STAT481/581: Introduction to Time Series Analysis

Ch4. Evaluating forecast accuracy

OTexts.org/fpp3/

Outline

Outline

Fitted values

- $\hat{y}_{t|t-1}$ is the forecast of y_t based on observations $y_1, \dots, y_t$.
- We call these "fitted values".
- Sometimes drop the subscript: $\hat{y}_t \equiv \hat{y}_{t|t-1}$.
- Often not true forecasts since parameters are estimated on all data.

For example:

- $\hat{y}_t = \bar{y}$ for average method.
- $\hat{y}_t = y_{t-1} + (y_T - y_1)/(T-1)$ for drift method.

Forecasting residuals

Residuals in forecasting: difference between observed value and its fitted value: $e_t = y_t - \hat{y}_{t|t-1}$.

Forecasting residuals

Residuals in forecasting: difference between observed value and its fitted value: $e_t = y_t - \hat{y}_{t|t-1}$.

Assumptions

1. $\{e_t\}$ uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero. If they don't, then forecasts are biased.

Forecasting residuals

Residuals in forecasting: difference between observed value and its fitted value: $e_t = y_t - \hat{y}_{t|t-1}$.

Assumptions

1. $\{e_t\}$ uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero. If they don't, then forecasts are biased.

Useful properties (for prediction intervals)

3. $\{e_t\}$ have constant variance.
4. $\{e_t\}$ are normally distributed.

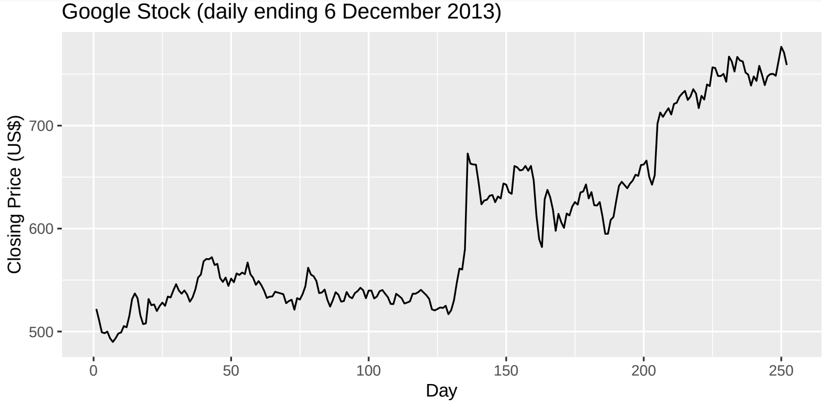
Example: Google stock price

```r
google_2015 <- tsibbledata::gafa_stock %>%
  filter(Symbol == "GOOG", year(Date) == 2015) %>%
  mutate(trading_day = row_number()) %>%
  update_tsibble(index = trading_day, regular = TRUE)
```

```
## # A tsibble: 252 x 9 [1]
## # Key:       Symbol [1]
##    Symbol Date        Open  High   Low Close
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>
##  1 GOOG   2015-01-02  526.  528.  521.  522.
##  2 GOOG   2015-01-05  520.  521.  510.  511.
##  3 GOOG   2015-01-06  512.  513.  498.  499.
##  4 GOOG   2015-01-07  504.  504.  497.  498.
##  5 GOOG   2015-01-08  495.  501.  488.  500.
##  6 GOOG   2015-01-09  502.  502.  492.  493.
##  7 GOOG   2015-01-12  492.  493.  485.  490.
##  8 GOOG   2015-01-13  496.  500.  490.  493.
##  9 GOOG   2015-01-14  492.  500.  490.  498.
## 10 GOOG   2015-01-15  503.  503.  495.  499.
```

Example: Google stock price

```
google_2015 %>%
  autoplot(Close) +
    xlab("Day") + ylab("Closing Price (US$)") +
    ggtitle("Google Stock (daily ending 6 December 2013)")
```



