cross_validate

Koki Yamanaka

2023-12-02

Imports

```
library(dplyr) # includes %>%
```

```
##
## 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(tsibble) # as_tsibble
```

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

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

```
library(ggplot2) # autoplot
library(fable) # model()
```

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

```
library(forecast) # na.interp
```

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

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

```
##
## Attaching package: 'forecast'

## The following object is masked from 'package:fabletools':
##
##     accuracy
```

```
library(tidyr) # pivot_longer
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:tsibble':
##
##     interval

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

```
library(feasts)
library(tsibbledata)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

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

##
## ############################ 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: 'xts'

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

## Loading required package: TTR
```

```
library(corrplot)
```

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

## corrplot 0.92 loaded
```

Data prepare

```
# Download data from yahoo Finance!
start.date = '2020-10-01' # starting date of stock
end.date = '2023-11-28' # ending date of stock

# target variable
getSymbols("^DJI", src = "yahoo", from = start.date, to = end.date,auto.assign = TRUE)
```

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

```
# predictors
getSymbols("^FTSE", src = "yahoo", from = start.date, to = end.date, auto.assign = TRUE)
```

```
## Warning: ^FTSE contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "FTSE"
```

```
getSymbols("^N225", src = "yahoo", from = start.date, to = end.date,auto.assign = TRUE)
```

