stat4990_pj

Koki Yamanaka

2023-11-30

Introduction

- The Dow Jones Industrial Average (^DJI) is a price-weighted index that tracks 30 large, public companies trading on the New York Stock Exchange and the Nasdaq. In a way, it represents the overall market and the largest sectors of U.S. economy. most expensive stocks on the index (UNH. HD, GS)
- DJI current prices of 30 stocks make up index are added then divided by dow divisor.

imports

```r
library(quantmod) # download from yahoo
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

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

```r
library(dplyr) # pipe operator
```

```
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
```

```
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop          #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last

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

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

```r
library(tsibble)
```

```
##
## Attaching package: 'tsibble'

## The following object is masked from 'package:zoo':
##
##     index

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

```r
library(ggplot2) # autoplot
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
## v readr     2.1.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()       masks stats::filter()
## x dplyr::first()        masks xts::first()
## x lubridate::interval() masks tsibble::interval()
## x dplyr::lag()          masks stats::lag()
## x dplyr::last()         masks xts::last()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(fable) # model()
```

```
## Loading required package: fabletools
```

```r
library(feasts) # model()

# install.packages('patchwork')
library(patchwork) # combine 2 plots
```

```
## Warning: package 'patchwork' was built under R version 4.3.2
```

Data preparation (tsibble)

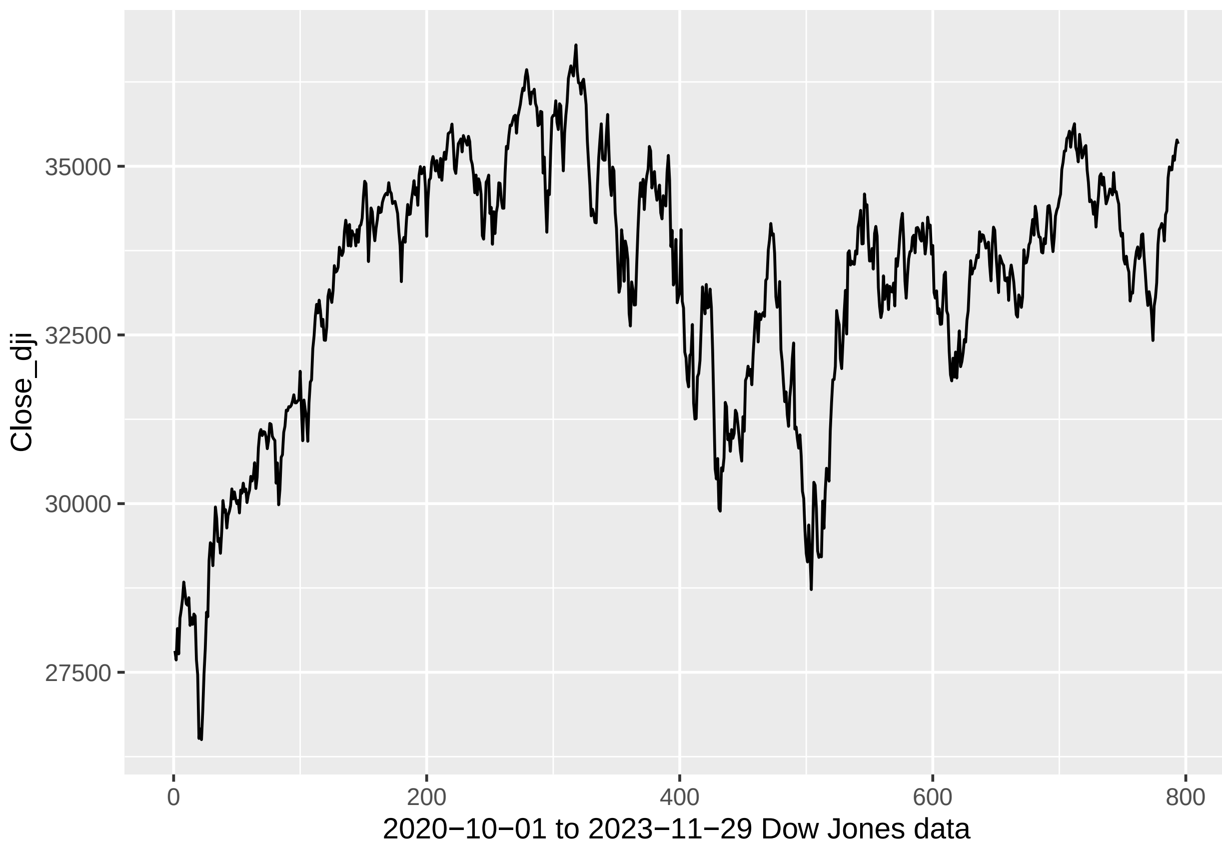
```r
# Download data from yahoo Finance!
# extract 3 years as train set and 2 months as test
start.date = '2020-10-01'     # starting date of stock
end.date = '2023-11-28'       # ending date of stock

# Download the selected stocks from Yahoo finance using `quantmod` package
getSymbols("^DJI", src = "yahoo", from = start.date, to = end.date, auto.assign = TRUE)
```

