

# Time Series Analysis & Forecasting Using R

The forecasters' toolbox-Evaluate  
forecasting performance

Bahman Rostami-Tabar

# Outline

- 1 Residual diagnostics
- 2 Evaluating point forecast accuracy
- 3 Time Series Cross Validation (TSCV)
- 4 Evaluating prediction interval accuracy
- 5 Lab session 5

# Outline

- 1 Residual diagnostics
- 2 Evaluating point forecast accuracy
- 3 Time Series Cross Validation (TSCV)
- 4 Evaluating prediction interval accuracy
- 5 Lab session 5

# 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:  $e_t = y_t - \hat{y}_{t|t-1}$ .

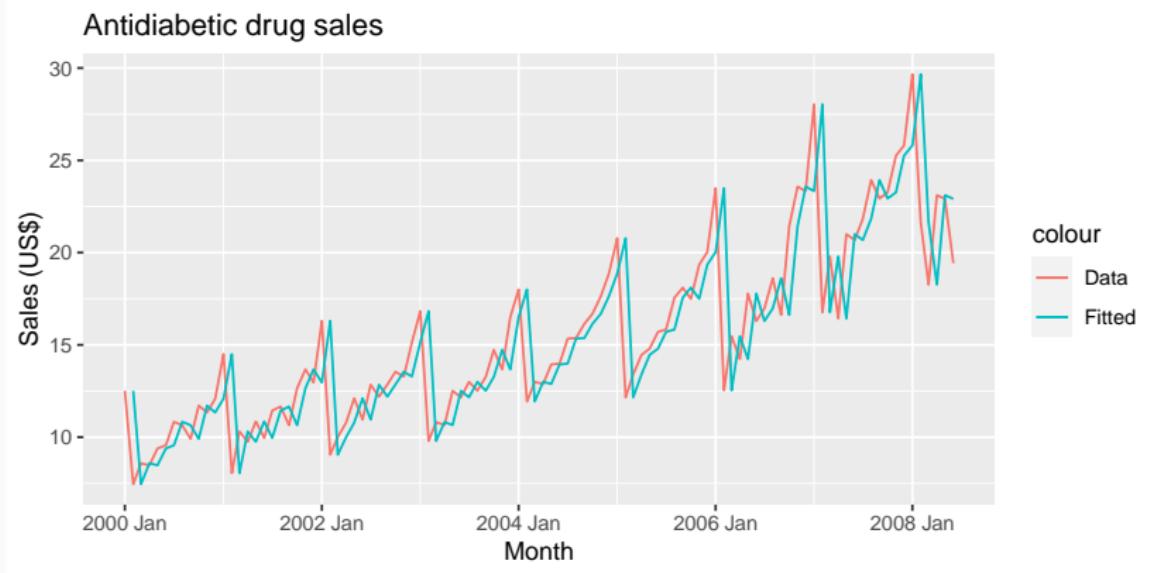
## Assumptions

- 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.