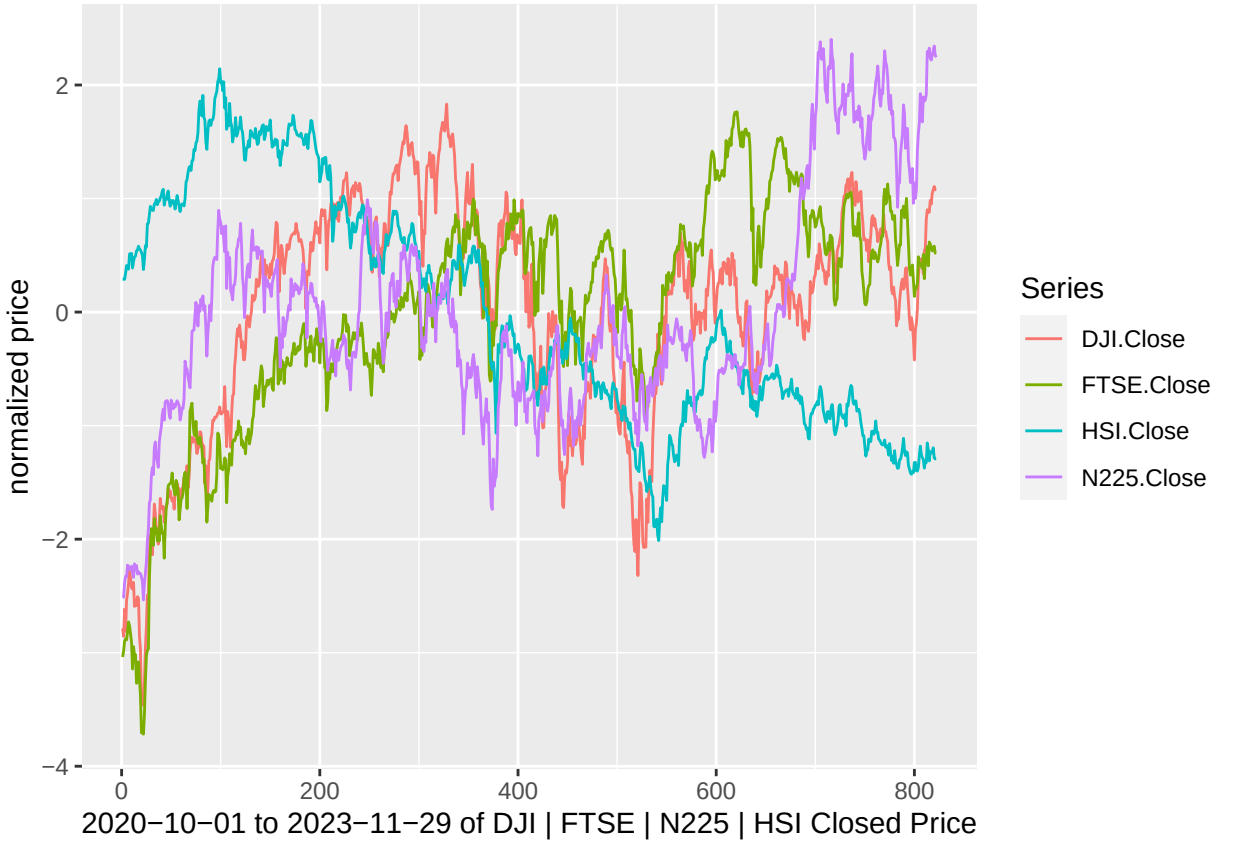
```
## [1] "N225"
```

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

```
## [1] "HSI"
```

```r
# store dji and visa close price
stocks <- merge(DJI = DJI[, "DJI.Close"],
                FTSE = FTSE[, "FTSE.Close"],
                N225 = N225[, "N225.Close"],
                HSI = HSI[, "HSI.Close"])

#  fill in those missing values with a constant value based on the nearest non-missing
stocks <- na.approx(stocks, rule = 2)

## normalize data ## ---- for plotting purpose

# normalize all stock price using scale() : Standard scaler
normalized_stocks <- as.data.frame(scale(stocks))
normalized_stocks <- zoo::fortify.zoo(normalized_stocks)
normalized_stocks <- normalized_stocks %>% rename(c("Date" = "Index"))
normalized_stocks <- as_tsibble(normalized_stocks, index = Date)
# Re-index based on trading days (as there are some missing days)
normalized_stocks <- normalized_stocks |>
  mutate(day = row_number()) |>
  update_tsibble(index = day, regular = TRUE)

## tsibble data   ## ---- for forecast purpose

# Create date variable
stocks <- zoo::fortify.zoo(stocks)
# Rename date variable
stocks <- stocks %>% rename(c("Date" = "Index"))
# Create tissble object
stocks <- as_tsibble(stocks, index = Date)
stocks <- stocks |>
  mutate(day = row_number()) |>
  update_tsibble(index = day, regular = TRUE)
```

```r
# Re-index based on trading days (as there are some missing days)
normalized_stocks <- normalized_stocks |>
  mutate(day_row_num = row_number()) |>
  update_tsibble(index = day, regular = TRUE)

# plot all stocks
normalized_stocks |>
  pivot_longer(c(DJI.Close, FTSE.Close, N225.Close, HSI.Close), names_to="Series") |>
  autoplot(value) +
  labs(y = "normalized price") +
  labs(x = "2020-10-01 to 2023-11-29 of DJI | FTSE | N225 | HSI Closed Price")
```

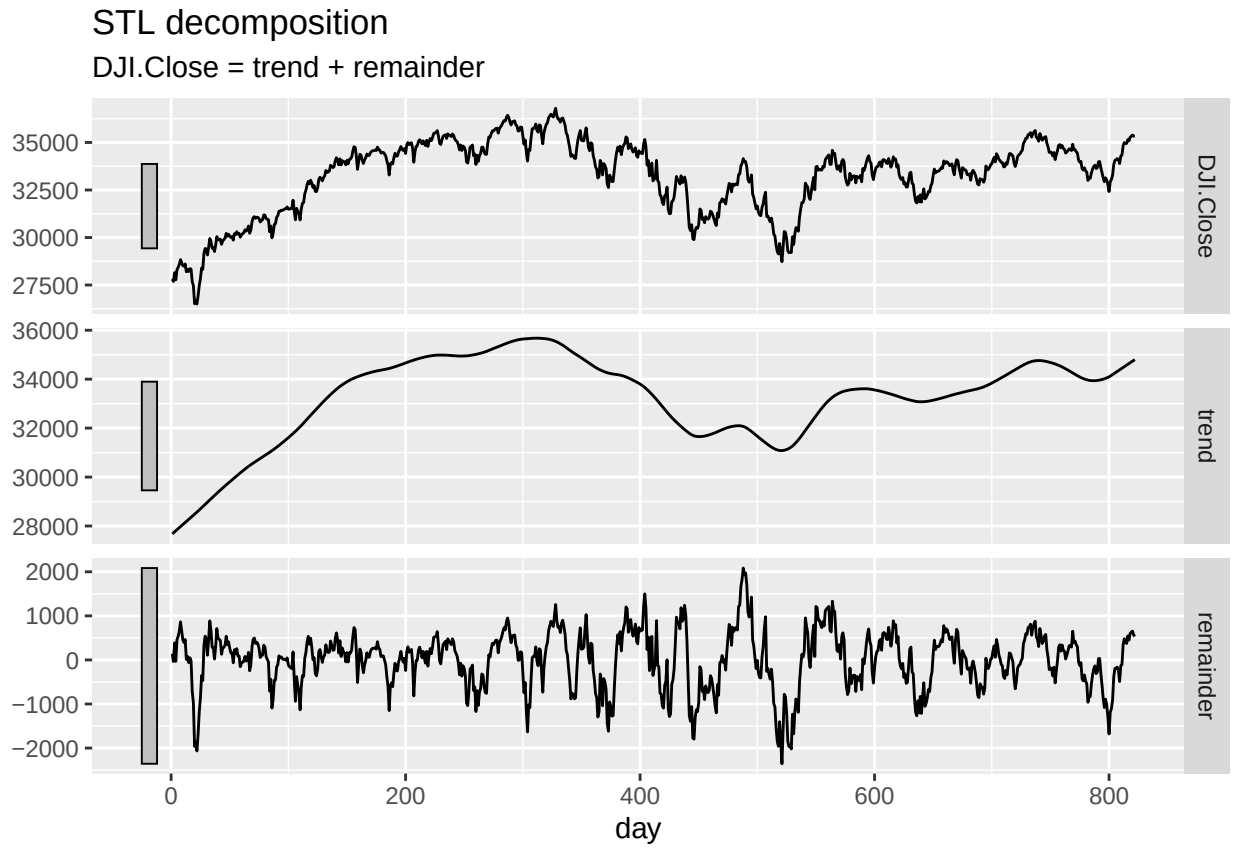



```
  theme(plot.caption = element_text(hjust = 0.5))
```

```
## List of 1
##  $ plot.caption:List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0.5
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi FALSE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
```

scale(): https://www.r-bloggers.com/2021/12/how-to-use-the-scale-function-in-r/. ## Seasonal-Trend decomposition

