https://workshop.f4sg.org/africast/

Africast-Time Series Analysis & Forecasting Using R

10. Residual diagnostics and cross validation

tsibble

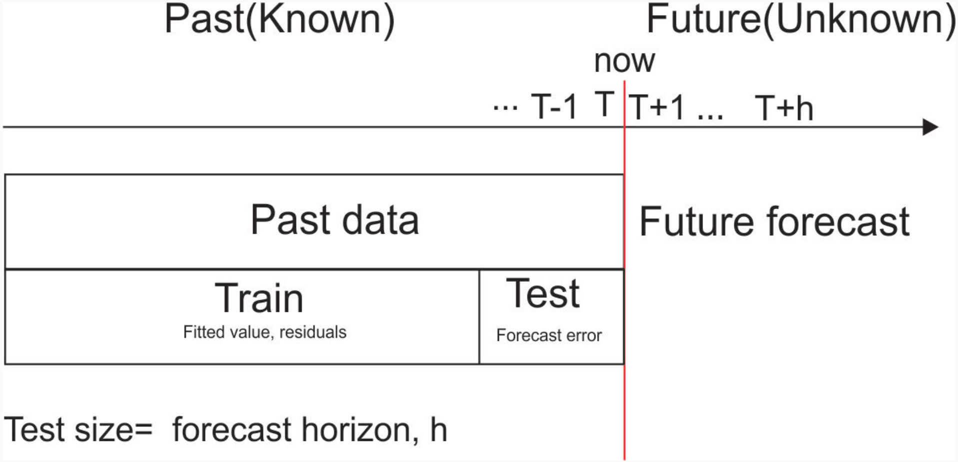
feasts

Fable

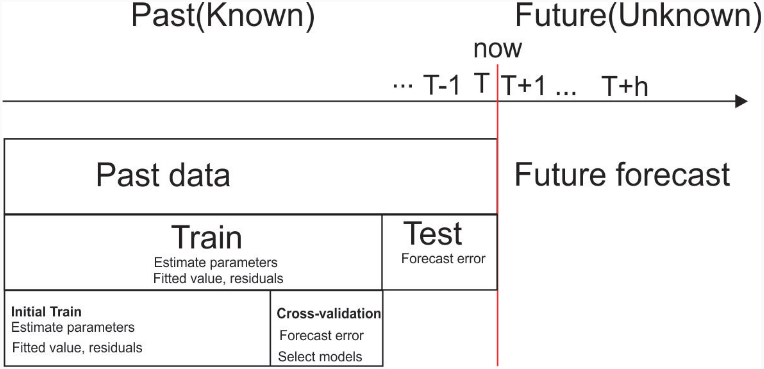
Outline

Outline

Issue with traditional train/test split



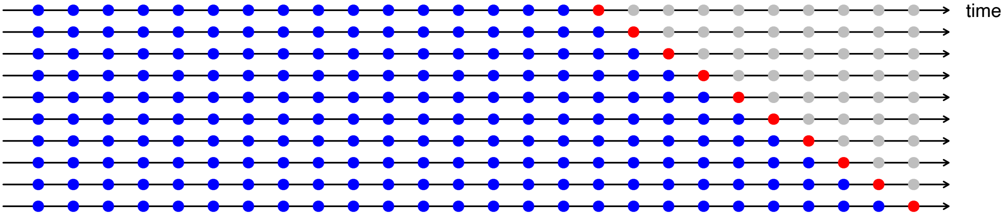
Time series cross-validation



Test size= forecast horizon, h

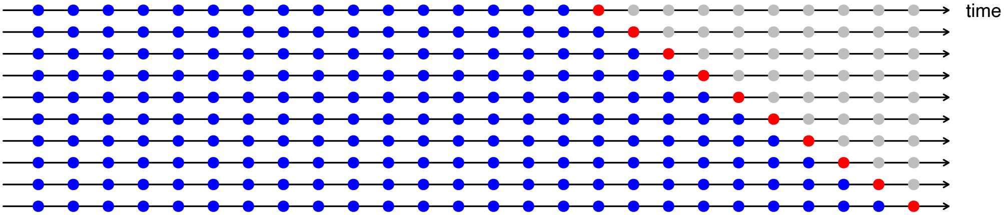
Cross-validation size=nb of experiment+h-1

Time series cross-validation



Time series cross-validation

Time series cross-validation



- Forecast accuracy averaged over test sets

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.