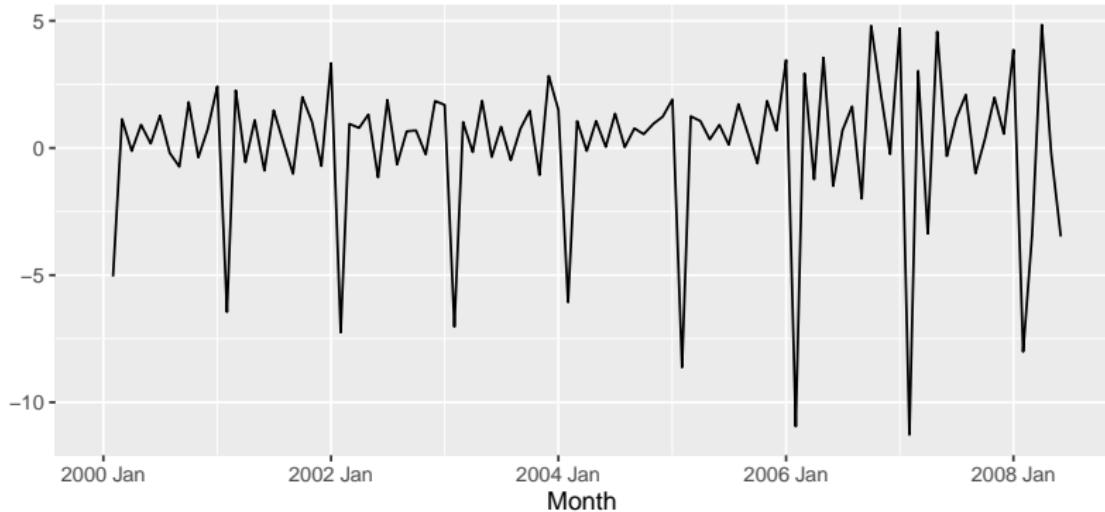
# Example: Antidiabetic drug sales



# Example: Antidiabetic drug sales

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

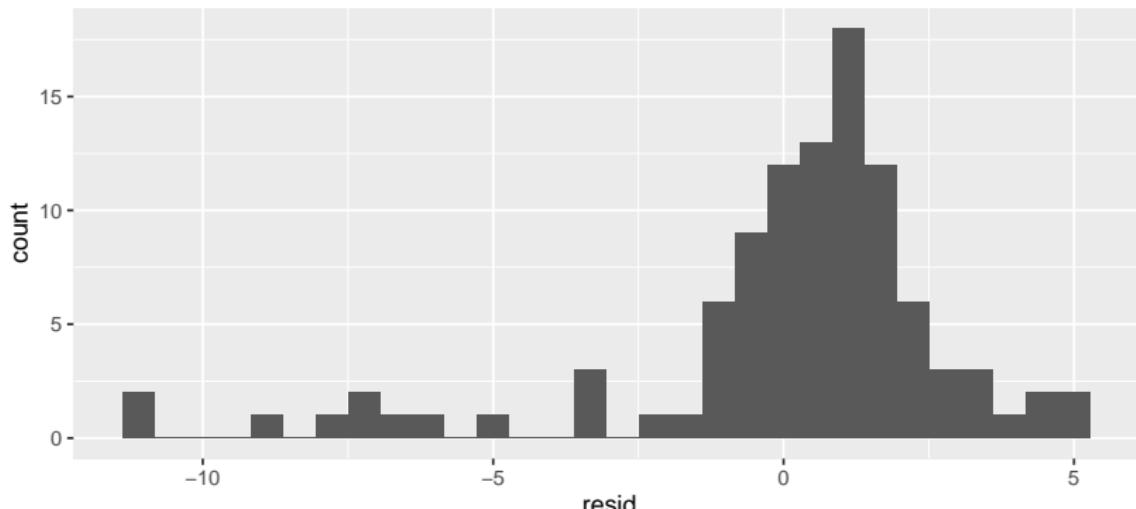
Residuals from naïve method



# Example: Antidiabetic drug sales

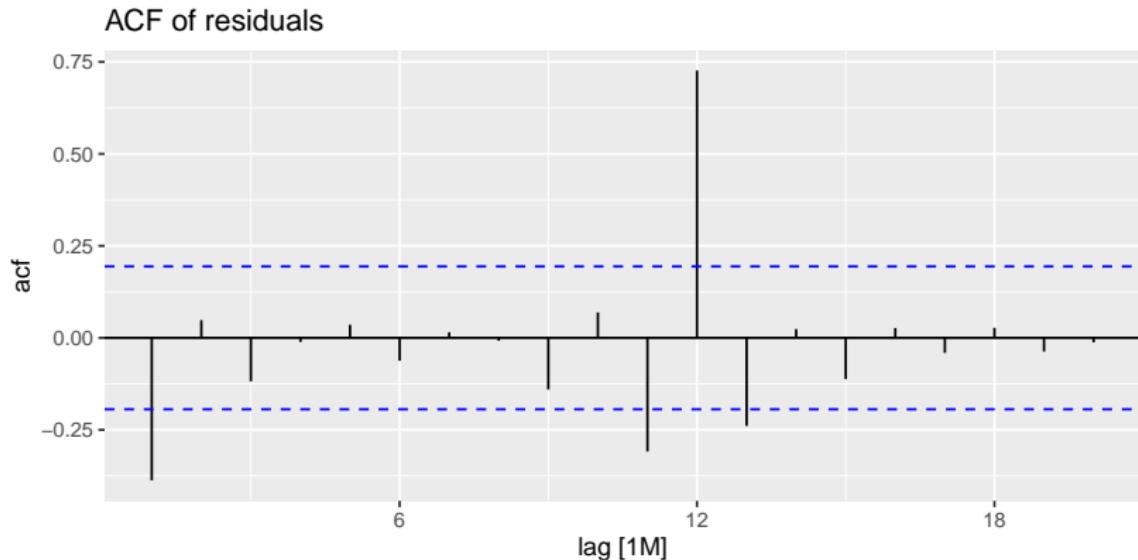
```
augment(fit) %>%
  ggplot(aes(x = .resid)) +
  geom_histogram(bins = 30) +
  ggtitle("Histogram of residuals")
```

