ISA 444: Business Forecasting

06: The Naive Forecast, Measures of Forecast Accuracy and the Prediction Interval

Fadel M. Megahed, PhD

Endres Associate Professor
Farmer School of Business
Miami University

@FadelMegahed
fmegahed
fmegahed@miamioh.edu
? Automated Scheduler for Office Hours

Spring 2023

Quick Refresher from Last Class

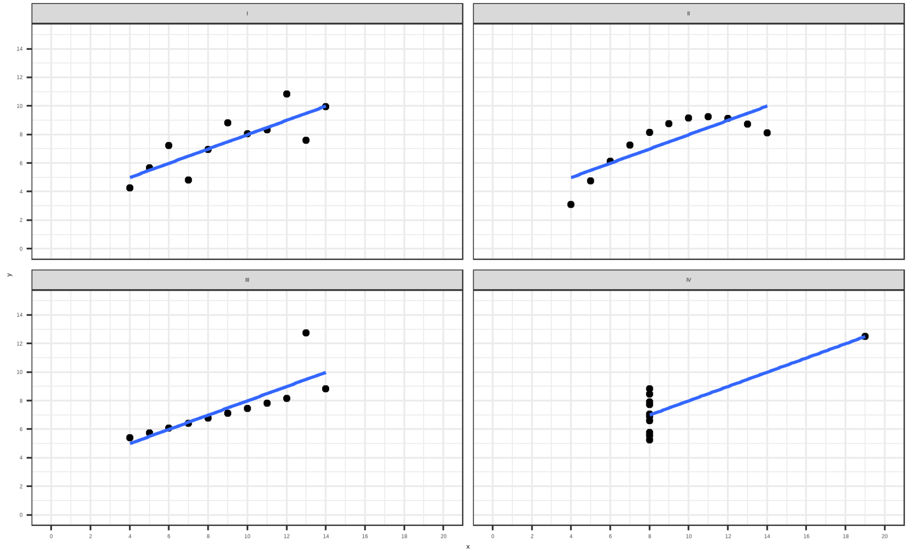
- ✅ Use numerical summaries to describe a time series.
- ✅ Explain what do we mean by correlation.
- ✅ Apply transformations to a time series.

Recap: Viz + Numerical Summary = Big Picture

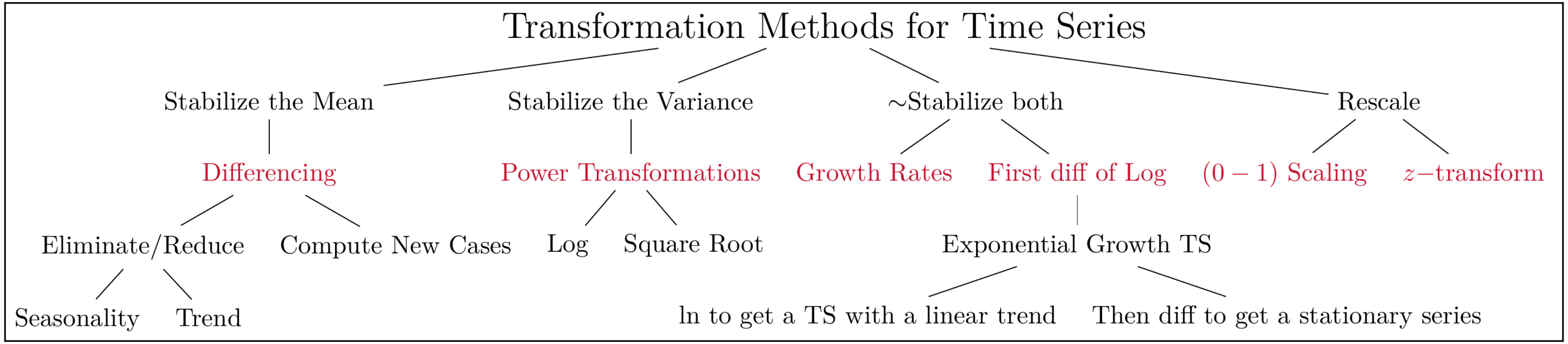
set	x.mean	x.sd	y.mean	y.sd	corr
I	9	3.32	7.5	2.03	0.82
II	9	3.32	7.5	2.03	0.82
III	9	3.32	7.5	2.03	0.82
IV	9	3.32	7.5	2.03	0.82