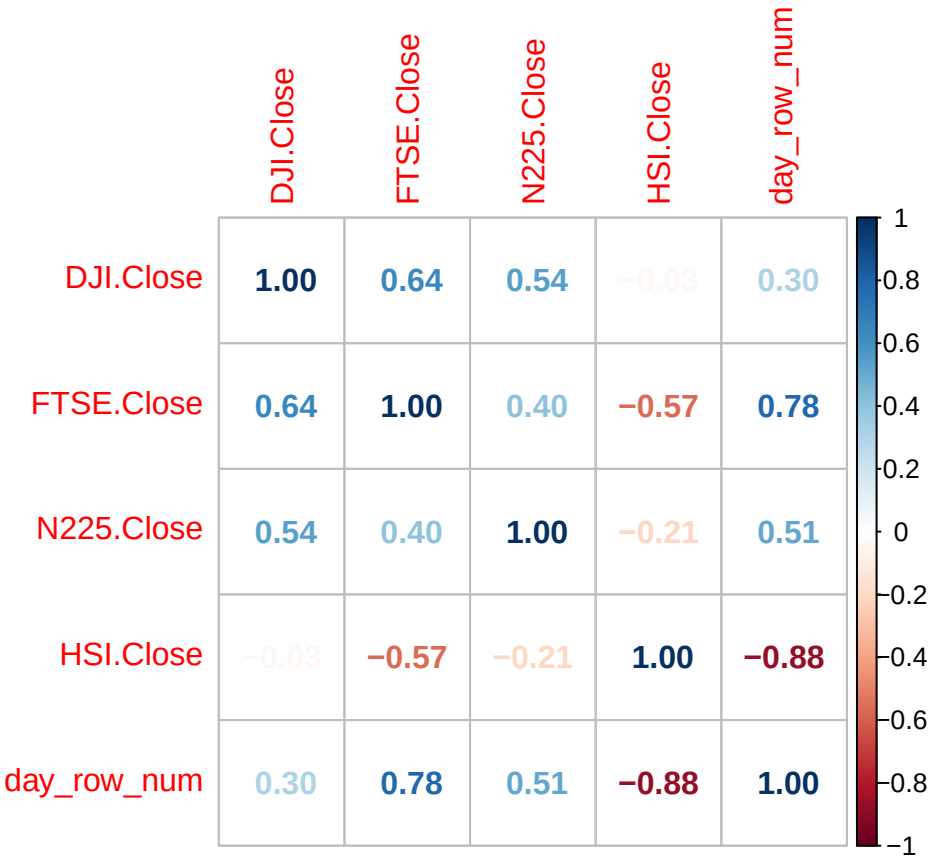
```r
# decompose the series
dcmp <- stocks %>%
  model(stl = STL(DJI.Close))

# plot the decomposition
components(dcmp)  %>% autoplot()
```



We see there is a strong trend.

Correlation analysis

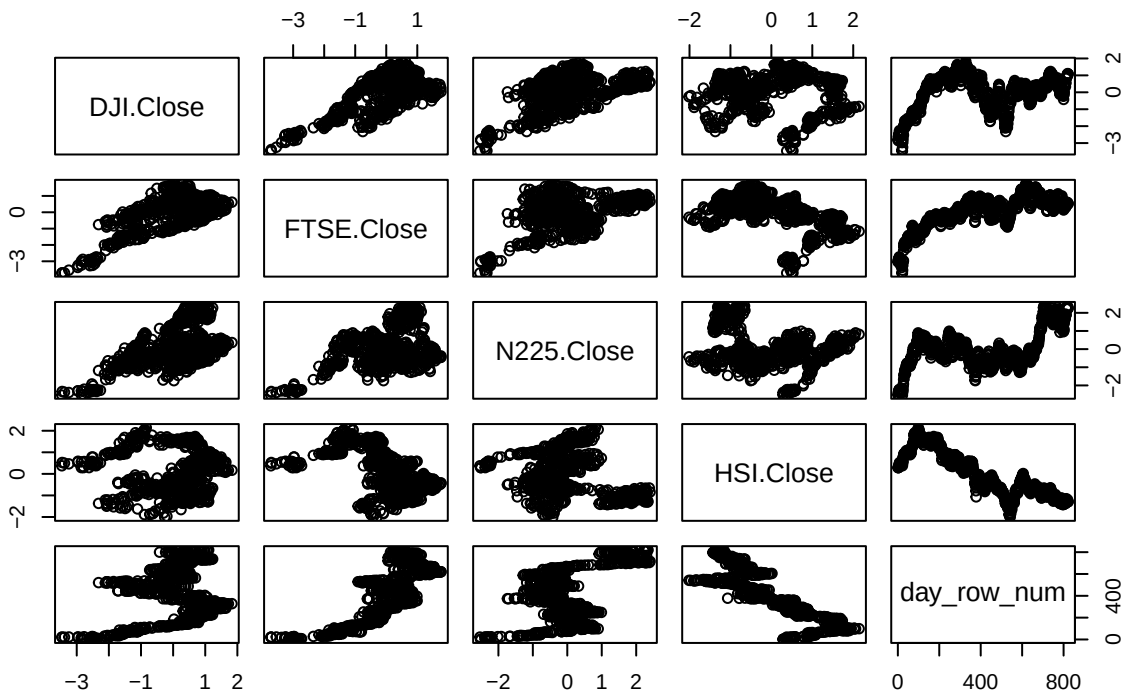
```r
# Convert tsibble to a data frame
normalized_stocks_df <- as.data.frame(normalized_stocks)

# remove date and day columns from data frame
columns_to_remove <- c("Date", "day") # specify
normalized_stocks_df <- normalized_stocks_df[, !(names(normalized_stocks) %in%columns_to_remove)]

# Compute correlation matrix
cor_mat_stocks <- cor(normalized_stocks_df, method = "pearson")

# plot correlation matrix
corrplot(cor_mat_stocks, method = "number")
```