Time series cross-validation

Stretch with a minimum length of 24, growing by 1 each step.

```
forecast_horizon <- 12
test <- antidiabetic_drug_sale |>
  slice((n()-forecast_horizon+1):n())
train <- antidiabetic_drug_sale |>
  slice(1:(n()-forecast_horizon))
drug_sale_tcsv <-  train |> slice(1:(n()-forecast_horizon)) |>
  stretch_tsibble(.init = 24, .step = 1)
```

```
# A tsibble: 2,805 x 3 [1M]
# Key:       .id [55]
     Month  Cost   .id
     <mth> <dbl> <int>
1 2000 Jan 12.5      1
2 2000 Feb  7.46     1
3 2000 Mar  8.59     1
4 2000 Apr  8.47     1
```

Time series cross-validation

Estimate RW w/ drift models for each window.

```
drug_fit_tr <- drug_sale_tcsv |>
  model(snaive=SNAIVE(Cost))
```

```
# A mable: 55 x 2
# Key:     .id [55]
    .id   snaive
  <int>  <model>
1     1 <SNAIVE>
2     2 <SNAIVE>
3     3 <SNAIVE>
4     4 <SNAIVE>
# i 51 more rows
```

Time series cross-validation

Produce 8 step ahead forecasts from all models.

```
drug_fc_tr <- drug_fit_tr |>
  forecast(h=forecast_horizon)
```

```
# A tsibble: 660 x 6 [1M]
# Key:       .id, .model [55]
    .id .model     Month         Cost .mean     h
  <int> <chr>      <mth>       <dist> <dbl> <int>
1     1 snaive 2002 Jan  N(14, 1.7) 14.5      1
2     1 snaive 2002 Feb   N(8, 1.7)  8.05     2
3     1 snaive 2002 Mar  N(10, 1.7) 10.3      3
4     1 snaive 2002 Apr N(9.8, 1.7)  9.75     4
# i 656 more rows
```

Time series cross-validation

```
# Cross-validated
drug_fc_tr |>
  accuracy(antidiabetic_drug_sale,
           measures = list( point_accuracy_measures,
                            interval_accuracy_measures,
                            distribution_accuracy_measures))
```

```
# A tibble: 1 x 13
  .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1 winkler percent
  <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>       <
1 snaive Test   1.48  1.90  1.56  9.09  9.59  1.17  1.20 0.254    9.73      1.
```

Outline

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

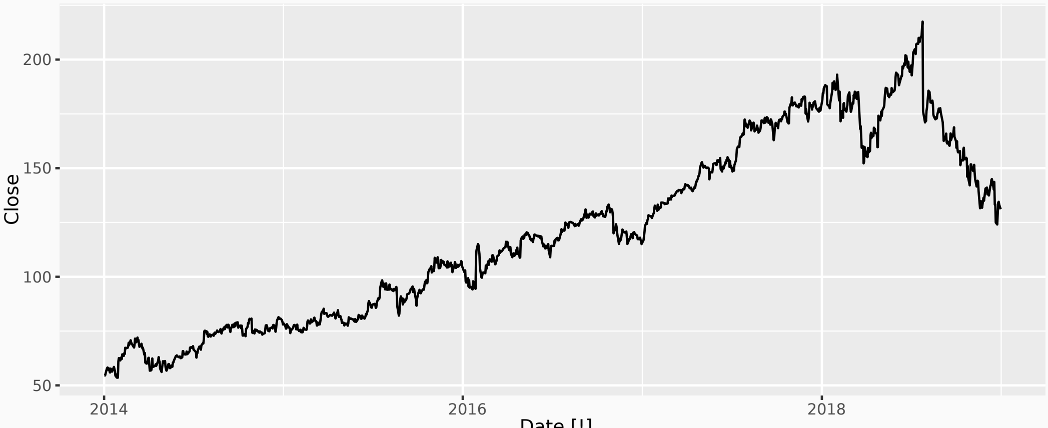
1. $\{e_t\}$ uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. $\{e_t\}$ have mean zero. If they don't, then forecasts are biased.

