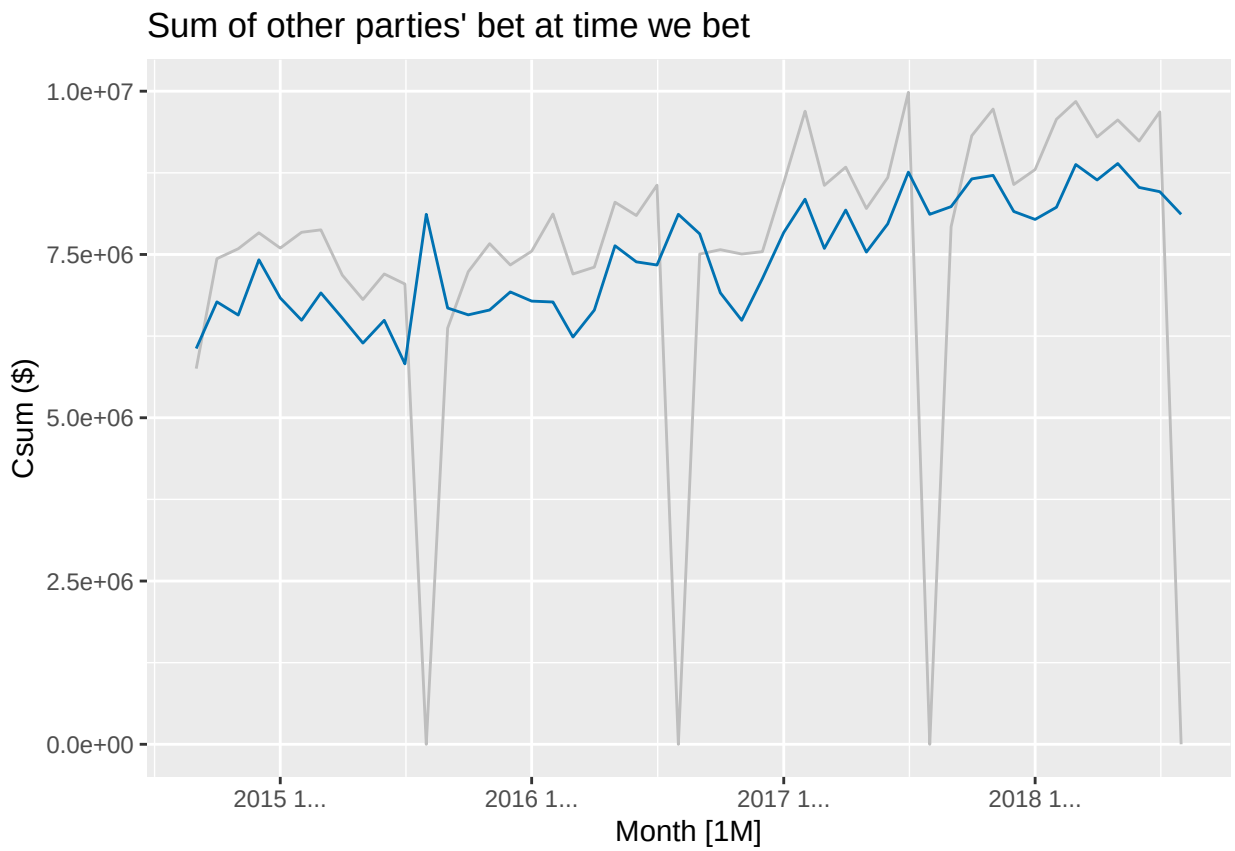
```
# STL: Seasonal and Trend decomposition using Loess
dcmp <- train %>%
  model(
    stl = STL(Csum ~ trend(window = 21) +
          season(window = "periodic"),
        robust = TRUE))
components(dcmp) %>% autoplot()
```




```
dcmp %>% gg_tsresiduals() +
  labs(title = "Residuals' plots from the STL method")
```




```
# seasonal adjusted data
components(dcmp) %>%
  as_tsibble() %>%
  autoplot(Csum, colour = "gray") +
  geom_line(aes(y=season_adjust), colour = "#0072B2") +
  labs(y = "Csum ($)",
       title = "Sum of other parties' bet at time we bet")
```



These plots show we can decompose the time series into trend part and seasonal part. The trend is nearly linear after 2016 (all we fit it as a ReLu function.). And the seasonal data shows an explicit seasonal pattern, which double confirms our guess.

