ISA 444: Business Forecasting

08: Nonseasonal Smoothing and Forecasting

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

Recap: Lab 01 - Tidying the Data

```
macro = tidyquant::tq_get(
  # macro indicators of interest
  x = c('UNRATE', 'GNP', 'RHORUSQ156N'),
  from = '2018-01-01', to = '2023-02-12',
  # FRED Data --> set get to 'economic.data'
  get = 'economic.data'
)

stocks_crypto = tidyquant::tq_get(
  # you can pull stocks and crypto from
  # the Yahoo Finance API with a single call
  x = c("PM", "UPS",  "SYK",  "PCAR",
        'BTC-USD', 'ETH-USD'),
  from = '2018-01-01', to = '2023-02-12'
) |> # three columns with same col names as macro
  dplyr::select(symbol, date, adjusted) |>
  # rename adjusted to price to match macro tibble
  dplyr::rename(price = adjusted)

# stacking/putting both tibbles on top of each othe
all_data = dplyr::bind_rows(stocks_crypto, macro)
```

```
tail(all_data, n = 12)
```

```
## # A tibble: 12 × 3
##    symbol       date       price
##    <chr>        <date>     <dbl>
##  1 RHORUSQ156N 2020-01-01  65.3
##  2 RHORUSQ156N 2020-04-01  67.9
##  3 RHORUSQ156N 2020-07-01  67.4
##  4 RHORUSQ156N 2020-10-01  65.8
##  5 RHORUSQ156N 2021-01-01  65.6
##  6 RHORUSQ156N 2021-04-01  65.4
##  7 RHORUSQ156N 2021-07-01  65.4
##  8 RHORUSQ156N 2021-10-01  65.5
##  9 RHORUSQ156N 2022-01-01  65.4
## 10 RHORUSQ156N 2022-04-01  65.8
## 11 RHORUSQ156N 2022-07-01  66
## 12 RHORUSQ156N 2022-10-01  65.9
```

Recap: Lab 01 - Naive Forecasting and Errors

```
# grouping by symbol to ensure calculations
# are done for each symbol separately
all_data = all_data |>
  dplyr::group_by(symbol)

all_data =
  all_data |> # now grouped
  # mutate creates new columns
  dplyr::mutate(
    # dplyr::lag since input is num vec
    naive_o = dplyr::lag(price),
    # forecasting error: actual - forecast
    error_o = price - naive_o,
    # percent error = 100*error/actual
    pe_o = 100*error_o/price
  )

all_data |>
  # summarise used --> each calc per group
  dplyr::summarise(
    # mean error per symbol
    me_o = mean(error_o, na.rm = T),
    # mean absolute percent error per symbol
    mape_o = mean(abs(pe_o), na.rm = T)
  )
```

From the printout below, **note the following:**

- Number of rows = number of symbols
- Number of columns = 1 col per each grouping variable + 1 col per each calc in summarise

```
## # A tibble: 9 × 3
##   symbol          me_o mape_o
##   <chr>          <dbl>  <dbl>
## 1 BTC-USD       4.35    2.58
## 2 ETH-USD       0.397   3.49
## 3 GNP         301.      2.44
## 4 PCAR          0.0256  1.24
## 5 PM            0.0182  1.16
## 6 RHORUSQ156N   0.0895  0.676
## 7 SYK           0.0896  1.31
## 8 UNRATE       -0.01    5.61
## 9 UPS           0.0615  1.28
```

Extension: Lab 01 - Let Us Talk about Nesting

```
# the nest function creates column-lists
# and 1 row per group
tidyr::nest(all_data |> dplyr::select(symbol, date, price) )
```

```
## # A tibble: 9 × 2
## # Groups:   symbol [9]
##   symbol      data
##   <chr>       <list>
## 1 PM          <tibble [1,287 × 2]>
## 2 UPS         <tibble [1,287 × 2]>
## 3 SYK         <tibble [1,287 × 2]>
## 4 PCAR        <tibble [1,287 × 2]>
## 5 BTC-USD     <tibble [1,869 × 2]>
## 6 ETH-USD     <tibble [1,869 × 2]>
## 7 UNRATE      <tibble [61 × 2]>
## 8 GNP         <tibble [19 × 2]>
## 9 RHORUSQ156N <tibble [20 × 2]>
```