Example: Google stock price

Naïve forecast:

$$\hat{y}_{t|t-1} = y_{t-1}$$

Example: Google stock price

Naïve forecast:

$$\hat{y}_{t|t-1} = y_{t-1}$$

$$e_t = y_t - y_{t-1}$$

Example: Google stock price

Naïve forecast:

$$\hat{y}_{t|t-1} = y_{t-1}$$

$$e_t = y_t - y_{t-1}$$

Note: e_t are one-step-forecast residuals

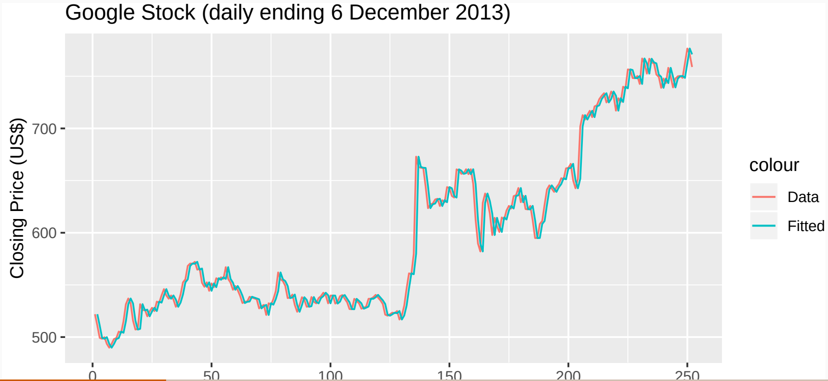
Example: Google stock price

```r
fit <- google_2015 %>% model(NAIVE(Close))
augment(fit)
```

```
## # A tsibble: 252 x 6 [1]
## # Key:       Symbol, .model [1]
##    Symbol .model trading_day Close .fitted  .resid
##    <chr>  <chr>        <int> <dbl>   <dbl>   <dbl>
##  1 GOOG   NAIVE~           1  522.     NA    NA
##  2 GOOG   NAIVE~           2  511.    522. -10.9
##  3 GOOG   NAIVE~           3  499.    511. -11.8
##  4 GOOG   NAIVE~           4  498.    499.  -0.855
##  5 GOOG   NAIVE~           5  500.    498.   1.57
##  6 GOOG   NAIVE~           6  493.    500.  -6.47
##  7 GOOG   NAIVE~           7  490.    493.  -3.60
##  8 GOOG   NAIVE~           8  493.    490.   3.61
##  9 GOOG   NAIVE~           9  498.    493.   4.66
## 10 GOOG   NAIVE~          10  499.    498.   0.915
## # ... with 242 more rows
```

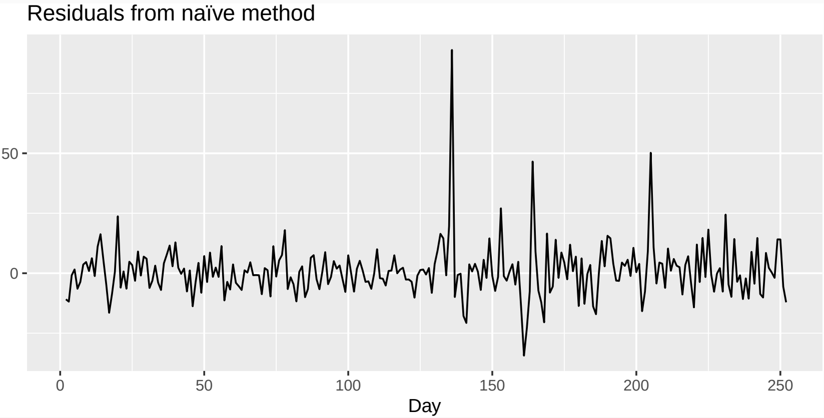
Example: Google stock price

```r
augment(fit) %>%
  ggplot(aes(x = trading_day)) +
    geom_line(aes(y = Close, colour = "Data")) +
    geom_line(aes(y = .fitted, colour = "Fitted")) +
    xlab("Day") + ylab("Closing Price (US$)") +
    ggtitle("Google Stock (daily ending 6 December 2013)")
```