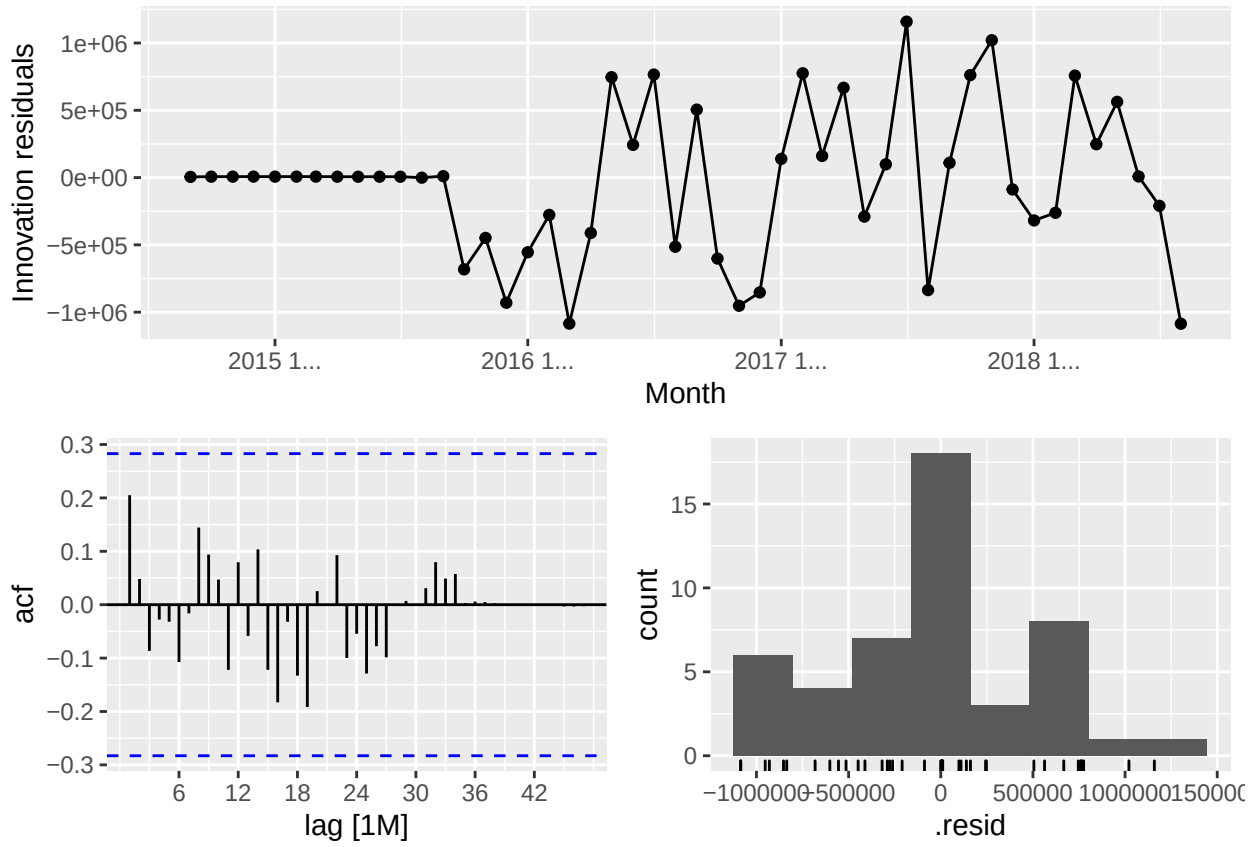
Model fitting

Because I have confirmed the seasonality and trend in the time series, Therefore, I fit the ARIMA model on the transformed data. Selecting appropriate values for p, d and q can be difficult. However, the ARIMA() function from the fable package will do it for you automatically.

```
## ARIMA
fit <- train %>%
  model(ARIMA(Csum))
report(fit)
```

```
## Series: Csum
## Model: ARIMA(0,0,0)(0,1,1)[12] w/ drift
##
## Coefficients:
##          sma1   constant
##       -0.6313  607738.57
## s.e.   0.4124   75318.58
##
## sigma^2 estimated as 4.176e+11:  log likelihood=-534.59
## AIC=1075.18   AICc=1075.93   BIC=1079.93
```

```
fit %>% gg_tsresiduals(lag_max=48)
```