We see DJI has correlation with by FTSE and N225. HSI has less as this reflects HSI series has an opposite direction in to our normalized price plot for 4 index.

```
# Create scatterplot matrix
pairs(coredata(normalized_stocks_df), main = "Figure 1.2 : Normalized Stocks Pairs Plot")
```

Figure 1.2 : Normalized Stocks Pairs Plot



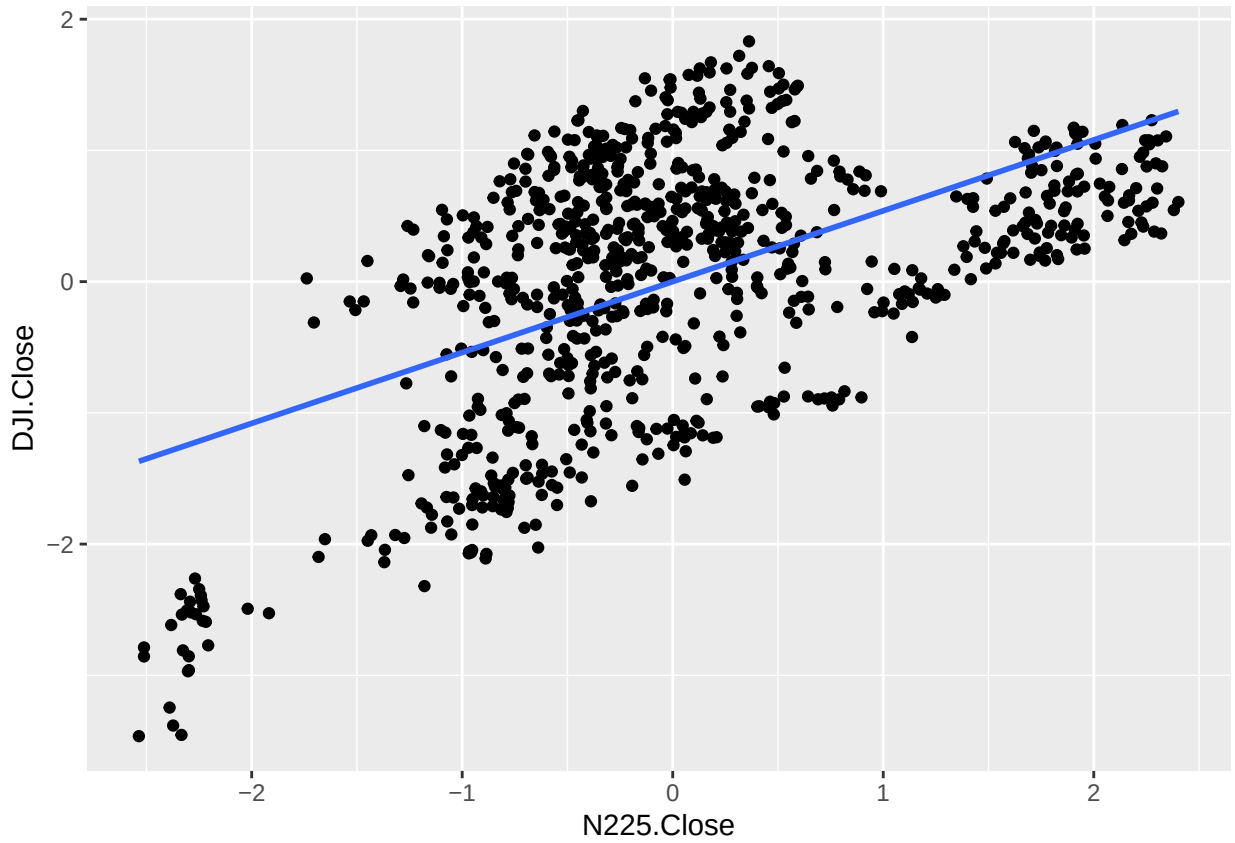
we FTSE and N225 has a positive correlation. Let's scatter plot for FTSE and N225.

```
# scatter plot FTSE.Close vs DJI.Close
normalized_stocks_df %>%
  ggplot(aes(x = FTSE.Close, y = DJI.Close)) +
  labs(y = "DJI.Close",
  x = "FTSE.Close") +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
```