Histogram of residuals



# Example: Antidiabetic drug sales

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

## 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.

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

# Portmanteau tests

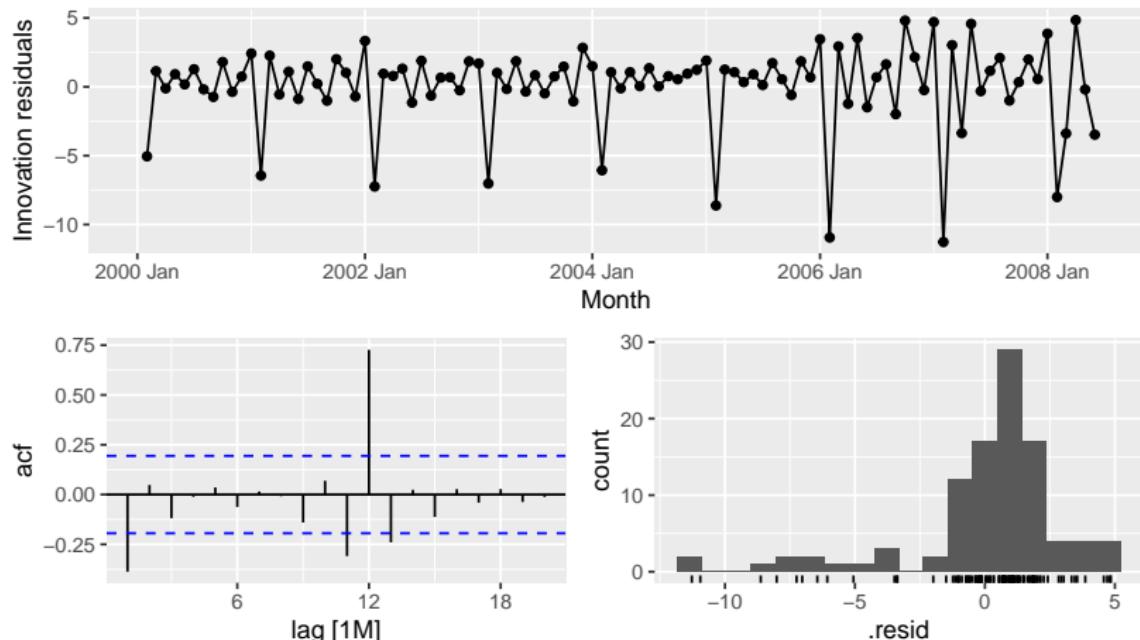
- If data are WN,  $Q^*$  has  $\chi^2$  distribution with  $(h - K)$  degrees of freedom where  $K$  = no. parameters in model.
- When applied to raw data, set  $K = 0$ .

```
augment(fit) %>% features(.resid, ljung_box, lag=10, dof=0)
```

```
## # A tibble: 1 x 3
##   .model      lb_stat lb_pvalue
##   <chr>        <dbl>     <dbl>
## 1 NAIVE(Cost)  20.7     0.0232
```

# gg\_tsresiduals function

```
fit %>%  
  gg_tsresiduals()
```



# Outline

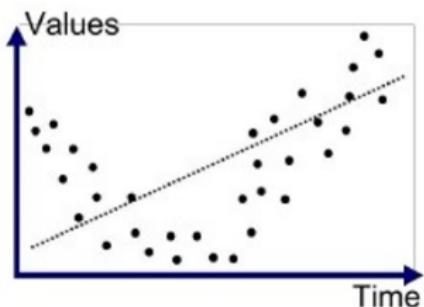
- 1 Residual diagnostics
- 2 Evaluating point forecast accuracy
- 3 Time Series Cross Validation (TSCV)
- 4 Evaluating prediction interval accuracy
- 5 Lab session 5