Showing 1 to 4 of 4 entries

Previous 1 Next



Recap: Guidelines for Transforming TS Data



A classification of common transformation approaches for time series data

A Quick Recap of Assignment 04

Question 01

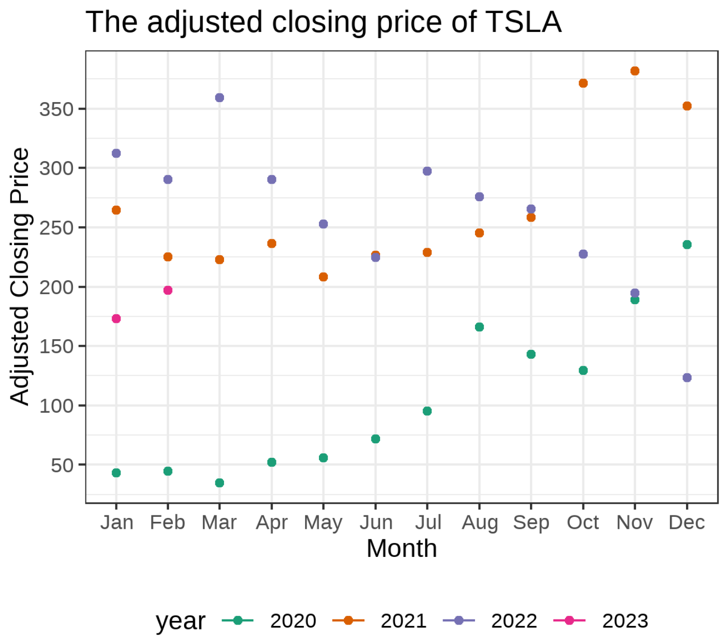
```r
tsla = tidyquant::tq_get(
  x = 'tsla', from = '2020-01-01', to = Sys.Date(),
  periodicity = 'monthly'
  ) |>
  dplyr::mutate(
    year = lubridate::year(date) |> as.factor(),
    month = lubridate::month(date, label = T)
    )

tsla
```

```
## # A tibble: 38 × 10
##    symbol date        open  high   low
##    <chr>  <date>     <dbl> <dbl> <dbl>
##  1 tsla   2020-01-01  28.3  43.5  28.1
##  2 tsla   2020-02-01  44.9  64.6  40.8
##  3 tsla   2020-03-01  47.4  53.8  23.4
##  4 tsla   2020-04-01  33.6  58.0  29.8
##  5 tsla   2020-05-01  50.3  56.2  45.5
##  6 tsla   2020-06-01  57.2  72.5  56.9
##  7 tsla   2020-07-01  72.2 120.   72.0
##  8 tsla   2020-08-01  96.6 167.   91
##  9 tsla   2020-09-01 167.  167.  110.
## 10 tsla   2020-10-01 147.  155.  126.
## # … with 28 more rows
```

A Quick Recap of Assignment 04

Question 01 ❌

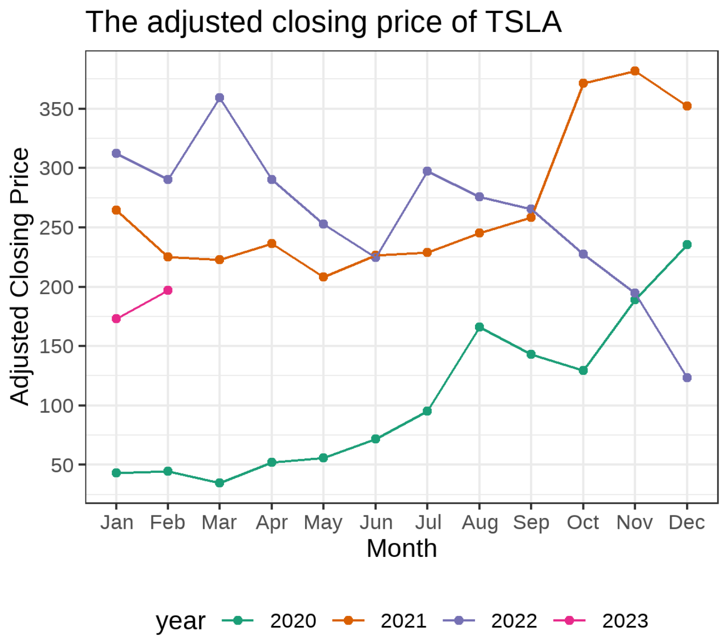
```r
tsla |>
  ggplot2::ggplot(
    ggplot2::aes(x = month, y = adjusted, color = year)
      ) + # aesthetics
  ggplot2::geom_point() + # adding the points
  ggplot2::geom_line() + # adding a line for each year
  ggplot2::scale_y_continuous(breaks = scales::pretty_breaks(n = 6))
  ggplot2::scale_color_brewer(palette = 'Dark2') + # color blind frie
  ggplot2::labs(
    x = 'Month',
    y = 'Adjusted Closing Price',
    title = 'The adjusted closing price of TSLA'
  ) +
  ggplot2::theme_bw() + # black and white theme
  ggplot2::theme(legend.position = 'bottom') # put legend below chart
```