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


```
# scatter plot N225.Close vs DJI.Close
normalized_stocks_df %>%
  ggplot(aes(x = N225.Close, y = DJI.Close)) +
  labs(y = "DJI.Close",
  x = "N225.Close") +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
```

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



In N225, we have datapoints aggregrated in the middle. This spread of data points is not uniform across the range of values, it suggests heteroscedasticity.

train test split

we will fit using non-normalized values

```
# filter train set
train_dow_jones <- stocks |>
  filter(between(Date, as.Date("2020-10-01"), as.Date("2023-09-30")))
# filter test set
test_dow_jones <- stocks  |>
  filter(between(Date, as.Date("2023-10-01"), as.Date("2023-11-29")))
```

we expect tslm will not perform well from our previous experiment, thus we skip fitting.

forecast and plot

```
# Fit a dynamic regression model for
# FTSE
fit_arima_FTSE <- train_dow_jones %>%
  model(ARIMA(DJI.Close ~ FTSE.Close))
# N225
```

```
fit_arima_N225  <- train_dow_jones %>%
  model(ARIMA(DJI.Close ~ N225.Close))

fit_arima_both <- train_dow_jones %>%
  model(ARIMA(DJI.Close ~ FTSE.Close + N225.Close ))

report(fit_arima_FTSE)
```

```
## Series: DJI.Close
## Model: LM w/ ARIMA(3,1,3) errors
##
## Coefficients:
##          ar1      ar2     ar3     ma1     ma2      ma3  FTSE.Close
##       -0.7136  -0.3467  0.2855  0.5964  0.1977  -0.4400       2.398
## s.e.   0.1950   0.2647  0.2234  0.1831  0.2455   0.2173       0.161
##
## sigma^2 estimated as 75001:  log likelihood=-5474.11
## AIC=10964.22   AICc=10964.41   BIC=11001.49
```

```
report(fit_arima_N225)
```

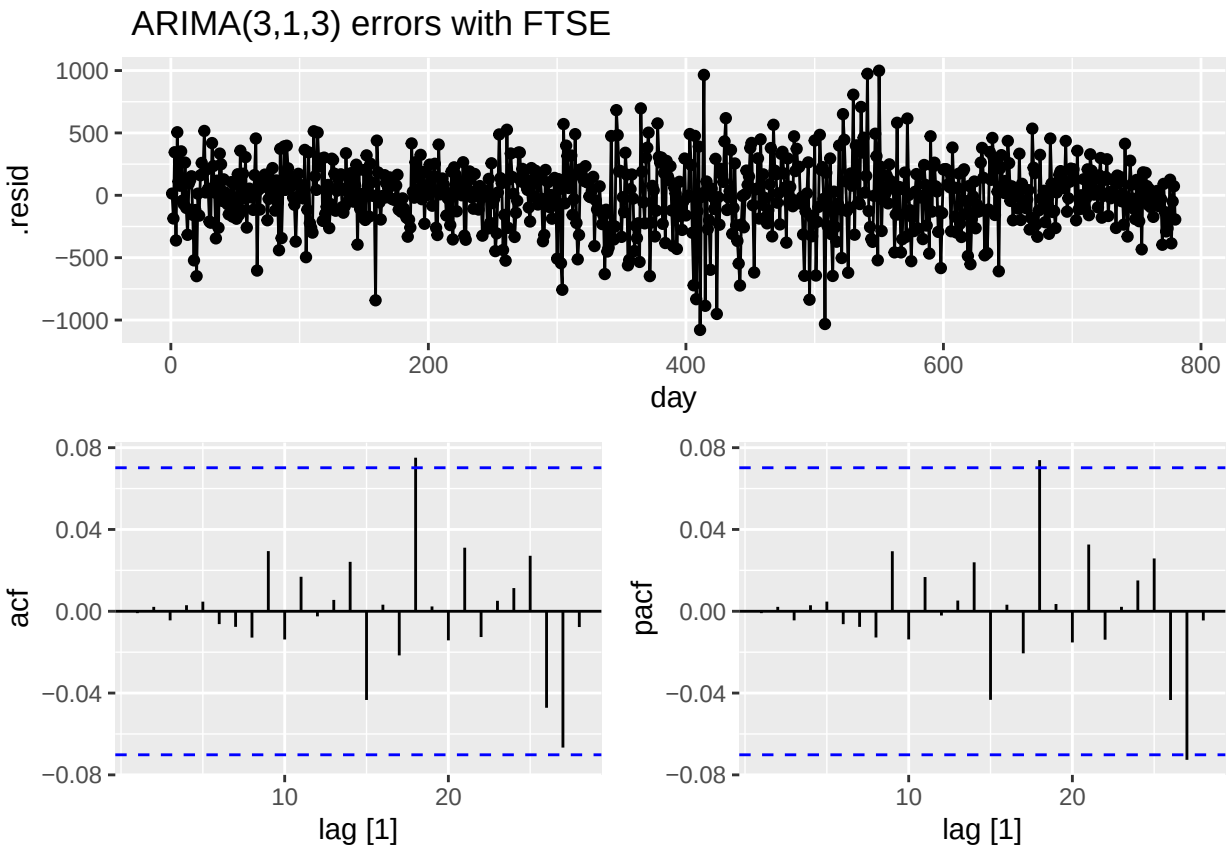
```
## Series: DJI.Close
## Model: LM w/ ARIMA(0,1,1) errors
##
## Coefficients:
##          ma1  N225.Close
##       -0.1076      0.2265
## s.e.   0.0419      0.0406
##
## sigma^2 estimated as 92358:  log likelihood=-5557.68
## AIC=11121.35   AICc=11121.39   BIC=11135.33
```

```
report(fit_arima_both)
```