Extension: Lab 01 - Let Us Talk about Nesting

```
# the nest function creates column-lists
# and 1 row per group
tidyr::nest(all_data |> dplyr::select(symbol, date, price) ) |>
  # combine mutate with purrr::map
  # map transform their input by applying a function to each list
  # returning an object of the same length as the input.
  # map() always returns a list.
  dplyr::mutate(
    # create a date column-list
    date = purrr::map(.x = data, .f = magrittr::extract2, 'date'),
    # create a price column-list
    price = purrr::map(.x = data, .f = magrittr::extract2, 'price'),
    # create a column list of naive forecasts
    naive_fit = purrr::map(
      .x = price, .f = dplyr::lag
    ),
    # create a column list of accuracy metrics
    # we used map2 since we wanted to apply the function to two lists
    acc_metrics = purrr::map2(
      .x = naive_fit, .y = price,
      .f = forecast::accuracy
    ),
    # using map_dbl to have a singular value in each column
    me = purrr::map_dbl(.x = acc_metrics, .f = magrittr::extract2, c(
    mae = purrr::map_dbl(.x = acc_metrics, .f = magrittr::extract2, c
    mape = purrr::map_dbl(.x = acc_metrics, .f = magrittr::extract2,
  )
```

```
## # A tibble: 9 × 9
## # Groups:   symbol [9]
##   symbol      data     date        pri
##   <chr>       <list>   <list>      <li
## 1 PM          <tibble> <date>      <db
## 2 UPS         <tibble> <date>      <db
## 3 SYK         <tibble> <date>      <db
## 4 PCAR        <tibble> <date>      <db
## 5 BTC-USD     <tibble> <date>      <db
## 6 ETH-USD     <tibble> <date>      <db
## 7 UNRATE      <tibble> <date [61]> <db
## 8 GNP         <tibble> <date [19]> <db
## 9 RHORUSQ156N <tibble> <date [20]> <db
## # … with abbreviated variable names ¹n
```

Learning Objectives for Today's Class

- Describe the benefits and drawbacks of judgmental and quantitative forecasing methods.
- Explain the difference between causal and extrapolative forecasting.
- Describe and apply smoothing with a cumulative average.
- Describe and apply forecasting with a moving average.

Extrapolative vs Causal Forecasts

Key Terms

Smoothing is usually done to help us better see patterns/trends, e.g., in time series.

- Generally smooth out the irregular roughness to see a clearer signal.
- For seasonal data, we can smooth out the seasonality so that we can identify the trend.
- Smoothing **does not provide us with a model**, but it can be a good first step in describing various components of the series.

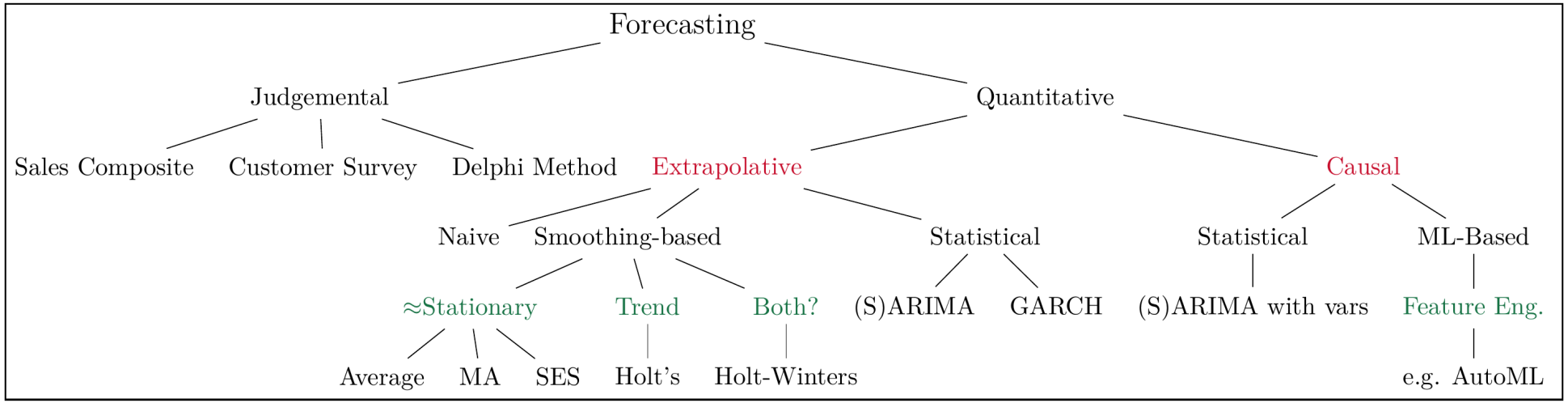
A **filter** is sometimes used to describe a smoothing procedure. For example, we have applied a median filter of window size 21 to smooth wearable sensors' data..