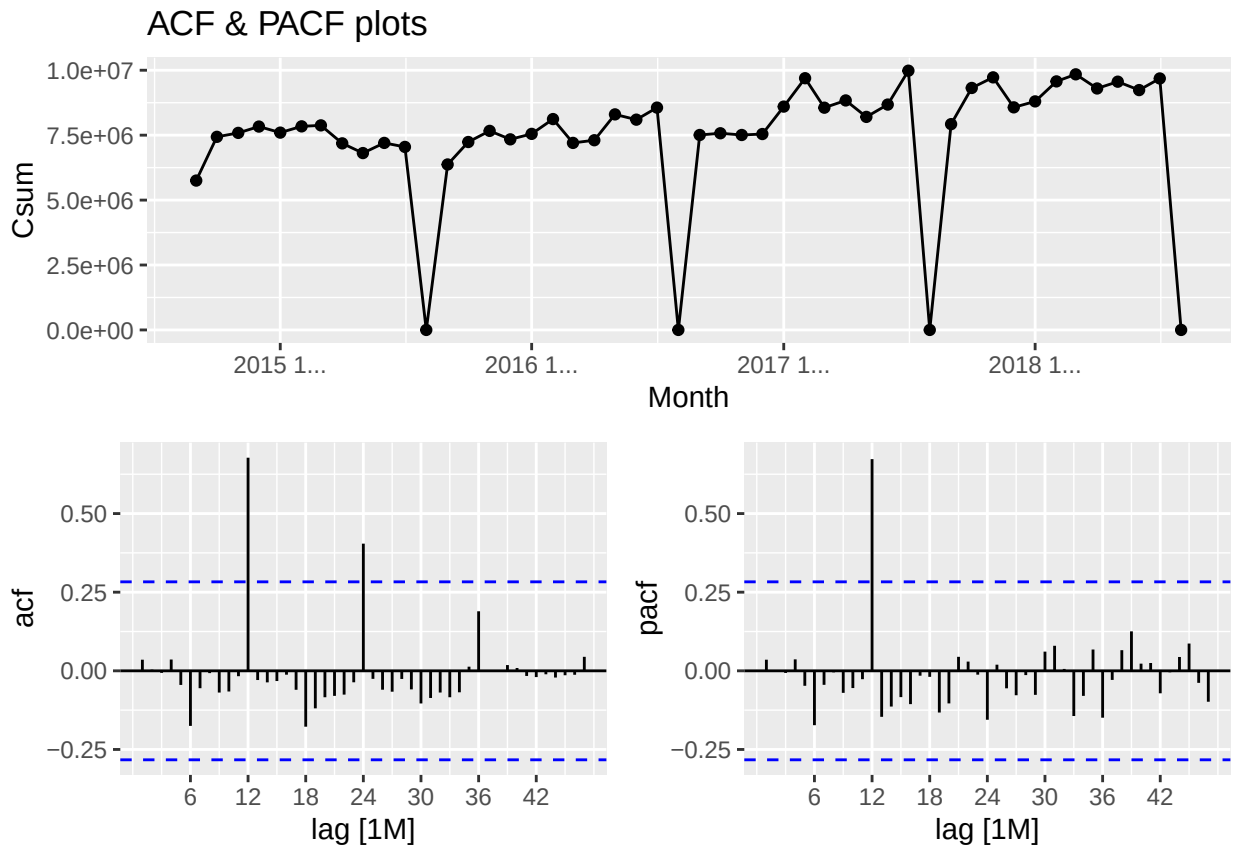


The residuals' ACF plot seems like white noise, which proves that our model is actually good enough. However, the histogram of residuals is left-skewed, which says that the residuals may not follow normal distribution. And the accuracy of this model is:

```
accuracy(fit)
```

```
## # A tibble: 1 x 10
##   .model      .type         ME    RMSE     MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>       <chr>      <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Csum) Training -33073. 543844. 400293.  -Inf   Inf 0.564 0.581 0.205
```