A Quick Recap of Assignment 04

Question 01 ✅

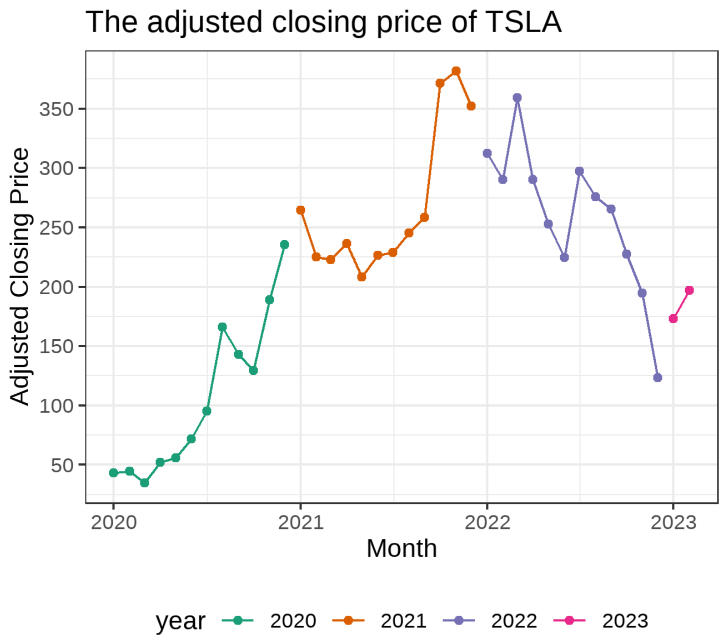
```
tsla |>
  ggplot2::ggplot(
    ggplot2::aes(x = month, y = adjusted, color = year, group = year)
      ) + # aesthetics
  ggplot2::geom_point() + # adding the points
  ggplot2::geom_line() + # adding a line for each year
  ggplot2::scale_y_continuous(breaks = scales::pretty_breaks(n = 6))
  ggplot2::scale_color_brewer(palette = 'Dark2') + # color blind frie
  ggplot2::labs(
    x = 'Month',
    y = 'Adjusted Closing Price',
    title = 'The adjusted closing price of TSLA'
  ) +
  ggplot2::theme_bw() + # black and white theme
  ggplot2::theme(legend.position = 'bottom') # put legend below chart
```



A Quick Recap of Assignment 04

Question 02 ✅

```
tsla |>
  ggplot2::ggplot(
    ggplot2::aes(x = date, y = adjusted, color = year, group = year)
      ) + # aesthetics
  ggplot2::geom_point() + # adding the points
  ggplot2::geom_line() + # adding a line for each year
  ggplot2::scale_y_continuous(breaks = scales::pretty_breaks(n = 6))
  ggplot2::scale_color_brewer(palette = 'Dark2') + # color blind frie
  ggplot2::labs(
    x = 'Month',
    y = 'Adjusted Closing Price',
    title = 'The adjusted closing price of TSLA'
  ) +
  ggplot2::theme_bw() + # black and white theme
  ggplot2::theme(legend.position = 'bottom') # put legend below chart
```



A Quick Recap of Assignment 04

Question 03 ✅

This will obviously depend on your stock. In my case, there is no evidence for a consistent seasonal pattern.

A Quick Recap of Assignment 04

Question 04 ✅

In question 4, the mean makes sense to compute. The standard deviation does not since you only have one observation per month.

```
tsla_summary =
  tsla |>
  dplyr::group_by(symbol, year, month) |>
  dplyr::summarise(
    adj_avg = mean(adjusted),
    adj_sd = sd(adjusted)
  )

print(tsla_summary, n = 10)
```

```
## # A tibble: 38 × 5
## # Groups:   symbol, year [4]
##    symbol year  month adj_avg adj_sd
##    <chr>  <fct> <ord>   <dbl>  <dbl>
##  1 tsla   2020  Jan      43.4     NA
##  2 tsla   2020  Feb      44.5     NA
##  3 tsla   2020  Mar      34.9     NA
##  4 tsla   2020  Apr      52.1     NA
##  5 tsla   2020  May      55.7     NA
##  6 tsla   2020  Jun      72.0     NA
##  7 tsla   2020  Jul      95.4     NA
##  8 tsla   2020  Aug     166.      NA
##  9 tsla   2020  Sep     143.      NA
## 10 tsla   2020  Oct     129.      NA
## # … with 28 more rows
```