```
## [1] "DJI"
```

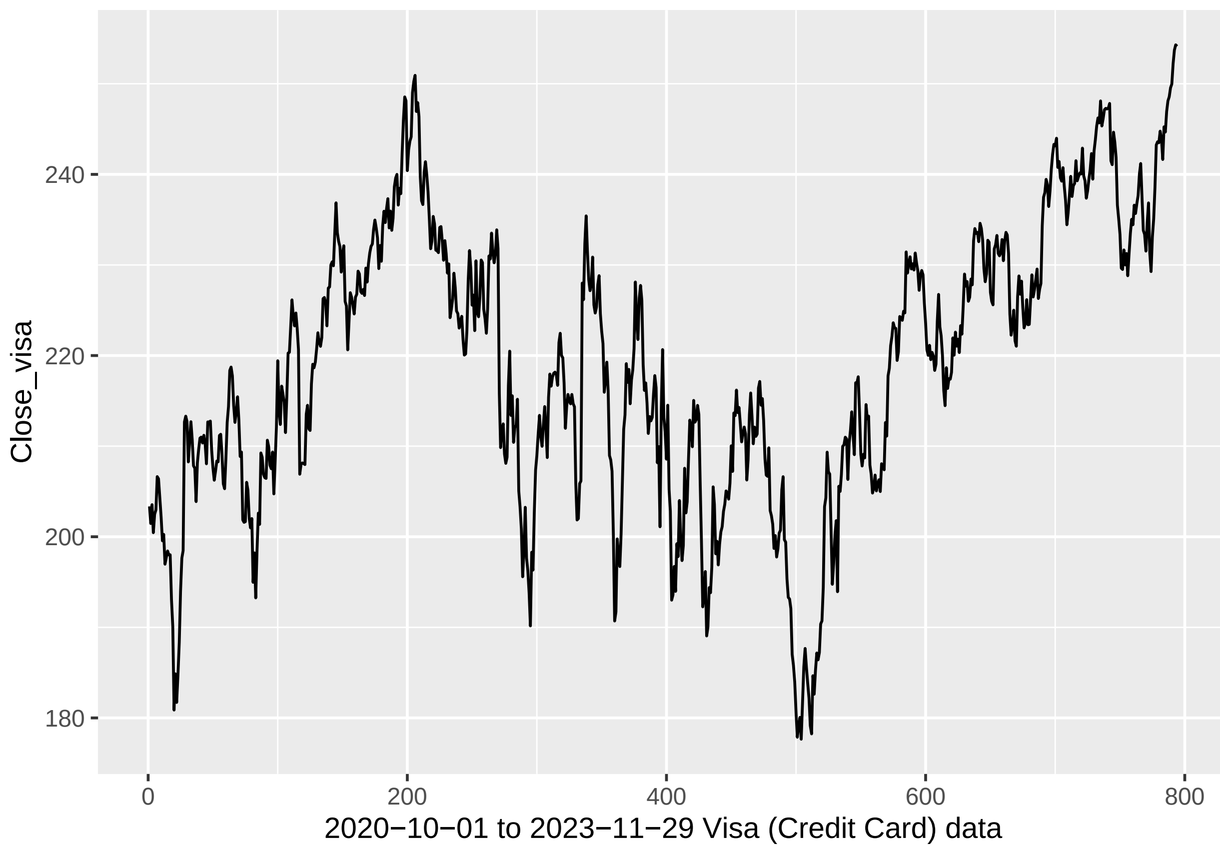
```r
getSymbols("V", src = "yahoo", from = start.date, to = end.date, auto.assign = TRUE)
```

```
## [1] "V"
```

```r
# take close price
DJI = DJI$DJI.Close
V = V$V.Close

# Create date variable and rename a few columns
DJI <- zoo::fortify.zoo(DJI)
DJI <- DJI %>% rename(c("Date" = "Index", "Close_dji" = "DJI.Close"))
Visa <- zoo::fortify.zoo(V)
Visa <- Visa %>% rename(c("Date" = "Index", "Close_visa" = "V.Close"))

# merge DJI AND Visa in df zoo object
data <- merge(DJI,Visa)

# create a tsibble assign Date column as time index
data <- as_tsibble(data, index = Date)

# create a new column to assign a unique row number to each row,, relocate unique row number to the fro
data <- data |>
  mutate(day = row_number()) |>
  update_tsibble(index = day, regular = TRUE) |>
  relocate(day)

# plot the close price of both
data |> autoplot(Close_dji) +   labs(x = "2020-10-01 to 2023-11-29 Dow Jones data")
```




```
data |> autoplot(Close_visa) +   labs(x = "2020-10-01 to 2023-11-29 Visa (Credit Card) data")
```



We see the plot are quite similar to each other. Note, visa accounts for 4% of dow jones index.

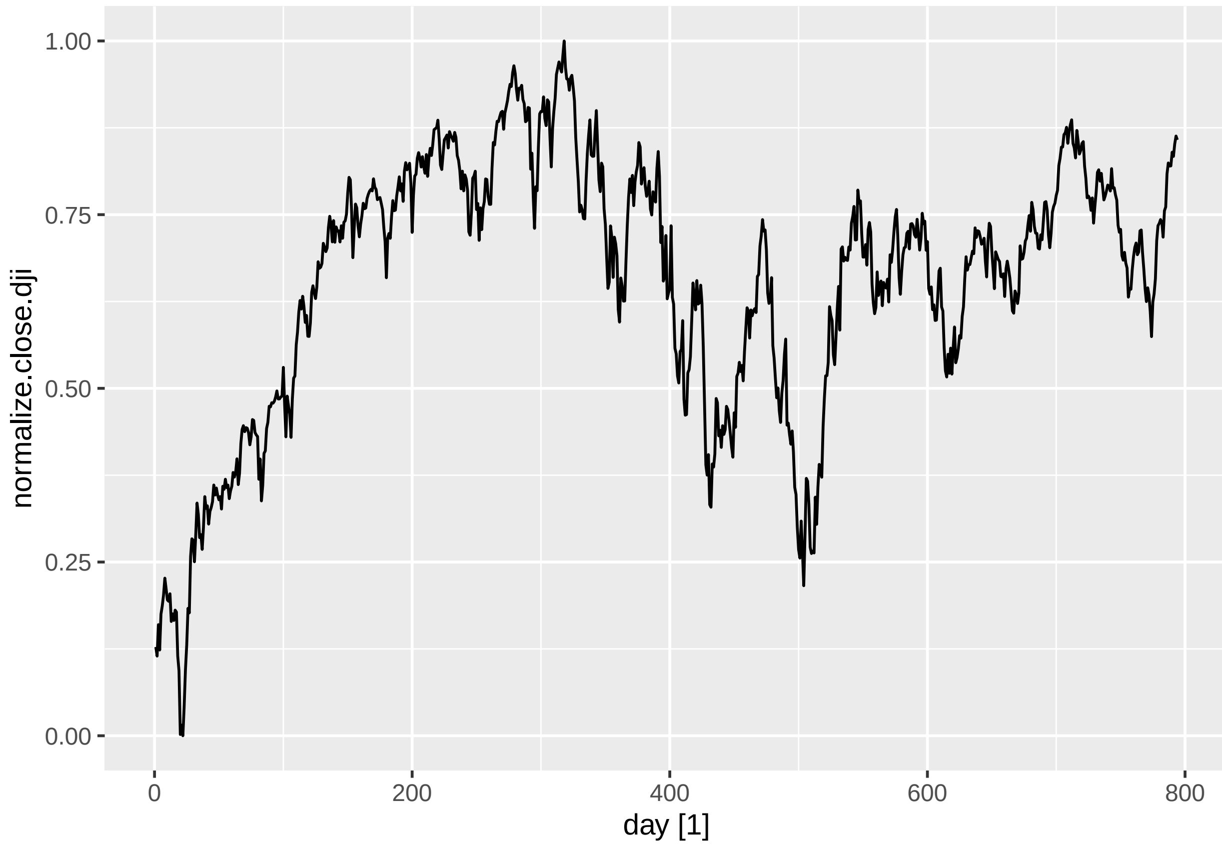
Exploration