A **forecast** is a prediction or estimate of an actual outcome expected in a future time period or for another situation

---

The smoothing definition and bullet points are based on STAT 510: Applied Time Series Analysis | Penn State. On the other hand, the forecast definition is from Ord, K., Fildes, R., & Kourentzes, N. (2017). *Principles of Business Forecasting* (2nd ed., p. 3).

Overview of Univariate Forecasting Methods



A 10,000 foot view of forecasting techniques

Notes: My (incomplete) classification of **univariate**forecasting techniques, i.e., they exclude popular approaches used in multivariate time series forecasting.

Judgemental Vs Quantitative Forecasts

Judgmental Forecasting: The process of producing forecasts based on purely subjective information. The integration of subjective information may be made informally or through a structured process. The forecasts may also be obtained by aggregating the subjective forecasts of a number of individuals.

Quantitative Forecasting: Forecasting based on the application of an explicit analysis of numerical data. This kind of forecasting may be extrapolative, causal, or a blend of both.

- **Causal Forecast**: a dependent variable is forecast using explanatory variables.
- **Extrapolative:** a dependent variable is forecast using only the past values of the dependent variables. The future is "extrapolated" from the past.

Definitions are based on Dr. Allison Jones-Farmer's lecture notes, Miami University, Spring 2020.

Potential Drawbacks for Judgmental Forecasts

When using judgmental forecasts, forecasters may succumb to:

- An **availability bias**, when the forecaster relies too heavily on easily available and memorable information.
- The **representativeness heuristic**, when the forecaster matches a situation to a similar earlier event without taking into account its frequency of occurrence.
- The **anchoring** and **adjustment heuristic**, when the forecaster uses (anchors onto) an initial value such as the last observation and then adjusts the value to give a revised forecast.
- Over-optimism or **motivational bias** when the forecaster is motivated to bias the forecast towards a preferred state.

These biases can lead to invalid forecasts; they may lead to poor decision making, particularly when combined with overconfidence in their beliefs as to the accuracy of their forecasts.

Slide adapted from Dr. Allison Jones-Farmer's lecture notes, Miami University, Spring 2020.

Causal Vs Extrapolative Forecasts

Causal Forecasts

- Causal methods use data from **sources other than the series being predicted**.
- If Y is the phenomenon to forecast and $X_1, X_2, \ldots, X_n$ are the n variables we believe to be related to Y, then a causal model is one in which the forecast for Y is some function of these variables: $Y = f(X_1, X_2, \ldots, X_n)$.
- Econometric models are causal models in which the relationship between Y and $X_1, X_2, \ldots, X_n$ is linear. That is:

$$Y = a_o + a_1 X_1 + a_2 X_2 + \cdots + a_n X_n$$

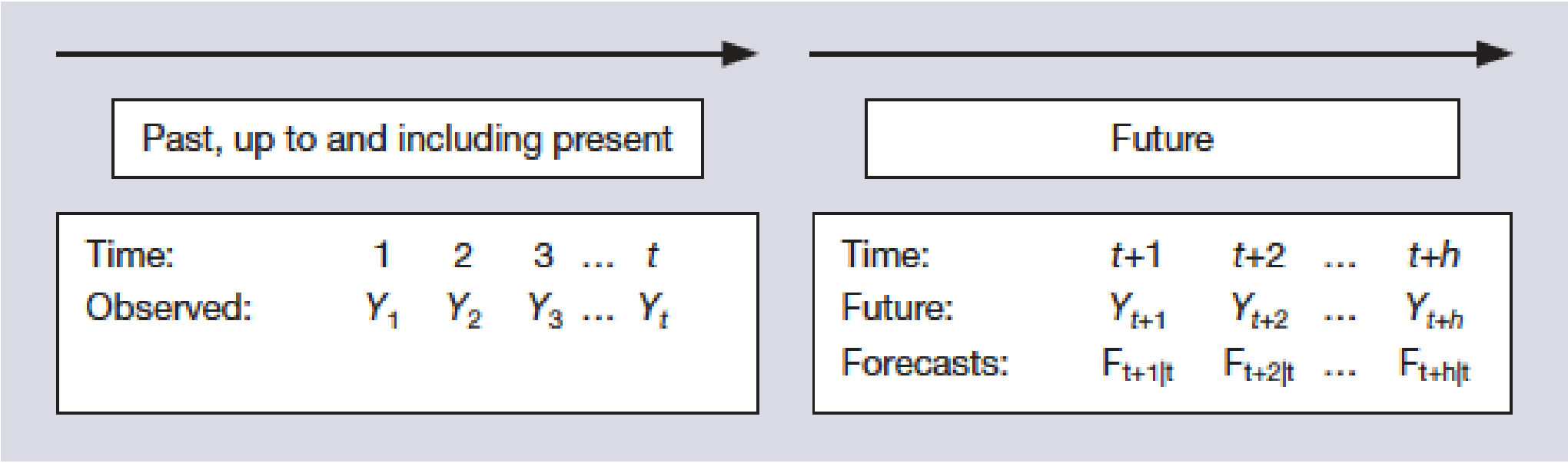
for some constants $a_1, a_2, \ldots, a_n$

Causal Forecasts

- After watching the video, please go to TED Ed.
- Answer the five questions in the think tab.

Extrapolative Methods: A General Framework

Some **forecasting methods** we will discuss use the values of the series to extrapolate into the future. These **extrapolative methods** often work well for short term forecasts.