A Quick Recap of Assignment 04

Question 05 ✅

```r
nasdaq =
  tidyquant::tq_get(x = '^IXIC', from = '2020-01-01', to = Sys.Date()
  periodicity = 'monthly') |>
  dplyr::mutate(
    year = lubridate::year(date) |> as.factor(),
    month = lubridate::month(date, label = T)
    )

# putting them next to each other (note the adjusted column name char
both_stocks = dplyr::left_join(
  x = tsla |> dplyr::select(date, year, month, adjusted),
  y = nasdaq |> dplyr::select(date, adjusted),
  by = 'date'
)

both_stocks
```

```
## # A tibble: 38 × 5
##    date       year  month adjusted.x a
##    <date>     <fct> <ord>      <dbl>
##  1 2020-01-01 2020  Jan         43.4
##  2 2020-02-01 2020  Feb         44.5
##  3 2020-03-01 2020  Mar         34.9
##  4 2020-04-01 2020  Apr         52.1
##  5 2020-05-01 2020  May         55.7
##  6 2020-06-01 2020  Jun         72.0
##  7 2020-07-01 2020  Jul         95.4
##  8 2020-08-01 2020  Aug        166.
##  9 2020-09-01 2020  Sep        143.
## 10 2020-10-01 2020  Oct        129.
## # … with 28 more rows
```

A Quick Recap of Assignment 04

Question 05 ✅

```r
nasdaq =
  tidyquant::tq_get(x = '^IXIC', from = '2020-01-01', to = Sys.Date()
  periodicity = 'monthly') |>
  dplyr::mutate(
    year = lubridate::year(date) |> as.factor(),
    month = lubridate::month(date, label = T)
    )

# putting them next to each other (note the adjusted column name char
both_stocks = dplyr::left_join(
  x = tsla |> dplyr::select(date, year, month, adjusted),
  y = nasdaq |> dplyr::select(date, adjusted),
  by = 'date'
)

both_stocks |>
  dplyr::group_by(year) |>
  dplyr::summarise(corr = cor(adjusted.x, adjusted.y))
```

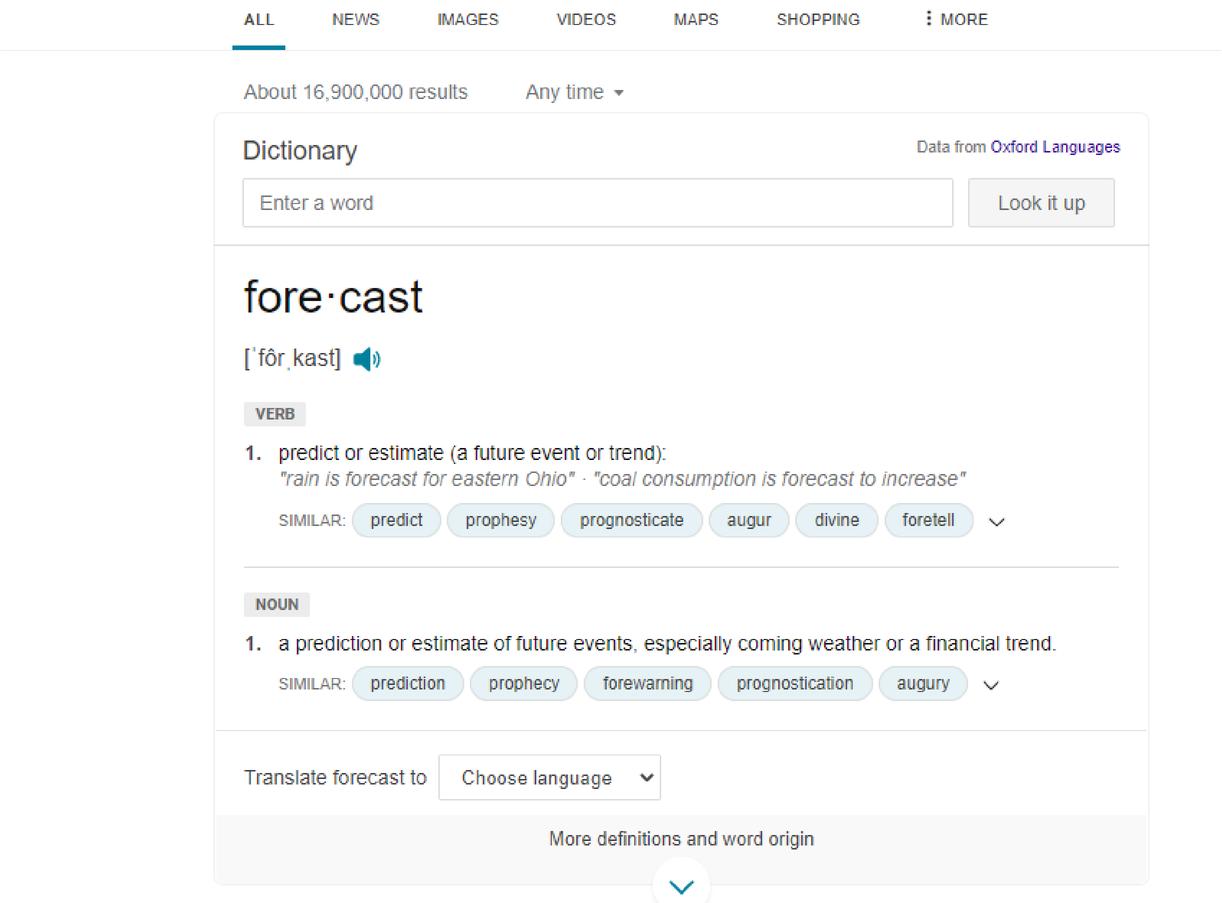
```
## # A tibble: 4 × 2
##   year   corr
##   <fct> <dbl>
## 1 2020  0.959
## 2 2021  0.721
## 3 2022  0.792
## 4 2023  1
```