```
# normalize 2 closing prices using minmaxscaler
data <- data %>% # for dji
  mutate(normalize.close.dji = (Close_dji - min(Close_dji)) / (max(Close_dji) - min(Close_dji)))
data <- data %>% # for visa
  mutate(normalize.close.visa = (Close_visa - min(Close_visa)) / (max(Close_visa) - min(Close_visa)))

data
```

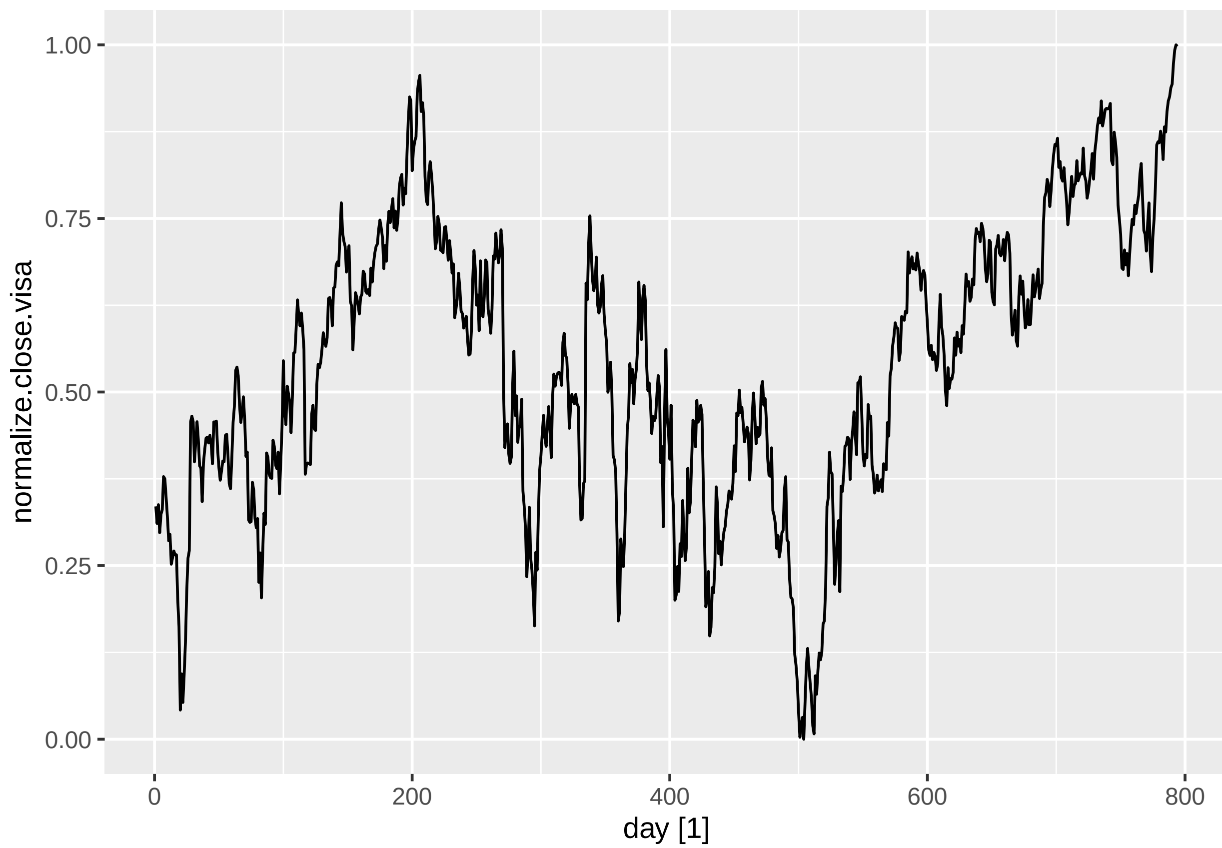
```
## # A tsibble: 794 x 6 [1]
##      day Date       Close_dji Close_visa normalize.close.dji
##    <int> <date>         <dbl>      <dbl>               <dbl>
##  1     1 2020-10-01    27817.       203.               0.128
##  2     2 2020-10-02    27683.       201.               0.115
##  3     3 2020-10-05    28149.       204.               0.160
##  4     4 2020-10-06    27773.       200.               0.123
##  5     5 2020-10-07    28303.       202.               0.175
##  6     6 2020-10-08    28426.       203.               0.187
##  7     7 2020-10-09    28587.       207.               0.202
##  8     8 2020-10-12    28838.       206.               0.227
##  9     9 2020-10-13    28680.       204.               0.212
## 10    10 2020-10-14    28514        202.               0.195
```