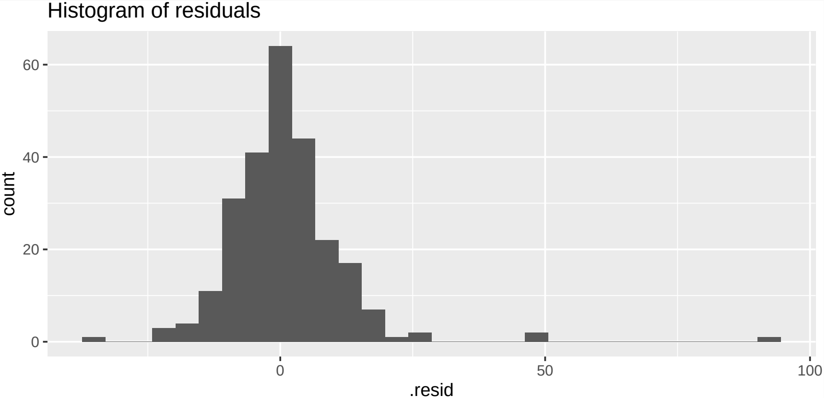
Example: Google stock price

```
augment(fit) %>%
  autoplot(.resid) + xlab("Day") + ylab("") +
    ggtitle("Residuals from naïve method")
```



Example: Google stock price

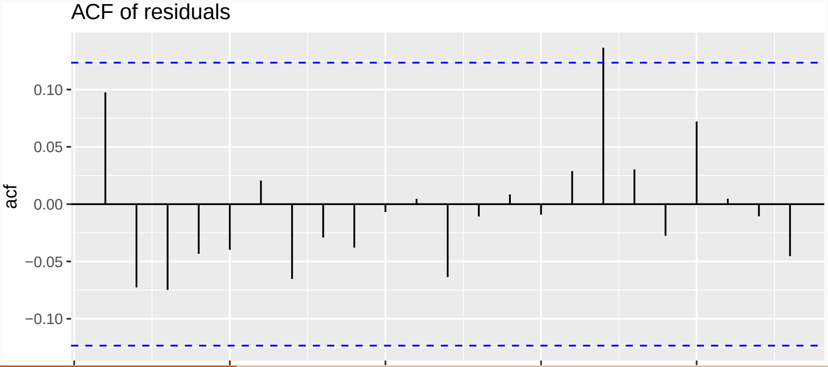
```r
augment(fit) %>%
  ggplot(aes(x = .resid)) +
    geom_histogram(bins = 30) +
    ggtitle("Histogram of residuals")
```



Example: Google stock price

```r
augment(fit) %>% ACF(.resid) %>%
  autoplot() + ggtitle("ACF of residuals")
```

```
## Warning in mutate_impl(.data, dots, caller_env()):
## Vectorizing 'cf_lag' elements may not preserve
## their attributes
```



ACF of residuals

- We assume that the residuals are white noise (uncorrelated, mean zero, constant variance). If they aren't, then there is information left in the residuals that should be used in computing forecasts.
- So a standard residual diagnostic is to check the ACF of the residuals of a forecasting method.
- We *expect* these to look like white noise.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Box-Pierce test

$$Q = T \sum_{k=1}^{h} r_k^2$$

where h is max lag being considered and T is number of observations.

- If each r_k close to zero, Q will be **small**.
- If some r_k values large (positive or negative), Q will be **large**.

Portmanteau tests

Consider a *whole set* of r_k values, and develop a test to see whether the set is significantly different from a zero set.

Ljung-Box test

$$Q^* = T(T+2)\sum_{k=1}^{h}(T-k)^{-1}r_k^2$$

where h is max lag being considered and T is number of observations.

- My preferences: $h = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- Better performance, especially in small samples.