Learning Objectives for Today's Class

- Compute the nonseasonal naive forecast.
- Apply and interpret measures of forecast accuracy.
- Interpret prediction intervals for a simple forecast.

The Naive Forecast

Recap: What is Forecasting?



A Naïve Forecast

- A naïve forecast for an observation, Y_t , is the observation prior, Y_{t-1}.
- For some types of time series (e.g. **Random Walks**), a naïve forecast is the **best possible forecast one can make**.
- For almost any time series, the naive forecast should be included as a **benchmark/baseline.** How much is your forecast better than the naïve forecast? Is it worth it?
- In the case of seasonal data, a naïve forecast could be the observation from the prior period.
    - For example, in the case of monthly data, the naïve forecast for the observation $Y_{Jan2024}$ could be $Y_{Jan2023}$. In this case, we would denote the frequency, m=12, and the naïve forecast for Y_t is the observation m periods prior, or Y_{t-m}.

Intuition and Code Behind the Naïve Forecast

```r
# printing a few columns and observations from TSLA

tsla |>
  # selecting the two columns of interest
  dplyr::select(date, adjusted) |>
  # printing the last 5 observations for demo
  dplyr::slice_tail(n = 5) |>
  # printing empty space under a naive_f column
  dplyr::mutate(naive_f = '---')
```

```
## # A tibble: 5 × 3
##   date       adjusted naive_f
##   <date>        <dbl> <chr>
## 1 2022-10-01     228. ---
## 2 2022-11-01     195. ---
## 3 2022-12-01     123. ---
## 4 2023-01-01     173. ---
## 5 2023-02-01     197. ---
```

Intuition and Code Behind the Naïve Forecast

Go to **www.menti.com/alqezoh8oh13**

I know how to compute the naive forecast in R

Mentimeter

Yes	No
0	0

Measures of Forecast Accuracy

- The measures of accuracy we will discuss all deal with the difference between the **actual observed value** (Y_t) and the **forecasted value** (F_t) for time t.
- In order to measure forecast accuracy, we assume we have m actual values available, thus we have $Y_{t+1}, Y_{t+2}, \ldots, Y_{t+m}$ and forecasts $F_{t+1}, F_{t+2}, \ldots, F_{t+m}$.
- This is important because we will be averaging the forecast errors over m.
- **Forecast Error:** $e_{t+i} = Y_{t+i} - F_{t+i}$.

Measures of "Average" Forecast Performance

- A positive average error measure indicates that: for your m forecasts, on average your **actuals** (Y_{t+i}) are **larger than** their corresponding **forecasted values** (F_{t+i}), i.e., you are underestimating.
- A negative average error measure indicates that you are overestimating on average.
- If your average error **(percent)** measure is ~0; you have an **unbiased** forecast.
    - An unbiased average measure on its own is meaningless; look at its variability.
- **Mean Error:**