```
## # i 784 more rows
## # i 1 more variable: normalize.close.visa <dbl>
```

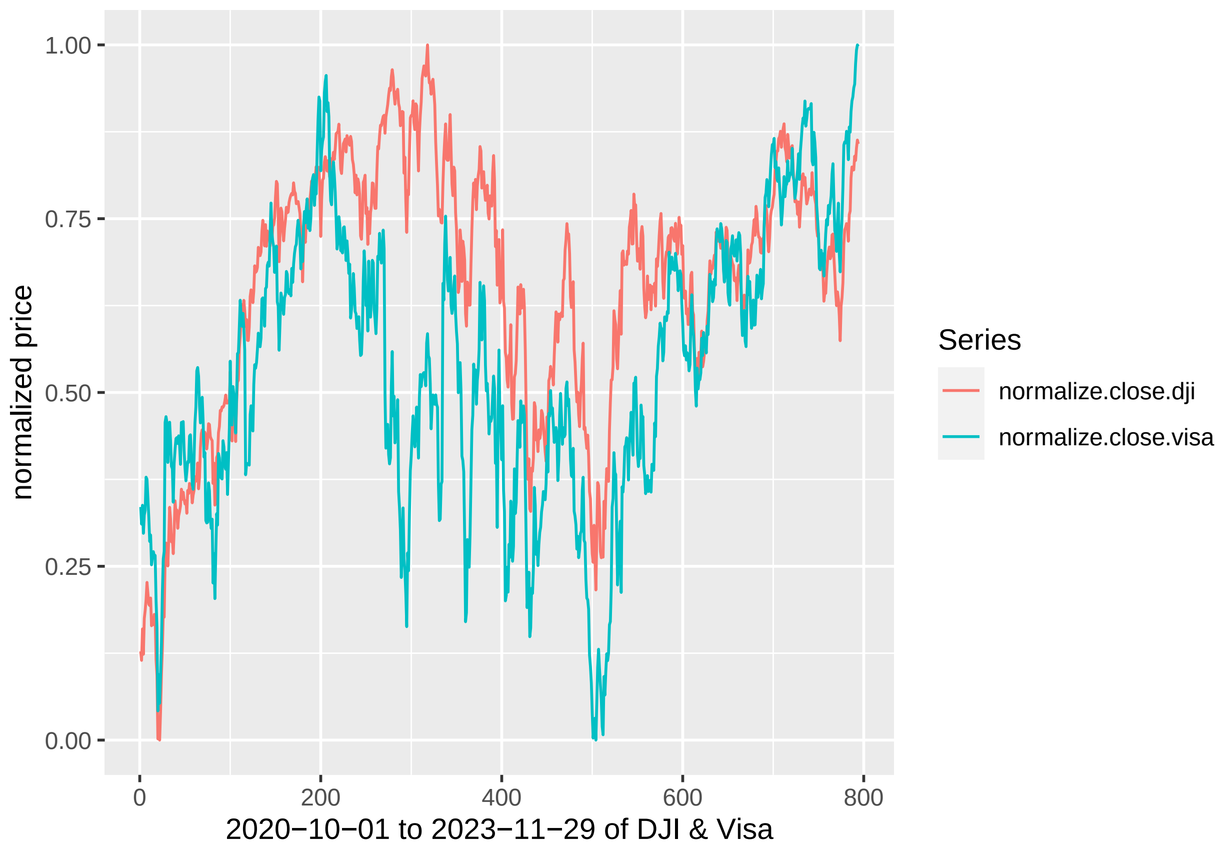
```
data |> autoplot(normalize.close.dji)
```




```
data |> autoplot(normalize.close.visa)
```




```
data |>
  pivot_longer(c(normalize.close.dji, normalize.close.visa), names_to="Series") |>
  autoplot(value) +
  labs(y = "normalized price") +
  labs(x = "2020-10-01 to 2023-11-29 of DJI & Visa")
```

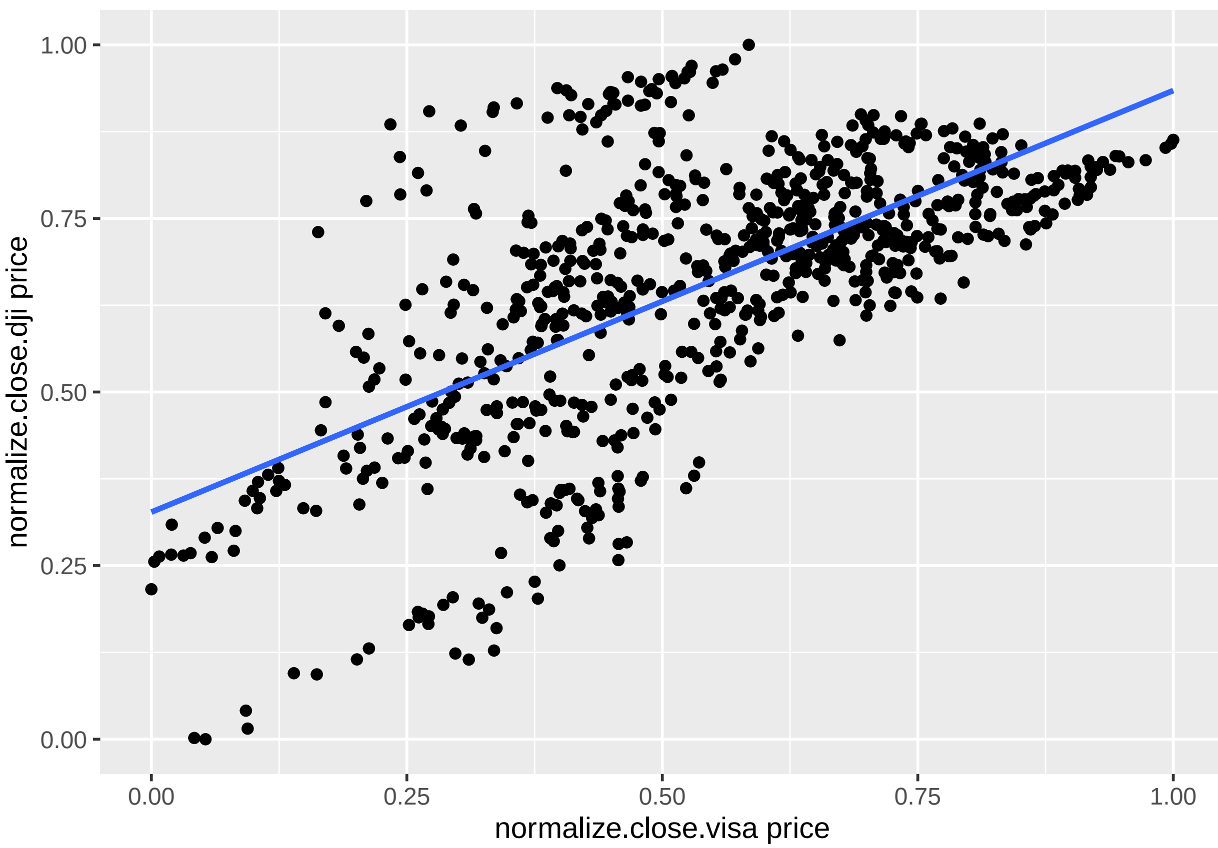


The moving pattern are quite similar.

Correlation analysis

```
# plot normalize.close.visa vs normalize.close.dji
data %>%
  ggplot(aes(x = normalize.close.visa, y = normalize.close.dji)) +
  labs(y = "normalize.close.dji price",
  x = "normalize.close.visa price") +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



There is a positive correlation between visa and dji. Overall data follows homoscedasticity, but bottom left has some heteroscedasticity.

train/test split

```
# filter train set
train_dow_jones <- data |>
  filter(between(Date, as.Date("2020-10-01"), as.Date("2023-09-30")))
# filter test set
test_dow_jones <- data  |>
  filter(between(Date, as.Date("2023-10-01"), as.Date("2023-11-29")))
# note : weekends observation is omitted.
```