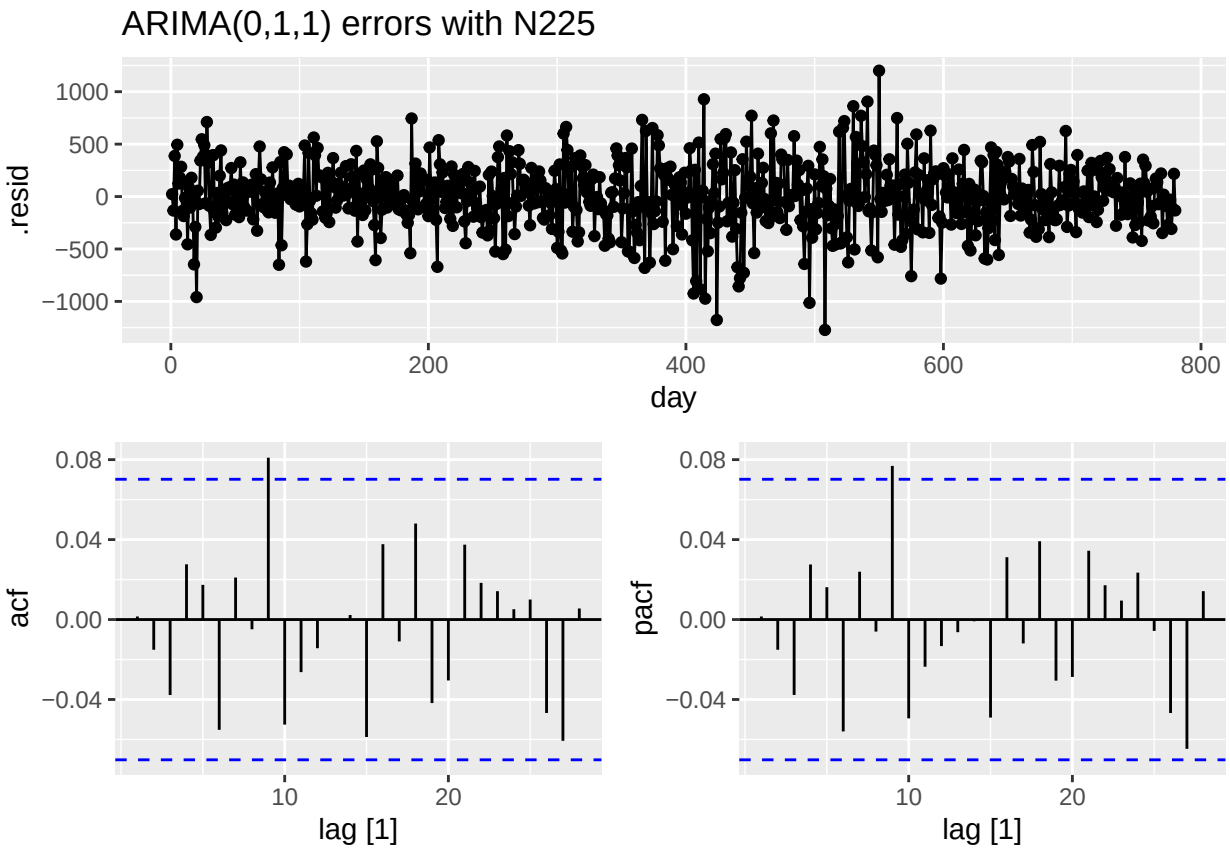
```
## Series: DJI.Close
## Model: LM w/ ARIMA(0,1,3) errors
##
## Coefficients:
##          ma1      ma2      ma3  FTSE.Close  N225.Close
##       -0.1615  -0.0628  -0.0558      2.2817      0.1016
## s.e.   0.0400   0.0351   0.0365      0.1647      0.0356
##
## sigma^2 estimated as 74318:  log likelihood=-5471.55
## AIC=10955.09   AICc=10955.2   BIC=10983.04
```