$$ME = \frac{\sum_{i=1}^{m} e_{t+i}}{m}.$$

- **Mean Percentage Error:**

$$MPE = \frac{100}{m} \sum_{i=1}^{m} \frac{e_{t+i}}{Y_{t+i}}.$$

Measures of "Variability" in Forecast Performance

- **Absolute Forecast Error:**

$$|e_{t+i}| = |Y_{t+i} - F_{t+i}|.$$

- **Squared Forecast Error:**

$$(e_{t+i})^2 = (Y_{t+i} - F_{t+i})^2.$$

- **Mean Absolute Error:**

$$MAE = \frac{\sum_{i=1}^{m} |e_{t+i}|}{m}.$$

- **Root Mean Squared Error:**

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m} (e_{t+i})^2}{m}}.$$

Measures of "Relative" Forecast Performance

- **Mean Absolute Percentage Error:**

$$MAPE = \frac{100}{m} \sum_{i=1}^{m} \frac{|e_{t+i}|}{m}.$$

- **Relative Mean Absolute Error:**

$$RelMAE = \frac{\sum_{i=1}^{m} |e_{t+i}|}{\sum_{i=1}^{m} |Y_{t+i} - Y_{t+i-1}|}.$$

- **Thiel's U:**

$$U = \sqrt{\frac{\sum_{i=1}^{m} (e_{t+i})^2}{\sum_{i=1}^{m} (Y_{t+i} - Y_{t+i-1})^2}}.$$

Computing these Measures in R via the `forecast` 📦

```r
tsla_df =
  tsla |>
  # keeping relevant columns so df prints nicely
  dplyr::select( symbol, date, year, month, adjusted) |>
  # the naive forecast = Y_{t-1} = lag(Y_t, 1)
  dplyr::mutate( naive_f = dplyr::lag(adjusted, n =1) )

# printing the first 3 rows
tsla_df |> dplyr::slice_head(n = 3)
```

```
## # A tibble: 3 × 6
##   symbol date       year  month adjusted naive_f
##   <chr>  <date>     <fct> <ord>    <dbl>   <dbl>
## 1 tsla   2020-01-01 2020  Jan       43.4    NA
## 2 tsla   2020-02-01 2020  Feb       44.5    43.4
## 3 tsla   2020-03-01 2020  Mar       34.9    44.5
```

```r
# computing the accuracy metrics via the forecast pkg
forecast::accuracy(
  # we start at row 2 since first fct is NA
  object = tsla_df$naive_f[2:nrow(tsla_df)],
  x = tsla_df$adjusted[2:nrow(tsla_df)]
)
```

```
##                ME    RMSE      MAE      MPE     MAPE
## Test set 4.146991 40.55307 32.06263 1.861465 16.42575
```

Computing these Measures in R with dplyr

```r
tsla_df |>
  dplyr::mutate(
    error = adjusted - naive_f, # actual - forecast
    perc_error = error / adjusted # error/actual
  ) |>
  dplyr::group_by(symbol) |> # not needed but it would allow you to compute for multiple TS
  dplyr::summarise(
    # measures of "average" forecast performance
    me = mean( error, na.rm = T ),
    mpe = 100* mean( perc_error, na.rm = T ),
    # measures of "variability" in forecast performance
    mae = mean( abs(error), na.rm = T ),
    mape = 100* mean( abs(perc_error), na.rm = T),
    rmse = mean( error^2, na.rm = T ) |> sqrt()
  )
```

```
## # A tibble: 1 × 6
##   symbol    me   mpe   mae  mape  rmse
##   <chr>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 tsla    4.15  1.86  32.1  16.4  40.6
```

Some Practical Insights

Main Insight(s): (Edit below)

- The **naive forecast** is .... of the series; thus, the forecast error is the ...
- In general, if the $MAE \approx |ME|$, then we conclude that ... . If we were using a naive forecast in such a case, then we can also conclude that ...
- Irrespective of the forecasting method, the MPE and MAPE are useful/valid if and only if ....

The Prediction Interval

Prediction Intervals $\neq$ Confidence Intervals

- Prediction intervals and confidence intervals are **not** the same.
- A **prediction interval** is an interval associated with a **random variable yet to be observed, with a specified probability of the random variable lying within the interval**.
  - For example, I might give an 80% interval for the forecast of GDP in 2024. The actual GDP in 2024 should lie within the interval with probability 0.8.
- A **confidence interval** is an interval associated with **a parameter** (e.g., the mean of a random variable) ... The parameter is assumed to be non-random but unknown, and the confidence interval is computed from data. Because the data are random, the interval is random ... That is, with a large number of repeated samples, 95% of the intervals would contain the true parameter if you built a 95%.
- **Prediction intervals** are wider than confidence intervals since it includes the variance of ϵ (the error in our predictions).

Adapted from Rob Hyndman: The difference between prediction intervals and confidence intervals

Point vs Interval Forecasts

- **Point Forecasts:** future observations for which we report a single forecast observation.
- **Interval Forecast:** a range of values that are reported to forecast an outcome.