Linear regression

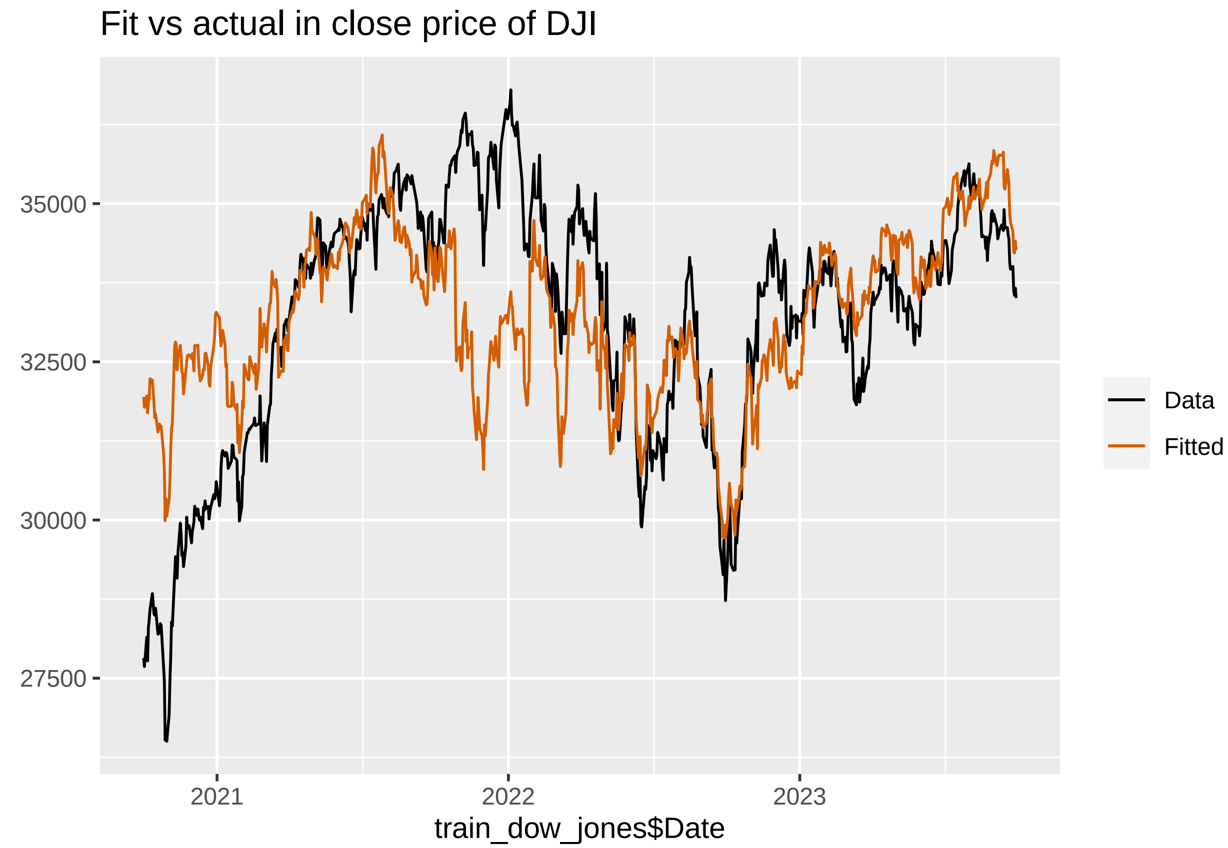
Fit a time series linear regression model

```
 # Fit a time series linear regression model with close_visa as predictors
fit_cons <- train_dow_jones %>%
  model(lm = TSLM(Close_dji ~ Close_visa))
# report the results
report(fit_cons)
```

```
## Series: Close_dji
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4128.81  -812.22   -51.23   826.25  4349.57
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14253.175    789.864   18.05   <2e-16 ***
## Close_visa     87.005      3.619   24.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1496 on 752 degrees of freedom
## Multiple R-squared: 0.4346,  Adjusted R-squared: 0.4338
## F-statistic:   578 on 1 and 752 DF, p-value: < 2.22e-16
```

Plot fit model on actual observations

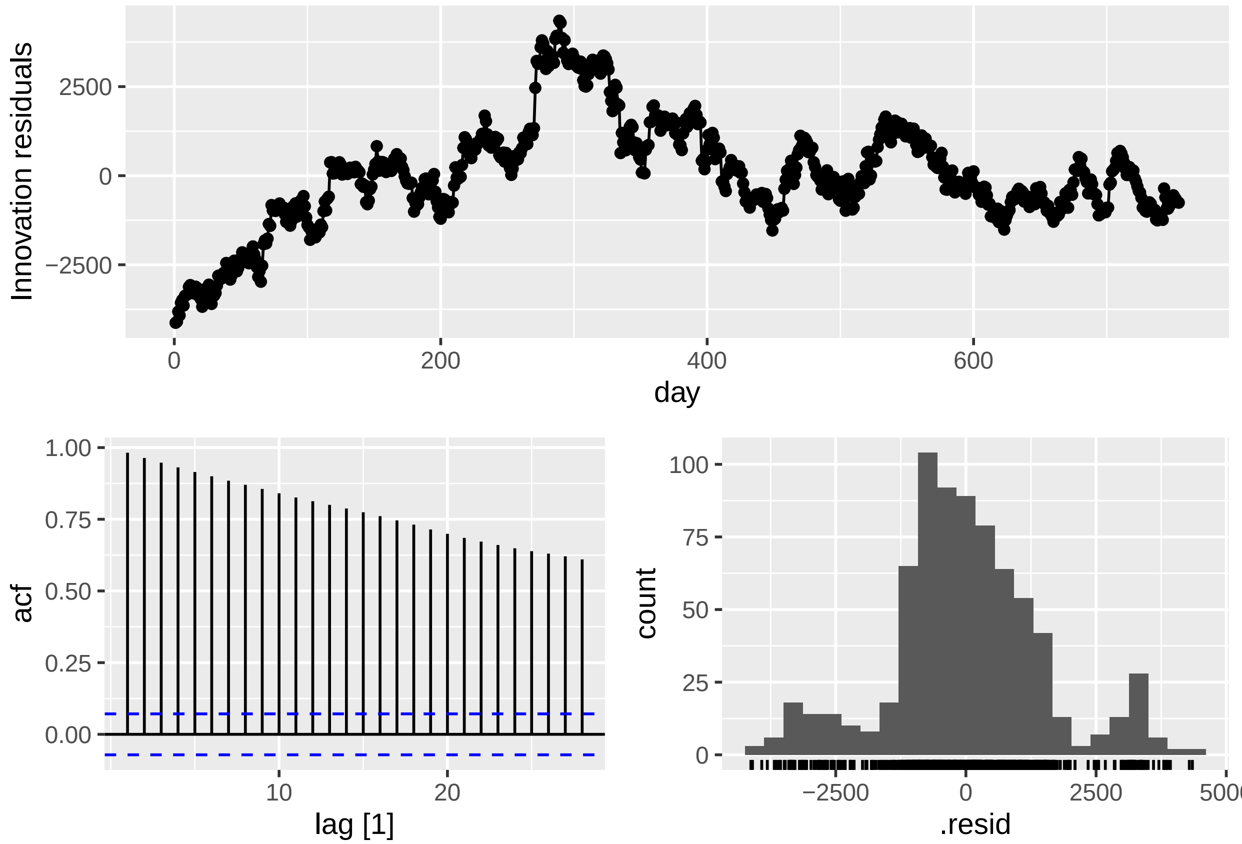
```r
augment(fit_cons) |>
  ggplot(aes(x = train_dow_jones$Date)) +
  geom_line(aes(y = Close_dji, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(y = NULL,
    title = "Fit vs actual in close price of DJI"
  ) +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```



our model somehow fit the actual observations. notice, is exact visa price

Check residuals