Portmanteau tests

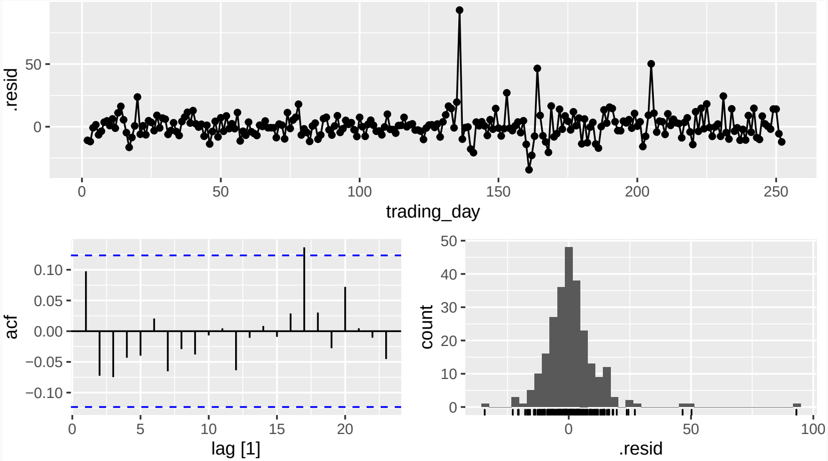
- If data are WN, Q^* has χ^2 distribution with $(h - K)$ degrees of freedom where $K =$ no. parameters in model.
- When applied to raw data, set $K = 0$.

```r
# lag=h and fitdf=K
Box.test(augment(fit)$.resid,
  lag = 10, fitdf = 0, type = "Lj")
```

```
##
##  Box-Ljung test
##
## data:  augment(fit)$.resid
## X-squared = 7.9141, df = 10, p-value =
## 0.6372
```

gg_tsdisplay function

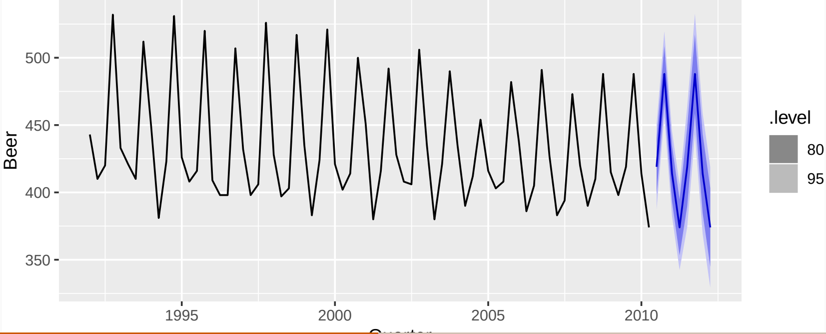
```r
augment(fit) %>%
  gg_tsdisplay(.resid, plot_type = "histogram")
```



Your turn

Compute seasonal naïve forecasts for quarterly Australian beer production from 1992.

```
recent <- aus_production %>% filter(year(Quarter) >= 1992)
fit <- recent %>% model(SNAIVE(Beer))
fit %>% forecast() %>% autoplot(recent)
```



Your turn

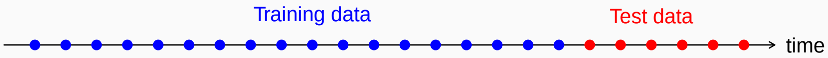
Test if the residuals are white noise.

```
Box.test(augment(fit)$.resid, lag=10, fitdf=0, type="Lj")
augment(fit) %>% gg_tsdisplay(.resid, plot_type = "hist")
```

What do you conclude?

Outline

Training and test sets



- A model which fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.
- The test set must not be used for *any* aspect of model development or calculation of forecasts.
- Forecast accuracy is based only on the test set.