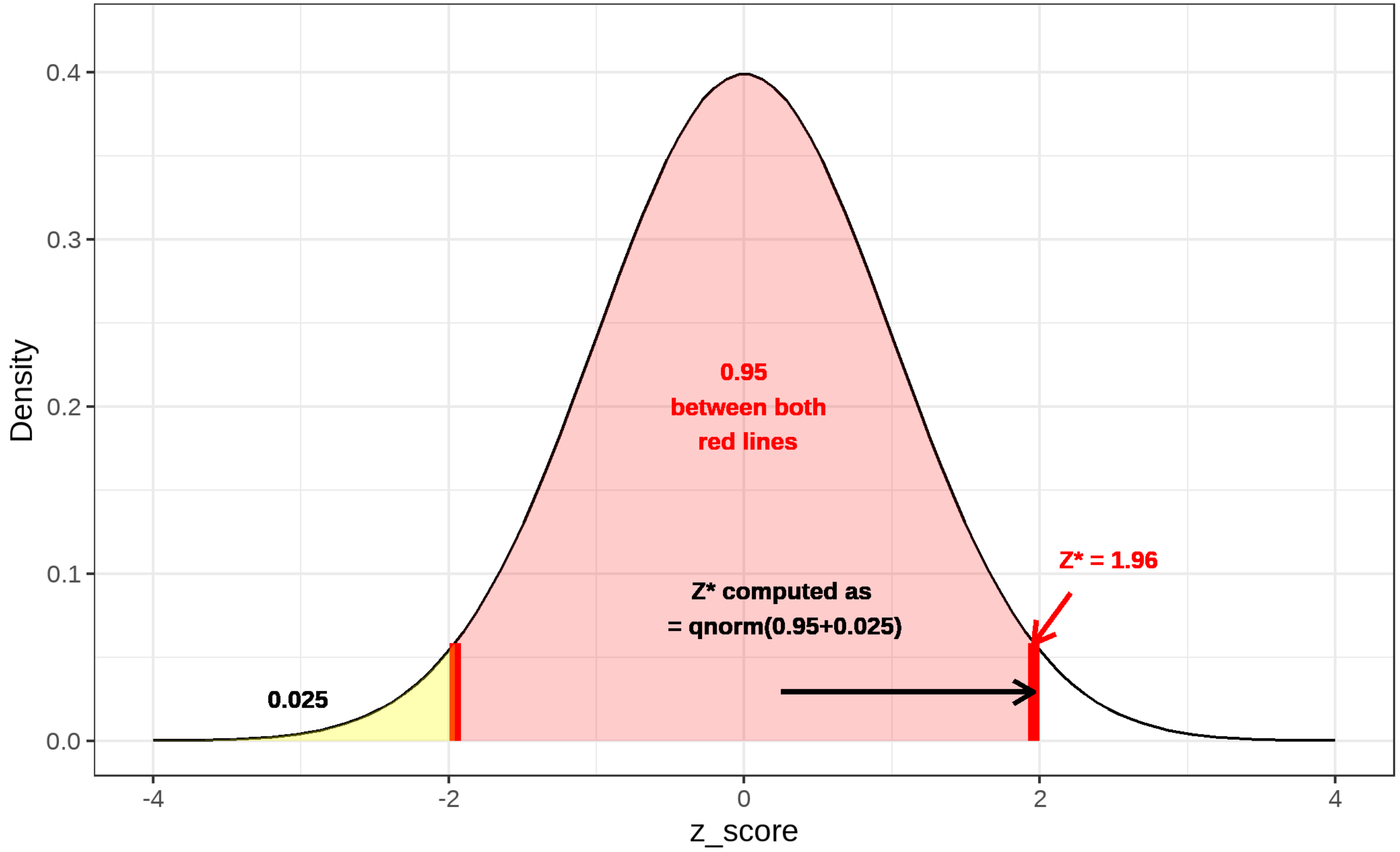
If we assume the forecast errors follow a Normal Distribution, an approximate $100(1-\alpha)$ prediction interval can be computed as follows:

$$\hat{F}_t \pm Z^* * SD,$$

where:

- $\hat{F}_t$ forecast at time t.
- The RMSE can be used as an estimate of the standard deviation of the forecast errors.
- Z^* is the quantile corresponding to $100(1-\frac{\alpha}{2})$.

Recall: Standard Normal Distribution



See the interactive version of this chart at Interactive Standard Normal Dist.