# Evaluating point forecast accuracy

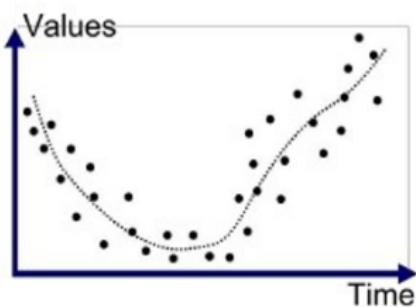
## Evaluate forecast accuracy

- Residual diagnostic is not a reliable indication of forecast accuracy
- 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

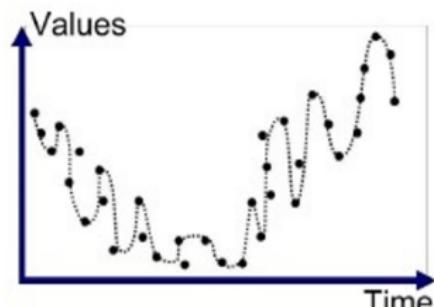
# Fitting



Underfitted



Good Fit/R robust



Overfitted

## Evaluate forecast accuracy

The accuracy of forecasts can only be determined by considering how well a model performs on new data that were not used when fitting the model

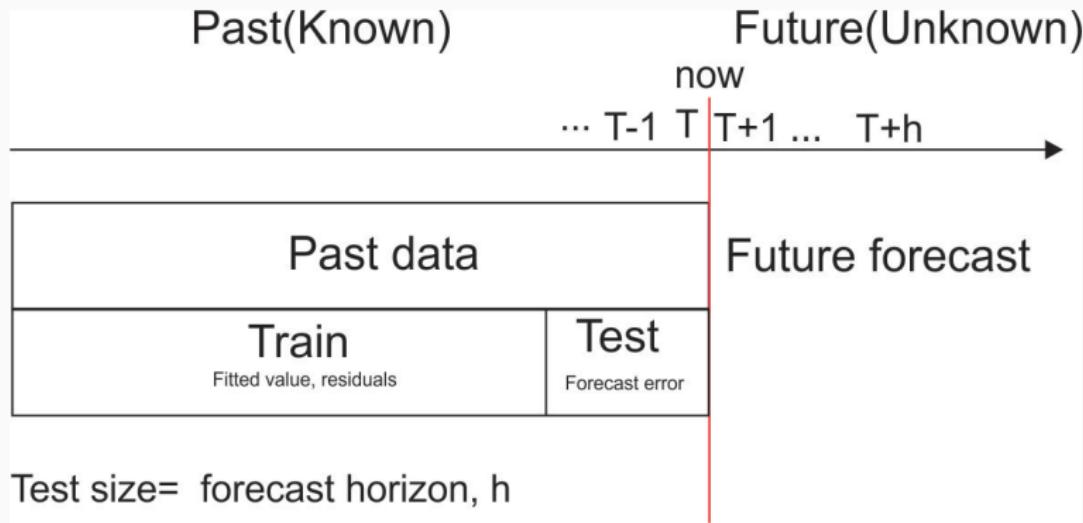
# Forecast accuracy evaluation using test sets

- We mimic the real life situation
- We pretend we don't know some part of data(new data)
- It must not be used for *any* aspect of model training
- Forecast accuracy is based only on the test set

# Outline

- 1 Residual diagnostics**
- 2 Evaluating point forecast accuracy**
- 3 Time Series Cross Validation (TSCV)**
- 4 Evaluating prediction interval accuracy**
- 5 Lab session 5**

# Training and test series



# Split the data

Use functions in `dplyr` and `lubridate` such as  
`filter`, `filter_index`, `slice`

```
# Filter the year of interest
antidiabetic_drug_sale %>%
  filter_index("2006"~.)
```

```
## # A tsibble: 30 x 2 [1M]
##       Month   Cost
##       <mth> <dbl>
## 1 2006 Jan  23.5
## 2 2006 Feb  12.5
## 3 2006 Mar  15.5
## 4 2006 Apr  14.2
```

## 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

$y_{T+h}$  =  $(T + h)$ th observation,  $h = 1, \dots, H$

$\hat{y}_{T+h|T}$  = its forecast based on data up to time  $T$ .