Forecast errors

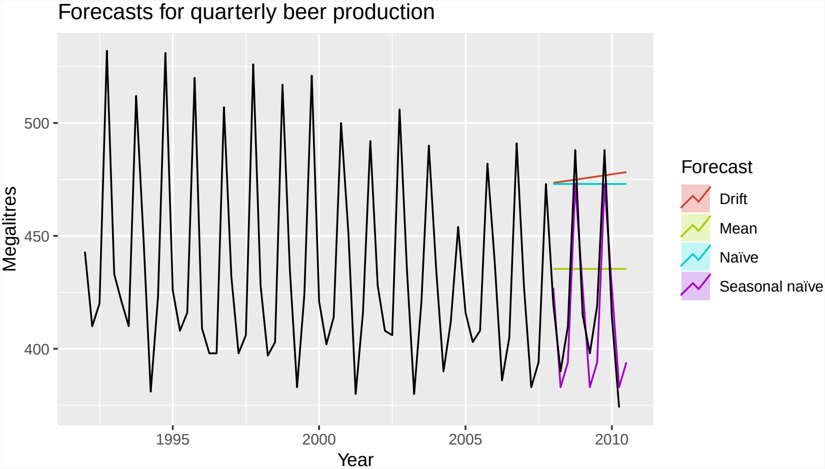
Forecast "error": the difference between an observed value and its forecast.

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

where the training data is given by $\{y_1, \dots, y_T\}$

- Unlike residuals, forecast errors on the test set involve multi-step forecasts.
- These are *true* forecast errors as the test data is not used in computing $\hat{y}_{T+h|T}$.

Measures of forecast accuracy



Measures of forecast accuracy

$$\begin{aligned} y_{T+h} &= \text{$(T+h)$th observation, } h = 1,\dots,H \\ \hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\ e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T} \end{aligned}$$

$$\begin{aligned} \text{MAE} &= \text{mean}(|e_{T+h}|) \\ \text{MSE} &= \text{mean}(e^2_{T+h}) \\ \text{RMSE} &= \sqrt{\text{mean}(e^2_{T+h})} \\ \text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|) \end{aligned}$$

Measures of forecast accuracy

$$\begin{aligned} y_{T+h} &= \text{$(T+h)$th observation, } h = 1, \dots, H \\ \hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\ e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T} \end{aligned}$$

$$\begin{aligned} \text{MAE} &= \text{mean}(|e_{T+h}|) \\ \text{MSE} &= \text{mean}(e_{T+h}^2) \\ \text{RMSE} &= \sqrt{\text{mean}(e_{T+h}^2)} \\ \text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|) \end{aligned}$$

Measures of forecast accuracy

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all t, and y has a natural zero.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006).

For non-seasonal time series,

$$Q = (T-1)^{-1}\sum_{t=2}^{T}|y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naïve method.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

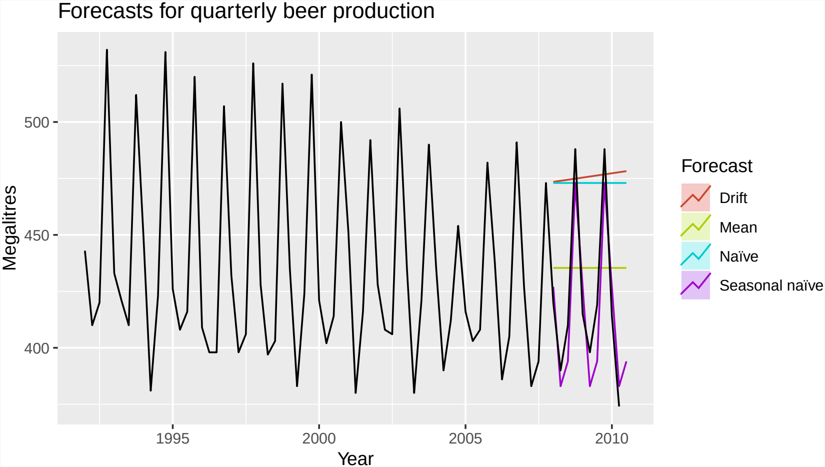
Proposed by Hyndman and Koehler (IJF, 2006).