Useful properties (for prediction intervals)

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

Facebook closing stock price

```
fb_stock <- gafa_stock |>
  filter(Symbol == "FB")
fb_stock |> autoplot(Close)
```



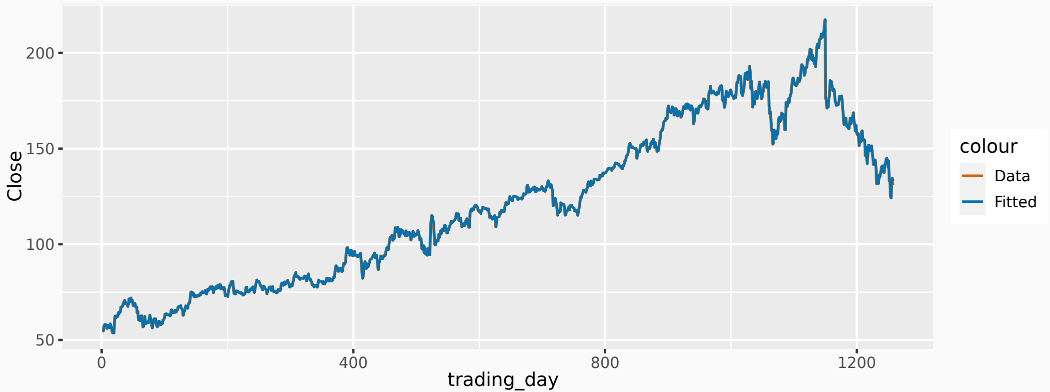
Facebook closing stock price

```
fb_stock <- fb_stock |>
  mutate(trading_day = row_number()) |>
  update_tsibble(index = trading_day, regular = TRUE)
fit <- fb_stock |> model(NAIVE(Close))
augment(fit)
```

```
# A tsibble: 1,258 x 7 [1]
# Key:       Symbol, .model [1]
   Symbol .model       trading_day Close .fitted .resid .innov
   <chr>  <chr>              <int> <dbl>   <dbl>  <dbl>  <dbl>
 1 FB     NAIVE(Close)           1  54.7    NA   NA     NA
 2 FB     NAIVE(Close)           2  54.6    54.7 -0.150 -0.150
 3 FB     NAIVE(Close)           3  57.2    54.6  2.64   2.64
 4 FB     NAIVE(Close)           4  57.9    57.2  0.720  0.720
 5 FB     NAIVE(Close)           5  58.2    57.9  0.310  0.310
 6 FB     NAIVE(Close)           6  57.2    58.2 -1.01  -1.01
 7 FB     NAIVE(Close)           7  57.9    57.2  0.720  0.720
 8 FB     NAIVE(Close)           8  55.9    57.9 -2.03  -2.03
```

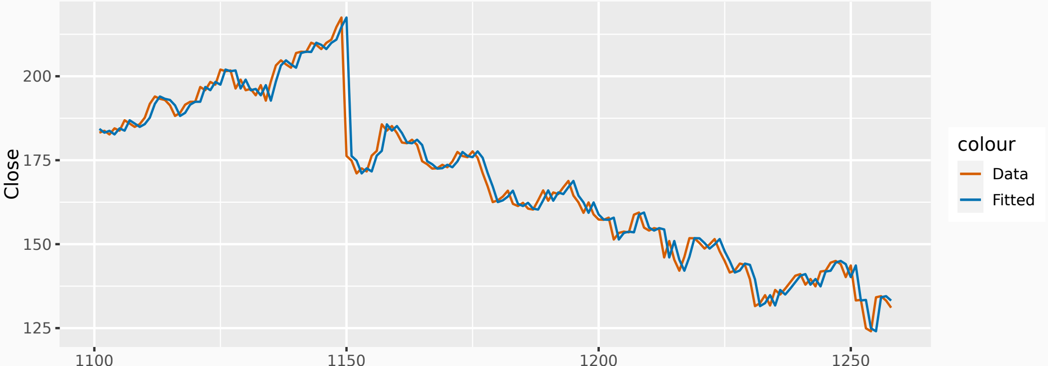
Facebook closing stock price

```
augment(fit) |>
  ggplot(aes(x = trading_day)) +
  geom_line(aes(y = Close, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



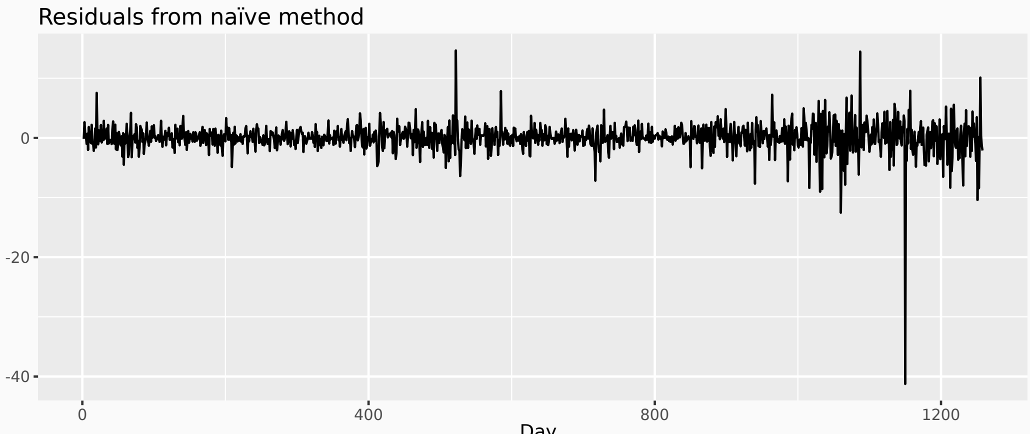
Facebook closing stock price