General Framework for Forecasting with a Single Series

Mean-Based Smoothing/Forecasting Techniques

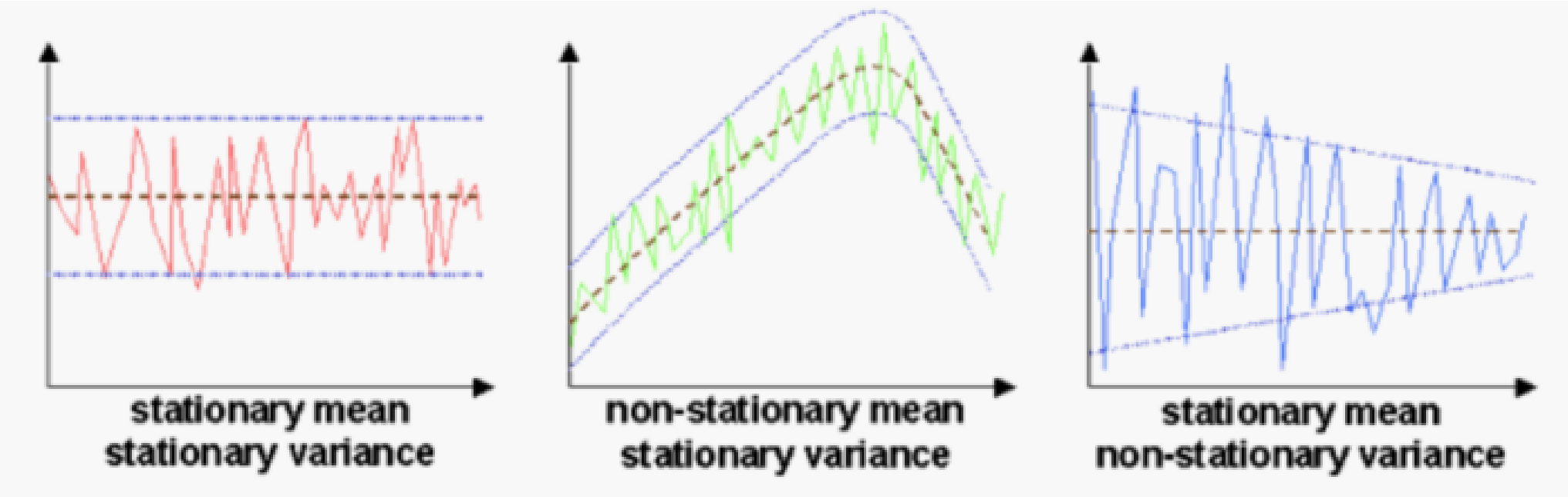
Key Point: "Weak" Stationarity

Today, you are being introduced to methodologies where the time series exhibits **NO** trends and no seasonal patterns.

- *A stationary time series is one whose properties do not depend on the time at which the series is observed.*
- *Some cases can be confusing — a time series with cyclic behavior (but with no trend or seasonality) is stationary. This is because the cycles are not of a fixed length, so before we observe the series we cannot be sure where the peaks and troughs of the cycles will be.*
- *In general, a stationary time series will have no predictable patterns in the long-term. Time plots will show the series to be roughly horizontal (although some cyclic behavior is possible), with constant variance.*

Source: The text presented here is from Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*, 2nd Edition, OTexts.com/fpp2.

Weak Stationarity: A Visual



Constancy in mean and variance

Image Source: Palachy, S. (2019). Stationarity in time series analysis, towardsdatascience.com.

Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
usdc = tidyquant::tq_get(
  x = 'usdc-USD', from = '2019-01-01', to = '2023-02-01',
  periodicity = 'monthly'
)#<<

# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

u_ts
```

```
##           Jan      Feb      Mar      A
## 2019 1.012485 1.013673 1.001494 1.0153
## 2020 1.028374 1.004586 1.001334 1.0018
## 2021 0.999961 1.000296 0.999605 0.9998
## 2022 0.999539 0.999853 0.999447 1.0002
## 2023 1.000058 1.000057
##           Sep      Oct      Nov      D
## 2019 1.001386 1.004975 0.997658 1.0037
## 2020 1.000564 1.000278 1.000192 0.9998
## 2021 1.000140 0.999724 0.999456 1.0000
## 2022 1.000066 0.999996 1.000008 1.0000
## 2023
```

Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf(
  y = u_ts, # time series object of interest
  h = 5, # forecast five months ahead
  level = 95 # 95% prediction interval
)

# exploring the fit_overall_mean object
class(fit_overall_mean)
names(fit_overall_mean)
```

```
## [1] "forecast"
```

```
##  [1] "method"    "level"     "x"
##  [7] "upper"     "model"     "lambda"
```

Some Definitions:

- *x:* The original time series
- *fitted:* Fitted values
- *residuals:* Residuals from the fitted model. That is x - fitted values
- *mean:* Point forecasts as a time series

Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# what does the model object return when printed
fit_overall_mean
```

```
##          Point Forecast     Lo 95    H
## Mar 2023       1.002167 0.9916601 1.01
## Apr 2023       1.002167 0.9916601 1.01
## May 2023       1.002167 0.9916601 1.01
## Jun 2023       1.002167 0.9916601 1.01
## Jul 2023       1.002167 0.9916601 1.01
```

Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# let us print fit_overall_mean$mean
fit_overall_mean$mean
```

```
##           Mar      Apr      May      J
## 2023 1.002167 1.002167 1.002167 1.0021
```