```
fit_cons %>% gg_tsresiduals()
```



- The residuals seems reasonable. Because,first 200 days follows an upward-trend which reflects our actual observations. - After 200 days, residuals bounds close to 0, which indicates some white noise. (random walk). The
- Our distribution somehow follows normal. - Thus, we say our model is suitable for series after day 200th.

fit_trends <- marathon %>% model(# Linear trend linear = TSLM(Minutes ~ trend()), # Exponential trend exponential = TSLM(log(Minutes) ~ trend()), # Piecewise linear trend piecewise = TSLM(Minutes ~ trend(knots = c(1940, 1980))))

Dynamic regression

Recall : a regression model with other predictors and errors are correlated. correlated errors capture past sequences to improve accuracy.

fit

```
# Fit a dynamic regression model and visa close as predictor
fit_lr_sarima <- train_dow_jones %>%
  model(ARIMA(Close_dji ~ Close_visa))

# report the results
report(fit_lr_sarima)
```

```
## Series: Close_dji
```

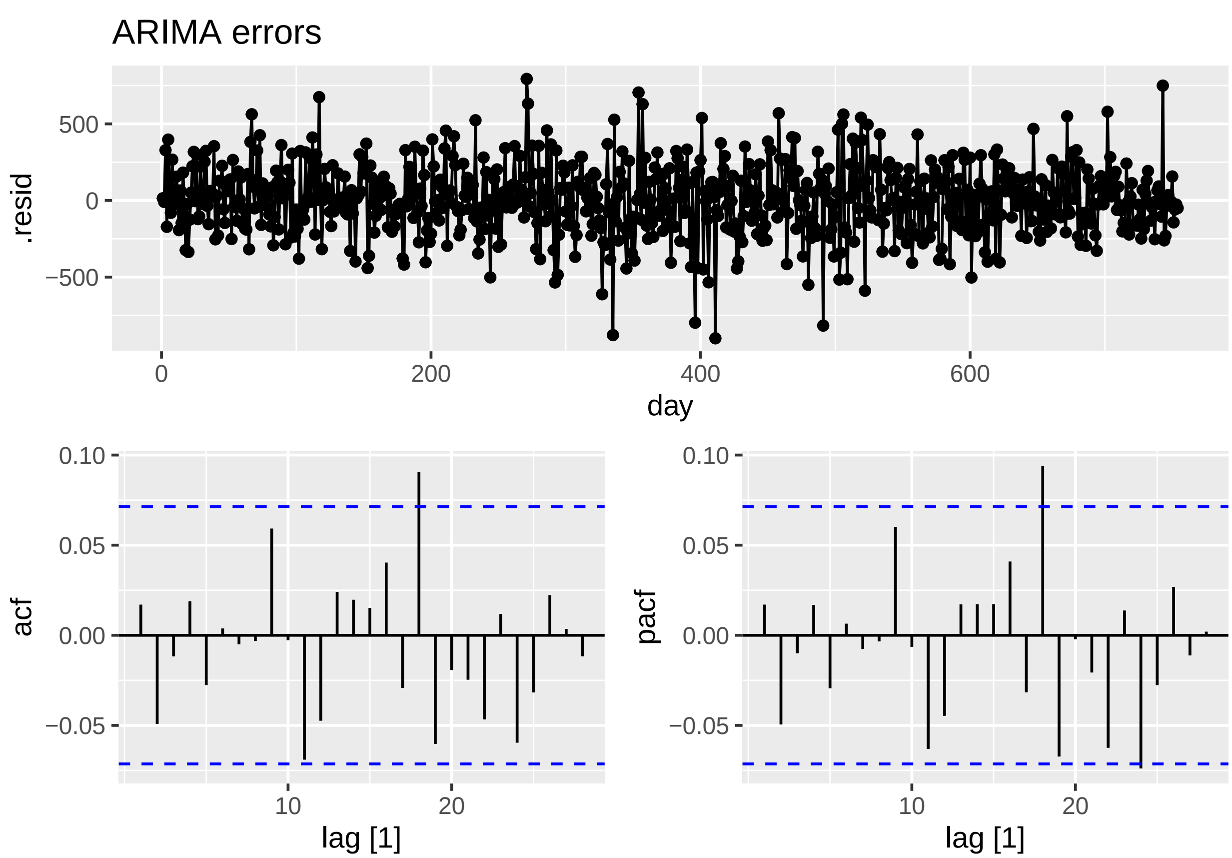
```
## Model: LM w/ ARIMA(0,1,0) errors
##
## Coefficients:
##       Close_visa
##          66.0573
## s.e.      2.4694
##
## sigma^2 estimated as 51719:  log likelihood=-5154.33
## AIC=10312.66   AICc=10312.68   BIC=10321.91
```

- we see dynamic's coefficient is 66.05 which is less than static regression. This implies dynamic allocate more weights to the series in close_dji itself rather than the predictors.

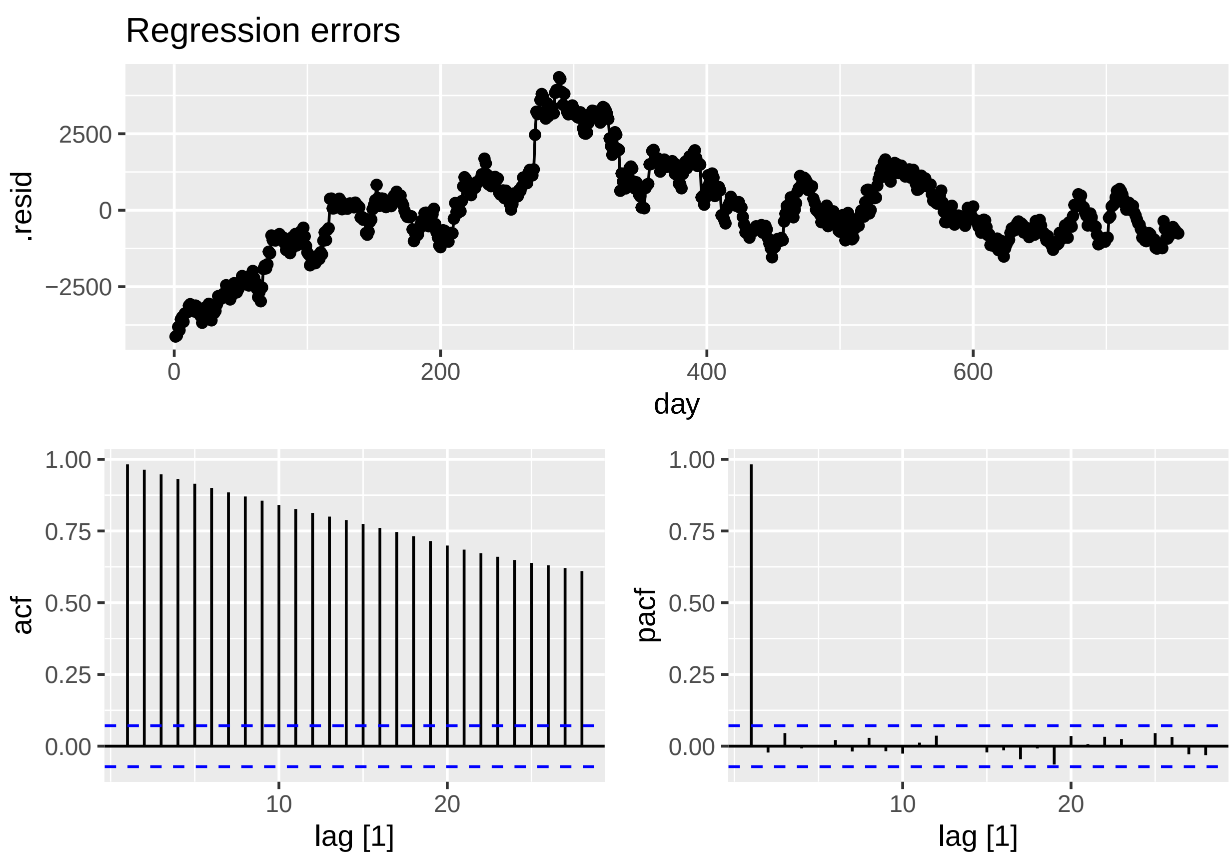
- best fitted is ARIMA(0,1,0) errors. This suggests residuals/unexplained variability in the our Dow Jones index follows a random walk. This reflect back to our actual observation.

residual check (compare with static and dynamic regression)