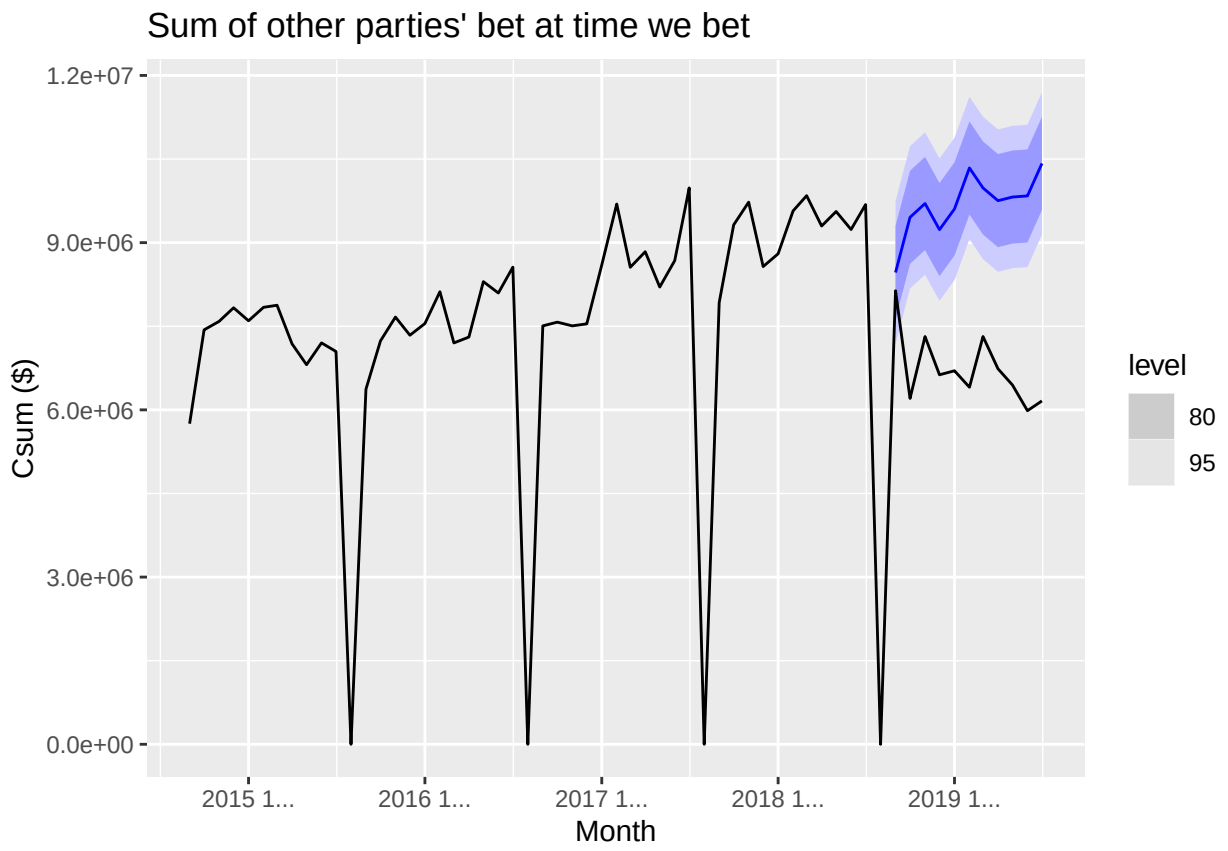
```
# ACF & PACF plots
train %>%
  gg_tsdisplay(Csum, plot_type = "partial", lag_max = 48) +
  labs(title = "ACF & PACF plots")
```



An ARIMA(0,0,0)(1,0,0)_12 model will show: 1. exponential decay in the seasonal lags of the ACF; 2. a single significant spike at lag 12 in the PACF.

Test the model

```
Test1["Month"] <- as.Date(as.character(Test1$WIN_TIME.y),format = "%Y%m%d")
test <- Test1 %>%
  mutate(Month = yearmonth(Month)) %>%
  tsibble(index = Month, key = ID) %>%
  summarise(across(WIN_POOL.x:WIN_MODEL_14.y, mean)) %>%
  mutate(Csum = WIN_POOL.y - WIN_POOL.x)

fit %>%
  forecast(h=11) %>%
  autoplot(bind_rows(train, test)) +
  labs(y = "Csum ($)", title = "Sum of other parties' bet at time we bet")
```




```
fct <- fit %>%
  forecast(h=11) %>%
  select(-.model)

Test1["Month"] <- yearmonth(Test1$Month)

Csum <- Test1 %>% mutate(Csum=WIN_POOL.y-WIN_POOL.x) %>% pull(Csum)
Csum_est <- Test1 %>%
  left_join(fct, by = "Month") %>%
  pull(.mean)

MSE <- mean((Csum_est-Csum)^2)
MSE
```

```
## [1] 1.358599e+13
```

Because the forecast is based on the model trained on the transformed data, we should transform the forecast back to the test data. The sum of bet would be the same within the same month. The MSE is 1.358599e+13.

Reflection

From the last plot, we can see that although our forecast is in line with the previous pattern, but it is far away from the real average data. This is maybe because we lose some information when we average the monthly data and treat every horse as the same. Therefore, it is better for us to forecast every horse's Csum.