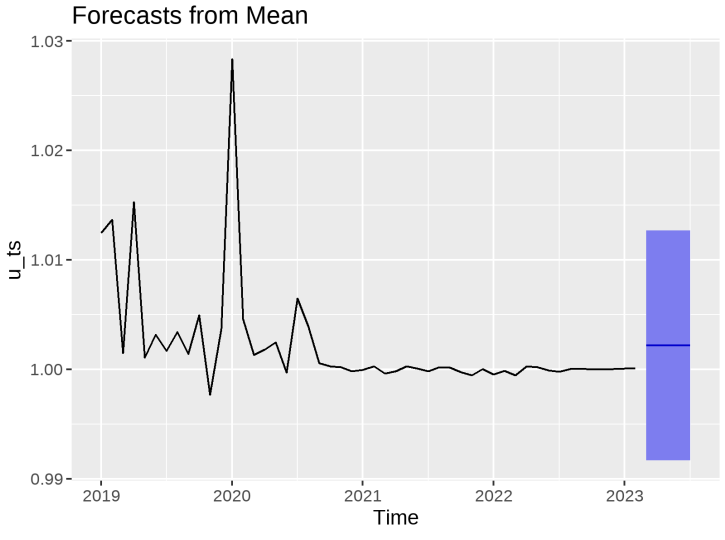
Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```r
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# let us plot what we have
forecast::autoplot(fit_overall_mean)
```



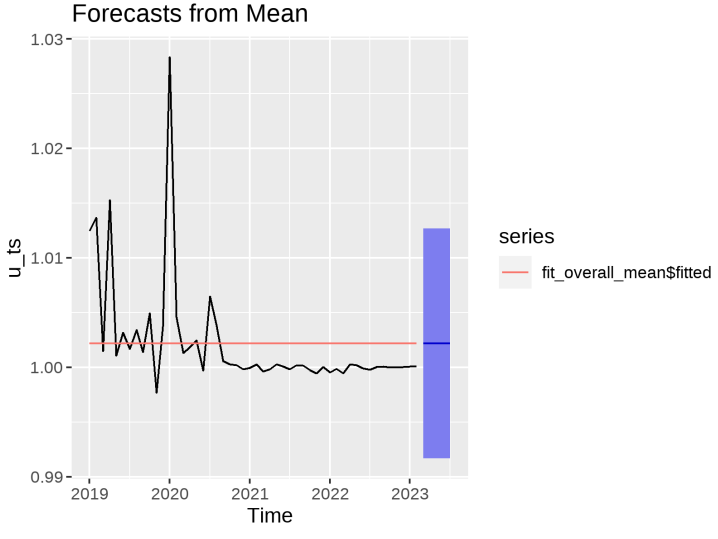
Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```r
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# let us plot what we have
forecast::autoplot(fit_overall_mean) +
  forecast::autolayer( fit_overall_mean$fitted )
```



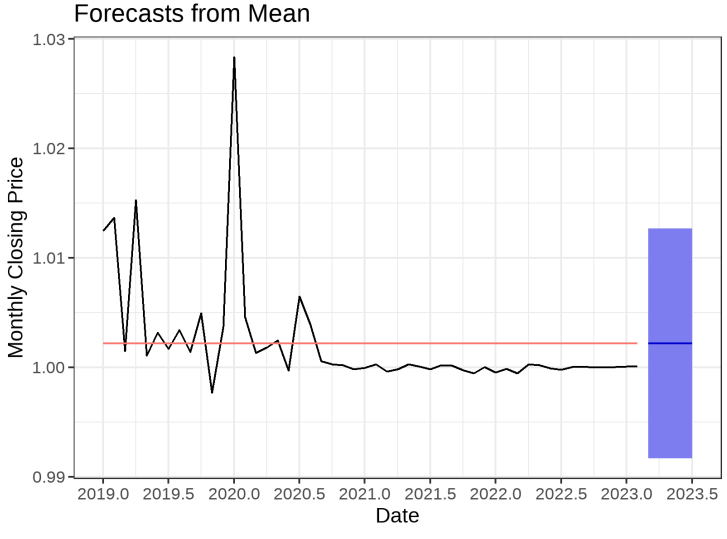
Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# let us plot what we have
forecast::autoplot(fit_overall_mean) +
  forecast::autolayer( fit_overall_mean$fitted ) +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = 'none') +
  ggplot2::scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  ggplot2::labs(x = 'Date', y = 'Monthly Closing Price')
```



Overall Average: Almost Never Used in Practice

If you have a series that stays pretty constant over time, you could just use the overall average for smoothing. In R, we can use the `forecast::meanf()` to smooth using the overall mean.