```
# plot residuals for dynamic regression
residuals(fit_lr_sarima, type='innovation') %>%
gg_tsdisplay(.resid, plot_type = 'partial') +
labs(title = "ARIMA errors")
```



```r
# plot residuals for static regression
residuals(fit_cons, type='innovation') %>%
gg_tsdisplay(.resid, plot_type = 'partial') +
labs(title = "Regression errors")
```



Residuals : In ARIMA errors, residuals are bounded around 0 in overall, which captures patterns in our index well.

ACF : In dynamic, each autocorrelation is close to zero and no 1 or more large spikes outside of confidence interval, thus series is a white noise. This suggests DJI is influenced by numerous unpredictable factors. In static regression, we see small lags are large, positive, has geometric, which imply a trend. This means our regression were only able to capture patterns in the big picture.

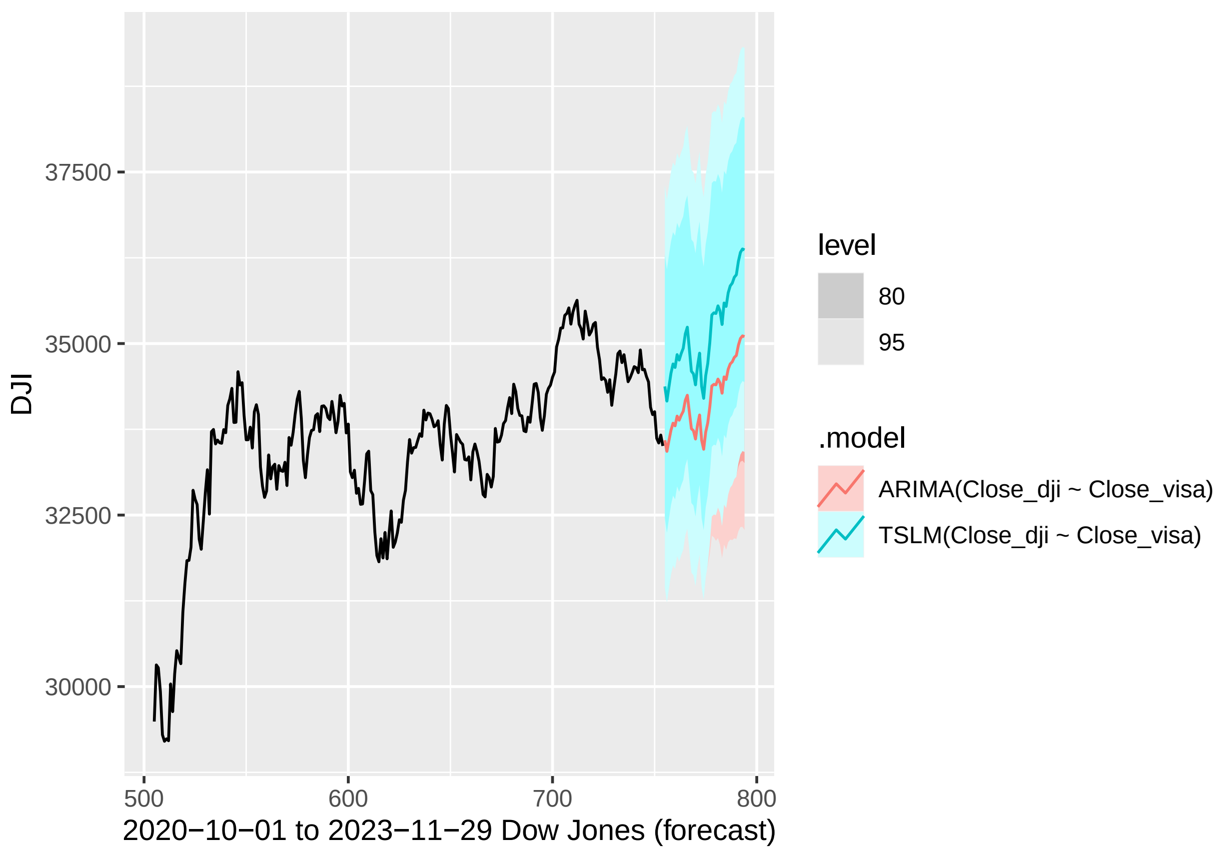
PACF : In Dynamic, its the same as ACF, suggests series is white noise. In static, PACF has large lag 1 spike, which indicates 1 day back influences the model very large.

From these inspect, dynamic model has captures adequate autocorrelations for DJI than static.

forecast and plot

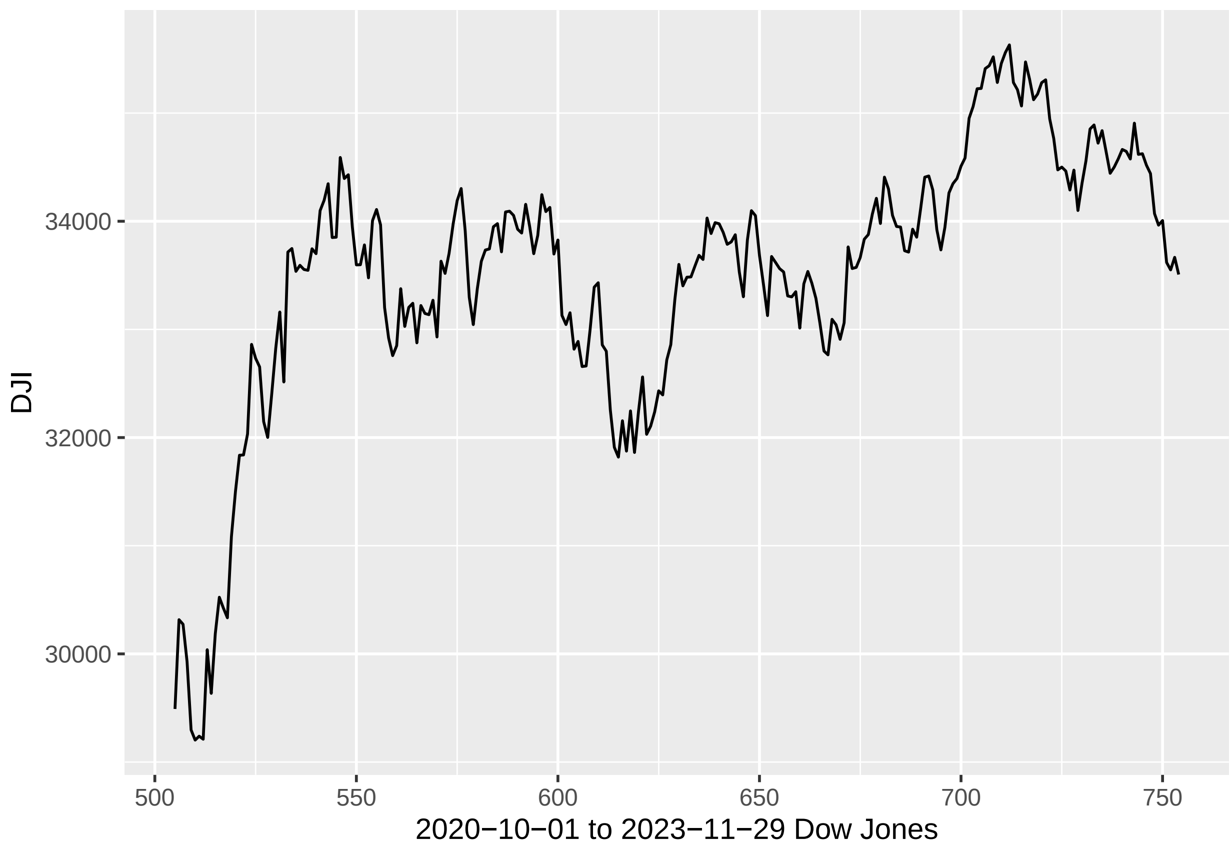
```r
# shorten the dataset to capture forecast plot better
data_for_plot <- data |>
  filter(between(Date, as.Date("2022-10-01"), as.Date("2023-09-30")))

# fit 2 models on train set
fit_2model <- train_dow_jones %>%