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

MAE =  $\text{mean}(|e_{T+h}|)$

MSE =  $\text{mean}(e_{T+h}^2)$

RMSE =  $\sqrt{\text{mean}(e_{T+h}^2)}$

MAPE =  $100\text{mean}(|e_{T+h}|/|y_{T+h}|)$

# Measures of forecast accuracy

$y_{T+h}$  =  $(T + h)$ th observation,  $h = 1, \dots, H$

$\hat{y}_{T+h|T}$  = its forecast based on data up to time  $T$ .

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

MAE =  $\text{mean}(|e_{T+h}|)$

MSE =  $\text{mean}(e_{T+h}^2)$

RMSE =  $\sqrt{\text{mean}(e_{T+h}^2)}$

MAPE =  $100\text{mean}(|e_{T+h}|/|y_{T+h}|)$

- 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  $\{y_t\}$ .

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  $\{y_t\}$ .

For seasonal time series,

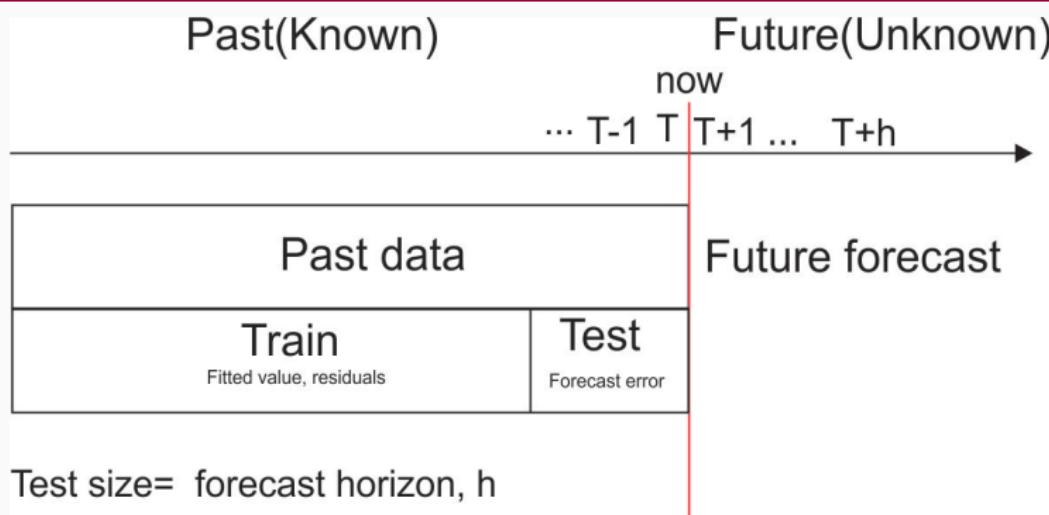
$$Q = (T - m)^{-1} \sum_{t=m+1}^T |y_t - y_{t-m}|$$

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

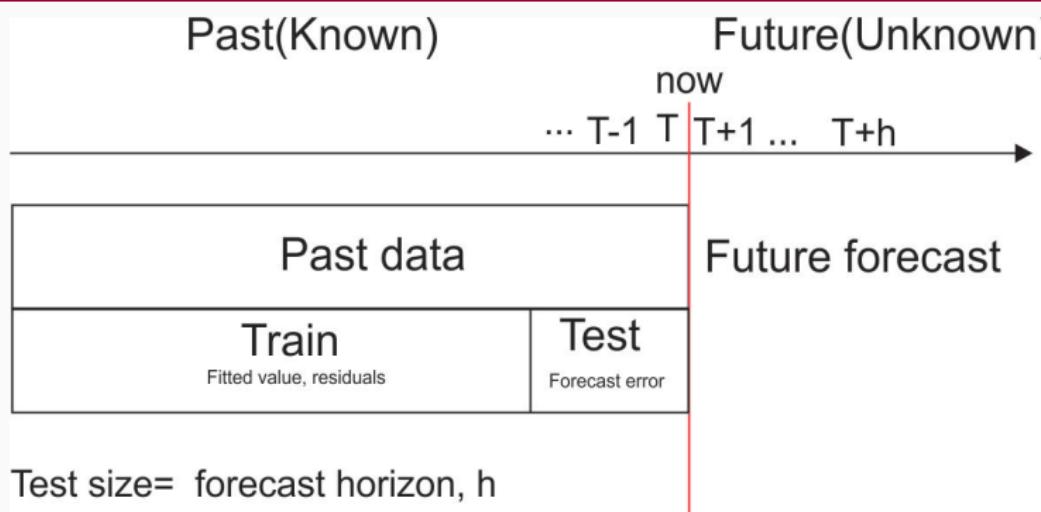
## Poll: true or false?