For seasonal time series,

$$Q = (T-m)^{-1} \sum_{t=m+1}^{T} |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naïve method.

Measures of forecast accuracy



Training set accuracy

```
recent_production <- aus_production %>%
  filter(year(Quarter) >= 1992)
train <- recent_production %>% filter(year(Quarter) <= 2007)
beer_fit <- train %>%
  model(
    Mean = MEAN(Beer),
    Naïve = NAIVE(Beer),
    Seasonal naïve = SNAIVE(Beer),
    Drift = RW(Beer ~ drift())
  )
accuracy(beer_fit)
```

	RMSE	MAE	MAPE	MASE
Mean method	43.62858	35.23438	7.886776	2.463942
Naïve method	65.31511	54.73016	12.164154	3.827284
Seasonal naïve method	16.78193	14.30000	3.313685	1.000000
Drift method	65.31337	54.76795	12.178793	3.829927

Test set accuracy

```r
beer_fc <- beer_fit %>%
  forecast(h = 10)
accuracy(beer_fc, recent_production)
```

	RMSE	MAE	MAPE	MASE
Drift method	64.90129	58.87619	14.577487	4.1172161
Mean method	38.44724	34.82500	8.283390	2.4353147
Naïve method	62.69290	57.40000	14.184424	4.0139860
Seasonal naïve method	14.31084	13.40000	3.168503	0.9370629

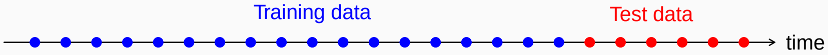
Poll: true or false?

1. Good forecast methods should have normally distributed residuals.
2. A model with small residuals will give good forecasts.
3. The best measure of forecast accuracy is MAPE.
4. If your model doesn't forecast well, you should make it more complicated.
5. Always choose the model with the best forecast accuracy as measured on the test set.

Outline

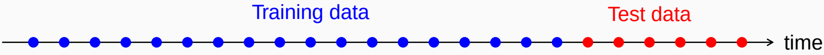
Time series cross-validation

Traditional evaluation

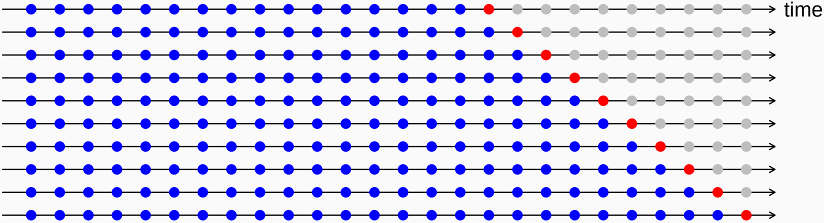


Time series cross-validation

Traditional evaluation

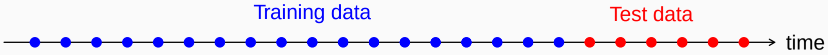


Time series cross-validation

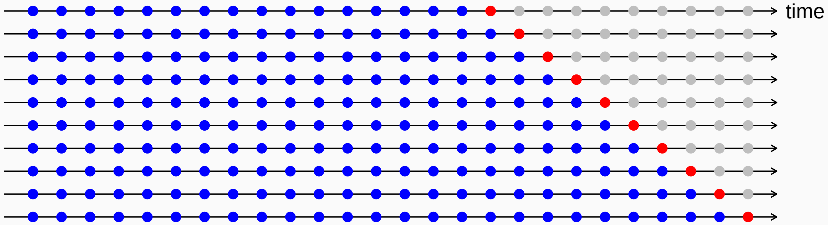


Time series cross-validation

Traditional evaluation



Time series cross-validation



- Forecast accuracy averaged over test sets.
- Also known as "evaluation on a rolling forecasting origin"