```
# recommendation create a ts_object of your data
u_ts = ts(data = usdc$adjusted, start = c(2019,01), frequency = 12)

# using the meanf function from the forecast package
fit_overall_mean = forecast::meanf( #
  y = u_ts, # time series object of interest #
  h = 5, # forecast five months ahead #
  level = 95 # 95% prediction interval #
)

# let us check the forecast accuracy
# We only need to input forecast object from
# (we could have also inputted the fitted and actuals)
forecast::accuracy(object = fit_overall_mean)
```

```
##                          ME       RMSE
## Training set -9.103775e-17 0.00512493
##                       ACF1
## Training set 0.2266401
```

Cumulative Average Smoothing

If you have a series that stays pretty constant over time, you could just use the **cumulative** average for smoothing. In R, we can use the `dplyr::cummean()` to smooth using the cumulative mean.

```
usdc = usdc |>
  dplyr::mutate(c_mean_s = dplyr::cummean(adjusted))

usdc |> dplyr::select(date, adjusted, c_mean_s)
```

```
## # A tibble: 50 × 3
##    date       adjusted c_mean_s
##    <date>        <dbl>    <dbl>
##  1 2019-01-01     1.01     1.01
##  2 2019-02-01     1.01     1.01
##  3 2019-03-01     1.00     1.01
##  4 2019-04-01     1.02     1.01
##  5 2019-05-01     1.00     1.01
##  6 2019-06-01     1.00     1.01
##  7 2019-07-01     1.00     1.01
##  8 2019-08-01     1.00     1.01
##  9 2019-09-01     1.00     1.01
## 10 2019-10-01     1.00     1.01
## # … with 40 more rows
```

Moving Average Smoothing

If you believe recent data is more pertinent, you could just use the **moving** average for smoothing. In R, we can use the `zoo::rollmeanr()` to smooth using a rolling mean (moving average).

```
usdc = usdc |>
  dplyr::mutate(
    ma3 = zoo::rollmeanr(x = adjusted, k =3, fill = NA),
    ma7 = zoo::rollmeanr(x = adjusted, k =7, fill = NA)
    )

usdc |>
  # what I am doing in the next function can be skipped
  # printing 5 decimal points since the numbers are not varying much
  dplyr::mutate(
    dplyr::across(dplyr::where(is.numeric), ~ tibble::num(., digits =
  dplyr::select(
    date, adjusted, c_mean_s, ma3, ma7
  )
```

```
## # A tibble: 50 × 5
##    date        adjusted  c_mean_s
##    <date>     <num:.5!> <num:.5!> <num
##  1 2019-01-01   1.01249   1.01249  NA
##  2 2019-02-01   1.01367   1.01308  NA
##  3 2019-03-01   1.00149   1.00922   1.
##  4 2019-04-01   1.01530   1.01074   1.
##  5 2019-05-01   1.00105   1.00880   1.
##  6 2019-06-01   1.00319   1.00787   1.
##  7 2019-07-01   1.00169   1.00698   1.
##  8 2019-08-01   1.00344   1.00654   1.
##  9 2019-09-01   1.00139   1.00597   1.
## 10 2019-10-01   1.00498   1.00587   1.
## # … with 40 more rows
```

From Smoothing to Forecasting

So Can We Use the Previous Results for Forecasting?

Comments on Previously Printed Results: (Insert below)

- **c_mean_s:** ....
- **ma3:** ....
- **ma7:** ....
- **Equivalence of c_mean_s with ma:** ....

A Live Demo to Generate Forecasts

Recap

Summary of Main Points

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

- Describe the benefits and drawbacks of judgmental and quantitative forecasing methods.
- Explain the difference between causal and extrapolative forecasting.
- Describe and apply smoothing with a cumulative average.
- Describe and apply forecasting with a moving average.

Things to Do to Prepare for Our Next Class

- **Recommended:** Thoroughly read Chapter 3.1-3.2 of our reference book.
- Go through the slides, examples and make sure you have a good understanding of what we have covered.
- If you are interested in additional practice problems, please consider the following problems from your textbook. To access these datasets, please click here.
    - **Practice Problem:** Exercise 3.1 (RMSE is 2.54 and 2.28 for the MA 3 and MA7, respectively).
- **Required:** Complete assignment06, and assignment07.