```r
augment(fit) |>
  filter(trading_day > 1100) |>
  ggplot(aes(x = trading_day)) +
  geom_line(aes(y = Close, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



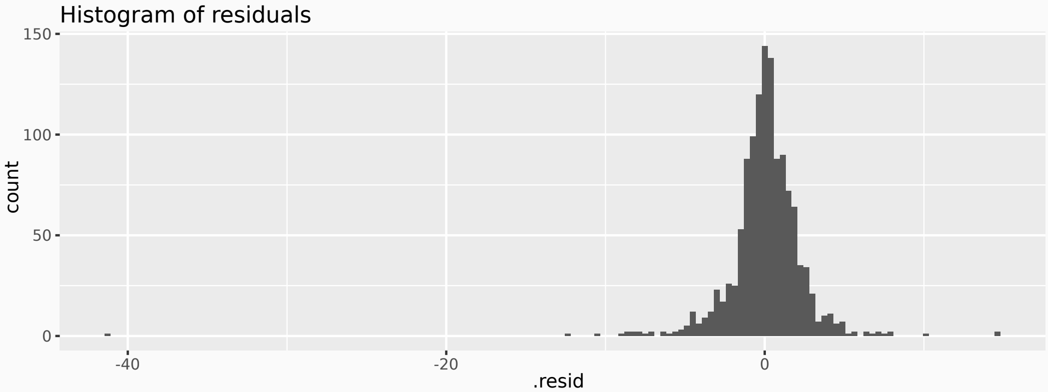
Facebook closing stock price

```
augment(fit) |>
  autoplot(.resid) +
  labs(x = "Day", y = "", title = "Residuals from naïve method")
```



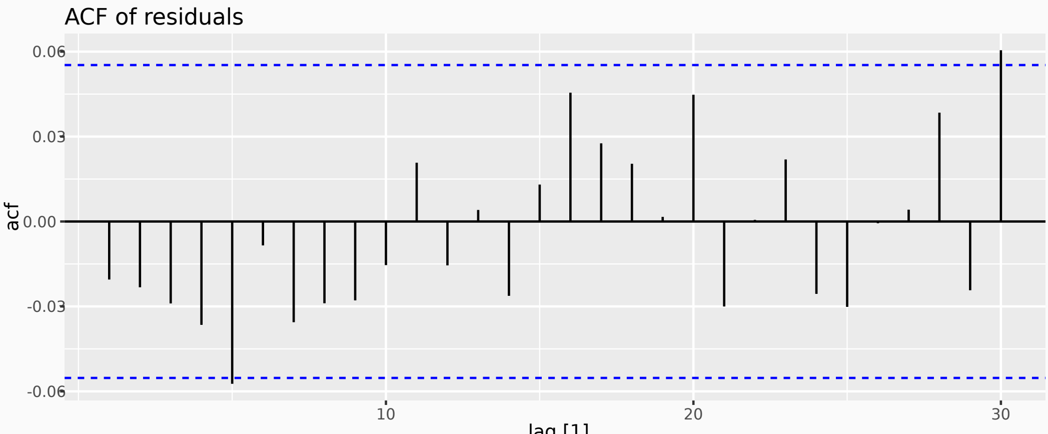
Facebook closing stock price

```
augment(fit) |>
  ggplot(aes(x = .resid)) +
  geom_histogram(bins = 150) +
  labs(title = "Histogram of residuals")
```