residuals check before forecast

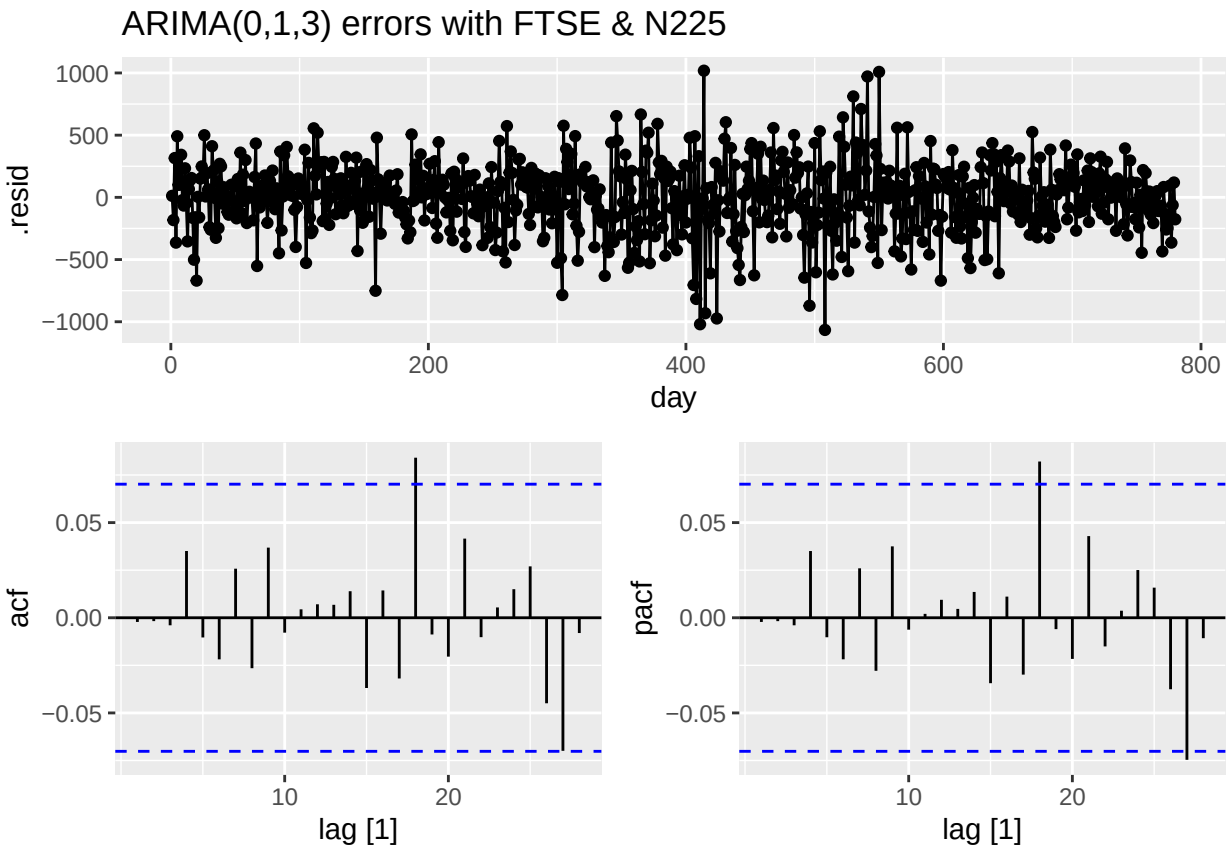
```
# plot residuals for dynamic regression
residuals(fit_arima_FTSE, type='innovation') %>%
gg_tsdisplay(.resid, plot_type = 'partial') +
labs(title = " ARIMA(3,1,3) errors with FTSE")
```




```
# plot residuals for dynamic regression
residuals(fit_arima_N225, type='innovation') %>%
gg_tsdisplay(.resid, plot_type = 'partial') +
labs(title = "ARIMA(0,1,1) errors with N225 ")
```




```
# plot residuals for dynamic regression with 2 features
residuals(fit_arima_both, type='innovation') %>%
gg_tsdisplay(.resid, plot_type = 'partial') +
labs(title = "ARIMA(0,1,3) errors with FTSE & N225")
```




```
# fit_arima_FTSE |> gg_residuals()
```

1. All model does't have any significant autocorrelation in the residuals, which means the prediction intervals may provide accurate coverage.
2. all models ACF's are bounded within 0, which suggest series is white noise. This can imply DJI itself in short term, it doesn't exhibit any specific pattern. Thus, it suggests ARIMA(0,1,1) can perform well.
3. Also, N225 residuals are more bounded close to 0 compared to others.
4. ACF and PACF are identical for all models, thus series has no unique correlation structure.

forecast

```
detach("package:forecast", unload=TRUE)

# fit 2 models on train set
fit_3model <- train_dow_jones %>%
  model(
    ARIMA(DJI.Close ~ FTSE.Close),
    ARIMA(DJI.Close ~ N225.Close),
    ARIMA(DJI.Close ~ FTSE.Close + N225.Close)
    )
```