- 1 Good point forecast models 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 Always choose the model with the best forecast accuracy as measured on the test set.

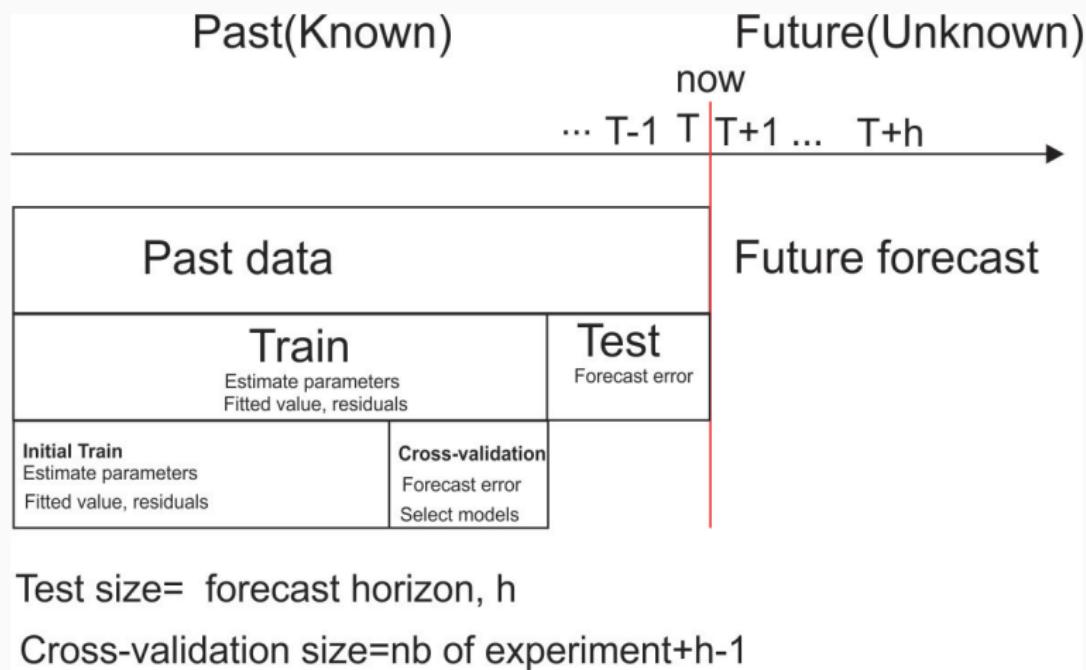
# Issue with train/test split without time series cross-validation