Facebook closing stock price

```
augment(fit) |>
  ACF(.resid) |>
  autoplot() + labs(title = "ACF of residuals")
```

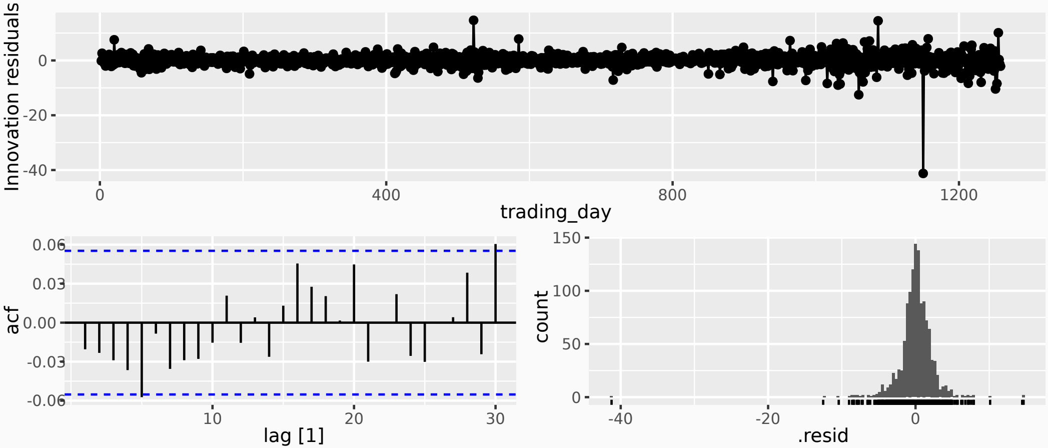


ACF of residuals

- 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.

Combined diagnostic graph

```
fit |> gg_tsresiduals()
```



Ljung-Box test

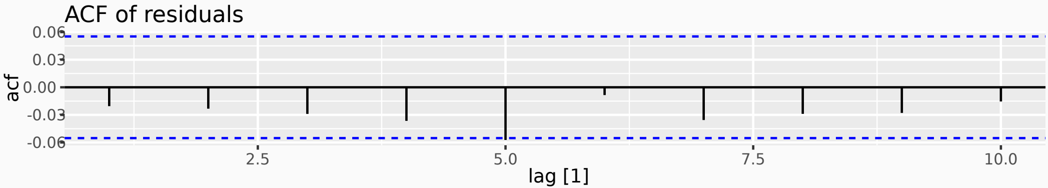
Test whether *whole set* of r_k values is significantly different from zero set.

$$Q = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2 \qquad \text{where } \ell = \text{max lag and } T = \text{\# observations}$$

- If each r_k close to zero, Q will be **small**.
- If some r_k values large ($+$ or $-$), Q will be **large**.
- My preferences: $h = 10$ for non-seasonal data, $h = 2m$ for seasonal data.
- If data are WN and T large, $Q \sim \chi^2$ with ℓ degrees of freedom.

Ljung-Box test

$$Q = T(T+2)\sum_{k=1}^{\ell}(T-k)^{-1}r_k^2$$ where $\ell =$ max lag and $T =$ # observations.




```
# lag = h
augment(fit) |> features(.resid, ljung_box, lag = 10)
```

```
# A tibble: 1 x 4
  Symbol .model       lb_stat lb_pvalue
  <chr>  <chr>          <dbl>     <dbl>
1 FB     NAIVE(Close)    12.1     0.276
```

Outline

Recap

1. First, import your data and prepare them using `tsibble` function.
2. Visualise and see whether your series contains key patetrns Use domain knowledge to understand your data and potential driving factors.
3. Split the data to create a training set, which you will use as an argument in your forecasting function(s). You can also create a test set to use later.
4. Create different rolling origins to evaluate forecast accuracy using time series cross validation

Recap

5. Train model to each origin
6. Computer forecast accuracy, use the `accuracy()` function with the `fable` as the first argument and original data as the second.
7. Compare methods using point, prediction interval and distributional accuracy measure; a smaller error indicates higher accuracy.
8. Forecast using all data for the future using the best method.
9. Use residual diagnostic based on residuals of the best model.