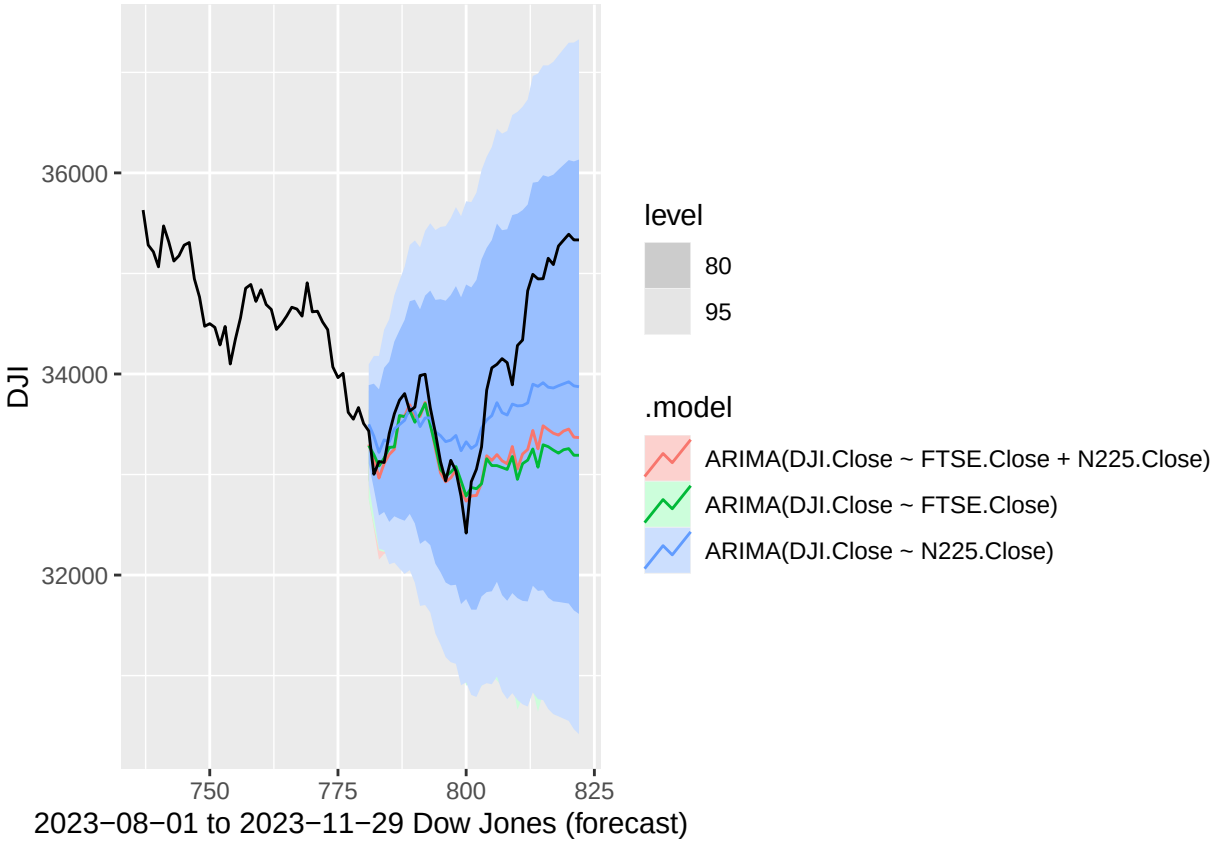
```
##---- accuracy for train ---- ##
fit_3model |> accuracy()
```

```
## # A tibble: 3 x 10
##   .model               .type    ME  RMSE   MAE     MPE  MAPE  MASE RMSSE     ACF1
##   <chr>                <chr> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 ARIMA(DJI.Close ~ ~ Trai~  2.68  272.  206. 0.00634 0.625 0.892 0.881 -9.20e-4
## 2 ARIMA(DJI.Close ~ ~ Trai~  5.35  303.  228. 0.0132  0.693 0.988 0.980  1.52e-3
## 3 ARIMA(DJI.Close ~ ~ Trai~  1.61  272.  205. 0.00292 0.622 0.888 0.878 -2.20e-3
```

```
##---- accuracy for test ----- ##
# forecast 3 models on test set
forecast_3model <- forecast(fit_3model,test_dow_jones)
# get accuracy for test set
accuracy(forecast_3model,stocks)
```

```
## # A tibble: 3 x 10
##   .model                   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(DJI.Close ~ FTSE.~ Test   704. 1006.  733.  2.03  2.11  3.18  3.25 0.909
## 2 ARIMA(DJI.Close ~ FTSE.~ Test   748. 1099.  789.  2.15  2.27  3.42  3.55 0.923
## 3 ARIMA(DJI.Close ~ N225.~ Test   366.  731.  563.  1.03  1.63  2.44  2.36 0.889
```

```
# filter close periods to see forecast clearer
stocks_plot_purpose <- stocks |>
  filter(between(Date, as.Date("2023-08-01"), as.Date("2023-11-29")))

# plot forecast on most recent period
forecast_3model |>
  autoplot (stocks_plot_purpose) +
  xlab("2023-08-01 to 2023-11-29 Dow Jones (forecast)") +
  ylab("DJI")
```



Conclusion

we know differencing was to capture the dynamics more.

ARIMA(3,1,3) errors with FTSE : 1. FTSE was not a good predictor, but it didn't have enough correlation with DJI. 2. AR of lag 3 was not effective, thus DJI can be random walk.

ARIMA(0,1,1) errors with N225 : 1. N225 is a good predictor for DJI. 2. fixing the error for model is best done by only considering lag 1. large MA lag value doesn't work.

ARIMA(0,1,3) errors with FTSE & N225 1. FTSE was not a good predictor, N225 solely itself as predictor was effective. 2. MA of lag 3 was not effective

Conclusion 1. Effective predictor :DJI seems to be influenced by N225 more than FTSE. 2. Short lags are effective : Short term movements in DJI series are best captured by immediate past, high order lag don't significantly improve model.