Prediction Intervals for TSLA: "By Hand"

```
tsla_df
```

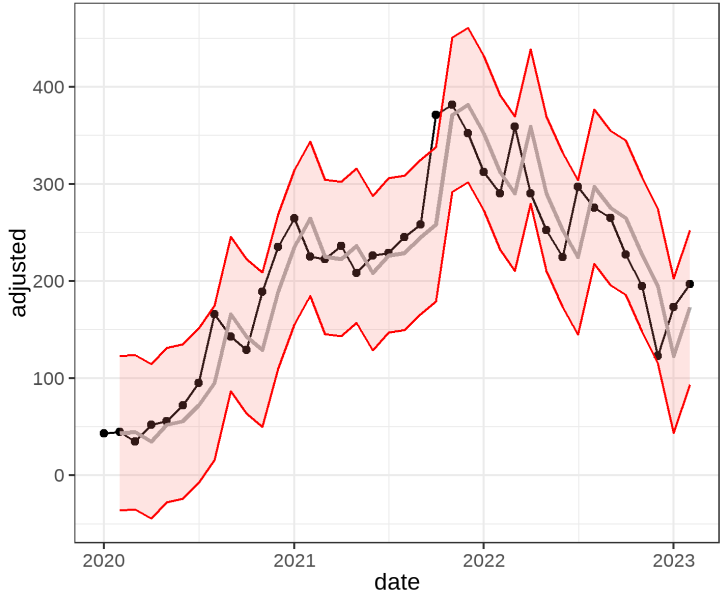
```
## # A tibble: 38 × 6
##    symbol date       year  month adjusted naive_f
##    <chr>  <date>     <fct> <ord>    <dbl>   <dbl>
##  1 tsla   2020-01-01 2020  Jan       43.4    NA  
##  2 tsla   2020-02-01 2020  Feb       44.5    43.4
##  3 tsla   2020-03-01 2020  Mar       34.9    44.5
##  4 tsla   2020-04-01 2020  Apr       52.1    34.9
##  5 tsla   2020-05-01 2020  May       55.7    52.1
##  6 tsla   2020-06-01 2020  Jun       72.0    55.7
##  7 tsla   2020-07-01 2020  Jul       95.4    72.0
##  8 tsla   2020-08-01 2020  Aug      166.     95.4
##  9 tsla   2020-09-01 2020  Sep      143.    166. 
## 10 tsla   2020-10-01 2020  Oct      129.    143. 
## # … with 28 more rows
```

Prediction Intervals for TSLA: "By Hand"

```r
# 95% prediction_interval
alpha = 0.95
curve_prob = 1 - ((1-alpha)/2)

# adding the error since it was not saved to tsla_df
tsla_df = tsla_df |> dplyr::mutate( error = adjusted - naive_f   )

# recomputing the rmse and saving it to an object titled rmse
tsla_df |>
  dplyr::group_by(symbol) |>
  dplyr::summarise(rmse = error^2 |> mean(na.rm = T) |> sqrt() ) |>
  dplyr::pull(rmse)  -> # pull rmse value from tibble (i.e., convert to tibble/vec)
  rmse

# computing the prediction intervals for our data
tsla_df =
  tsla_df |>
  dplyr::mutate(
    pi_l = naive_f - (abs(qnorm(curve_prob))*rmse),
    pi_u = naive_f + (abs(qnorm(curve_prob))*rmse)
  )
```

Prediction Intervals for TSLA: "By Hand"

```
tsla_df |>
  ggplot2::ggplot(ggplot2::aes(x = date)) +
  # Plotting the actual in black
  ggplot2::geom_point(ggplot2::aes(y = adjusted)) +
  ggplot2::geom_line(ggplot2::aes(y = adjusted)) +
  # Plotting the forecast in darkgray
  ggplot2::geom_line(ggplot2::aes(y = naive_f), color =
  # Plotting the PI in red
  ggplot2::geom_ribbon(
    ggplot2::aes(ymin = pi_l, ymax = pi_u, fill = '95%
  ) +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = 'none')
```



Recap

Summary of Main Points

By now, you should be able to do the following:

- Compute the nonseasonal naive forecast.
- Apply and interpret measures of forecast accuracy.
- Interpret prediction intervals for a simple forecast.

Things to Do to Prepare for Our Next Class

- Go over your notes and read through Chapter 2 of our reference book.
- **Potential Practice Problems:**
  - Extract data using either the `tq_get()` (tidyquant package) or the `covid19()` (COVID19 package), and compute the transformations using a manual (i.e., Excel) approach and R.
  - **Reference Book Example:** For the Means approaches in Example 2.7 (P.49), use R to compute the 7 error forecasting metrics (data available here).
  - **Reference Book Exercise 2.12:** Compute the forecast errors for the naive forecast.
- Complete Assignment 05 on Canvas.