Creating the rolling training sets

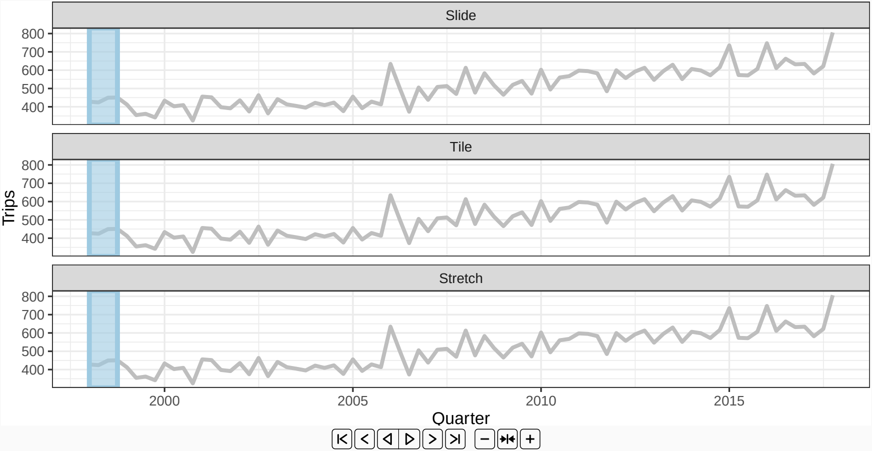
There are three main rolling types which can be used.

- Stretch: extends a growing length window with new data.
- Slide: shifts a fixed length window through the data.
- Tile: moves a fixed length window without overlap.

Three functions to roll a tsibble: `stretch_tsibble()`, `slide_tsibble()`, and `tile_tsibble()`.

For time series cross-validation, stretching windows are most commonly used.

Creating the rolling training sets



Time series cross-validation

Stretch with a minimum length of 3, growing by 1 each step.

```
google_2015_stretch <- google_2015 %>%
  stretch_tsibble(.init = 3, .step = 1) %>%
  filter(.id != max(.id))
```

```
## # A tsibble: 31,623 x 4 [1]
## # Key:       .id [249]
##   Date       Close trading_day   .id
##   <date>     <dbl>       <int> <int>
## 1 2015-01-02  522.           1     1
## 2 2015-01-05  511.           2     1
## 3 2015-01-06  499.           3     1
## 4 2015-01-02  522.           1     2
## 5 2015-01-05  511.           2     2
## 6 2015-01-06  499.           3     2
## 7 2015-01-07  498.           4     2
```

Time series cross-validation

Estimate RW w/ drift models for each window.

```
fit_cv <- google_2015_stretch %>%
  model(RW(Close ~ drift()))
```

```
## # A mable: 249 x 3
## # Key:     .id, Symbol [249]
##     .id Symbol RW(Close ~ drift())
##   <int> <chr>  <model>
## 1     1 GOOG   <RW w/ drift>
## 2     2 GOOG   <RW w/ drift>
## 3     3 GOOG   <RW w/ drift>
## 4     4 GOOG   <RW w/ drift>
## # ... with 245 more rows
```

Time series cross-validation

Produce one step ahead forecasts from all models.

```r
fc_cv <- fit_cv %>%
  forecast(h=1)
```

```
## # A fable: 249 x 5 [1]
## # Key:     .id, Symbol [249]
##      .id Symbol trading_day Close .distribution
##    <int> <chr>        <dbl> <dbl> <dist>
## 1      1 GOOG             4  488. N(488, 0.47)
## 2      2 GOOG             5  490. N(490, 37)
## 3      3 GOOG             6  494. N(494, 47)
## 4      4 GOOG             7  488. N(488, 35)
## # ... with 245 more rows
```

Time series cross-validation

```
# Cross-validated
fc_cv %>% accuracy(google_2015)
# Training set
google_2015 %>% model(NAIVE(Close)) %>% accuracy()
```

	RMSE	MAE	MAPE
Cross-validation	11.26819	7.261240	1.194024
Training	11.18958	7.127985	1.170985

A good way to choose the best forecasting model is to find the model with the smallest RMSE computed using time series cross-validation.