```

```r
  model(
    ARIMA(Close_dji ~ Close_visa),
    TSLM(Close_dji ~ Close_visa)
    )
# forecast 2 models on test set
forecast_2model <- forecast(fit_2model,test_dow_jones)



# plot forecast on most recent period
forecast_2model |>
  autoplot (data_for_plot) +
  xlab("2020-10-01 to 2023-11-29 Dow Jones (forecast)") +
  ylab("DJI")
```




```r
# plot original data
autoplot (data_for_plot) +
  xlab("2020-10-01 to 2023-11-29 Dow Jones") +
  ylab("DJI")
```

```
## Plot variable not specified, automatically selected `.vars = Close_dji`
```



accuracy

```
# compute the accuracy for 2 models
accuracy(forecast_2model,data)
```

```
## # A tibble: 2 x 10
##   .model                    .type      ME  RMSE   MAE    MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                     <chr>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Close_dji ~ Clo~ Test    -301.  468.  401. -0.911  1.20  1.68  1.47 0.879
## 2 TSLM(Close_dji ~ Clos~ Test   -1271. 1296. 1271. -3.77   3.77  5.33  4.08 0.802
```

cross valiadation

prepare cross-validation set.

cross_valid_set <- train_dow_jones |> # train set start from size 3 and increase the size of successive training sets by .step=1 stretch_tsibble(.init = 300, .step = 4) |> relocate(Date,day)

fit 2 models on cross validation for reliability

result <- cross_valid_set |> model(TSLM(Close_dji ~ Close_visa)) |> forecast(h = 1) |> accuracy()