# Issue with train/test split without time series cross-validation

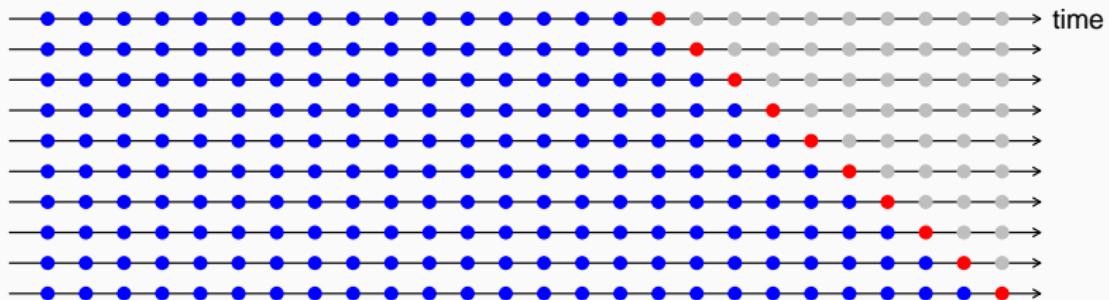


# Time series cross-validation



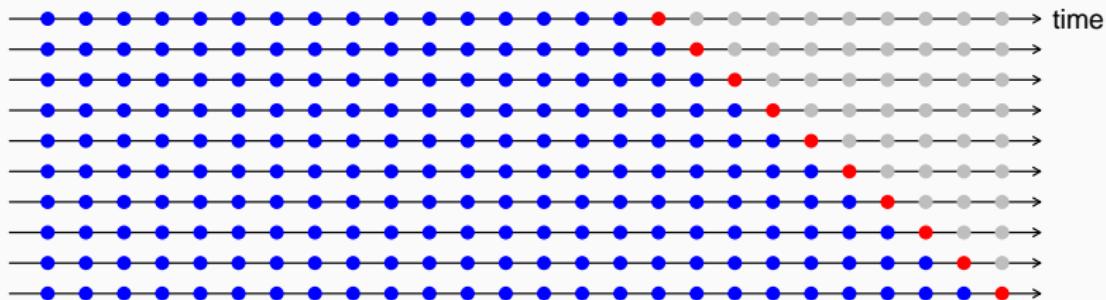
# Time series cross-validation

## 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 %>% filter_index(as.character(
  max(antidiabetic_drug_sale$`id`)))

train <- antidiabetic_drug_sale %>%
  filter_index(. ~ as.character(max(antidiabetic_drug_sale$`id`)))
drug_sale_tcsv <- train %>% slice(1:(n()-forecast_horizon))
stretch_tsibble(.init = 24, .step = 1)

## # A tsibble: 2,209 x 3 [1M]
## # Key:     .id [47]
##       Month  Cost   .id
##       <mth> <dbl> <int>
## 1 2000 Jan 12.5     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: 47 x 2
## # Key:     .id [47]
##     .id    snaive
##   <int>  <model>
## 1     1 <SNAIVE>
## 2     2 <SNAIVE>
## 3     3 <SNAIVE>
## 4     4 <SNAIVE>
## # ... with 43 more rows
```

# Time series cross-validation

Produce 8 step ahead forecasts from all models.

```
drug_fc_tr <- drug_fit_tr %>%
  forecast(h=forecast_horizon) %>%
  group_by(.id,.model) %>%
  mutate(h=row_number()) %>%
  as_fable(response="Cost",distribution="Cost") |>
  ungroup()
```

```
## # A fable: 564 x 6 [1M]
## # Key:      .id, .model [47]
##     .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
```

# Time series cross-validation

```
# Cross-validated  
drug_fc_tr %>%  
  accuracy(antidiabetic_drug_sale)
```

# Outline

- 1 Residual diagnostics
- 2 Evaluating point forecast accuracy
- 3 Time Series Cross Validation (TSCV)
- 4 Evaluating prediction interval accuracy
- 5 Lab session 5

# Winkler score

Winkler proposed a scoring method to enable comparisons between prediction intervals:

- it takes account of both coverage and width of the intervals.

## Winkler score

$$W(l_t, u_t, y_t) = \begin{cases} u_t - l_t & \text{if } l_t < y_t < u_t \\ (u_t - l_t) + \frac{2}{\alpha}(l_t - y_t) & \text{if } y_t < l_t \\ (u_t - l_t) + \frac{2}{\alpha}(y_t - u_t) & \text{if } y_t > u_t \end{cases}$$

# Prediction interval accuracy

```
# Compute interval accuracy
drug_fc_tr %>%
  accuracy(antidiabetic_drug_sale,
  measures = interval_accuracy_measures) %>%
  mutate(Method = paste(.model, "method")) %>%
  select(Method, winkler) %>%
  gt::gt() %>%
  gt::as_latex()
```

Method	winkler
snaive method	7.596448

# Outline

- 1 Residual diagnostics**
- 2 Evaluating point forecast accuracy**
- 3 Time Series Cross Validation (TSCV)**
- 4 Evaluating prediction interval accuracy**
- 5 Lab session 5**

# Lab session 5

# Recap

- 1 First, import your data and prepare them using `tsibble` function.
- 2 Visualise and see whether your series contains key features
- 3 Determine how much of your data you want to allocate to training, and how much to testing; the sets should not overlap.
- 4 Subset the data to create a training set, which you will use as an argument in your forecasting function(s). Optionally, you can also create a test set to use later.
- 5 Compute forecasts of the training set using whichever forecasting function(s) you choose, and set `h` equal to the number of values you want to forecast.

# Recap

- 6 Use residual diagnostic based on residuals in the training set to see whether all information is captured by models.
- 7 Create different windows to evaluate forecast accuracy using time series cross validation
- 8 Train model to each window
- 9 To calculate the forecast accuracy, use the `accuracy()` function with the `fable` as the first argument and original data as the second.
- 10 Pick a measure in the output to evaluate the forecast(s); a smaller error indicates higher accuracy.
- 11 Finally, produce forecast using the selected approach for real future.