Time Series Analysis in Health Research

Lecture 2: Forecasting Tools & Forecasting Methods

Erfan Hoque

Dept. of Community Health & Epidemiology, University of Saskatchewan

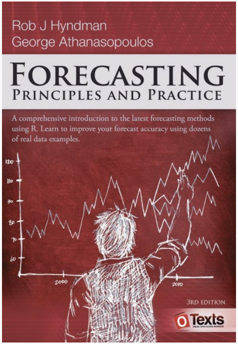
SEA 24th Annual Symposium Workshop, September 26, 2025

Schedule for this workshop

	Topic
Lecture 1	Introduction to forecasting, graphics
Lecture 2	Time Series toolbox and Forecasting Models
Lecture 3	Time Series Regression Models
Lecture 4	Hands-on session in R
Lecture 5	Advanced Forecasting Methods

Resources

Organization and presentation of material in these lectures will largely pull from



- Free and online (`https://otexts.com/fpp3/`)
- Data sets in associated R packages, R code for examples

Forcaster’s toolbox

We will discuss some general tools that are useful for forecasting.

- We will describe some benchmark and traditional forecasting methods, procedures for checking whether a forecasting method has adequately utilised the available information, and methods for evaluating forecast accuracy.

A forecasting workflow

The process of producing forecasts can be split up into a few fundamental steps.

1. Preparing data
2. Data visualisation
3. Specifying a model
4. Model estimation
5. Accuracy & performance evaluation
6. Producing forecasts

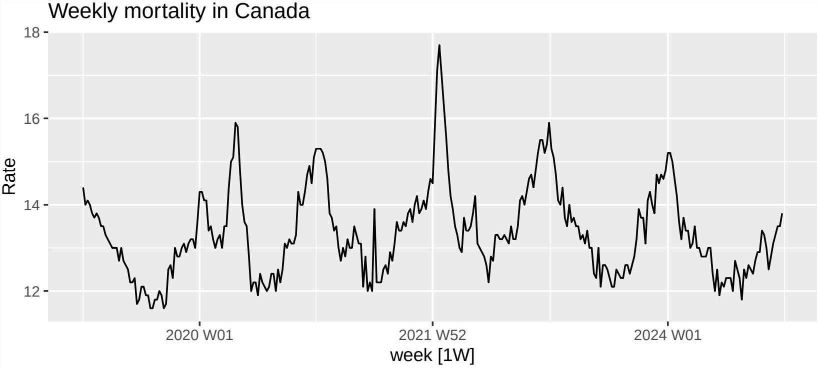
Data preparation (tidy)

```r
mortality_ts_canada <- mortality_ts %>%
  filter(GEO=="Canada" & Sex=="Both sexes")
mortality_ts_canada %>%
  head(10)
```

```
## # A tsibble: 10 x 5 [1W]
## # Key:       GEO, Sex [1]
##    Date       GEO    Sex         Rate     week
##    <chr>      <chr>  <chr>      <dbl>   <week>
##  1 2019-01-05 Canada Both sexes  14.4 2019 W01
##  2 2019-01-12 Canada Both sexes  14   2019 W02
##  3 2019-01-19 Canada Both sexes  14.1 2019 W03
##  4 2019-01-26 Canada Both sexes  14   2019 W04
##  5 2019-02-02 Canada Both sexes  13.8 2019 W05
##  6 2019-02-09 Canada Both sexes  13.7 2019 W06
```

Data visualisation

```
mortality_ts_canada %>%
  autoplot(Rate) +
    labs(title = "Weekly mortality in Canada", y = "Rate")
```



Model estimation

The `model()` function trains models to data.

```
fit <- mortality_ts_canada %>%
  model(trend_model = TSLM(Rate ~ trend()))
fit
```

```
## # A mable: 1 x 3
## # Key:     GEO, Sex [1]
##   GEO    Sex         trend_model
##   <chr>  <chr>           <model>
## 1 Canada Both sexes       <TSLM>
```

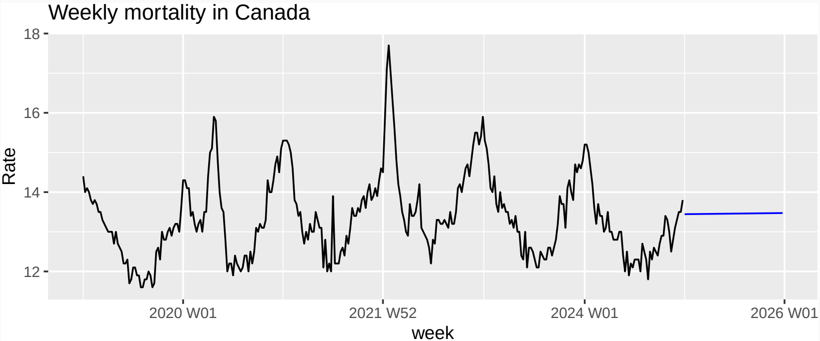
Producing forecasts

```r
fit %>% forecast(h = "1 years")
```

```
## # A fable: 52 x 6 [1W]
## # Key:     GEO, Sex, .model [1]
##    GEO    Sex        .model          week         Rate .mean
##    <chr>  <chr>      <chr>         <week>       <dist> <dbl>
##  1 Canada Both sexes trend_model 2025 W01 N(13, 1.1)  13.4
##  2 Canada Both sexes trend_model 2025 W02 N(13, 1.1)  13.4
##  3 Canada Both sexes trend_model 2025 W03 N(13, 1.1)  13.4
##  4 Canada Both sexes trend_model 2025 W04 N(13, 1.1)  13.4
##  5 Canada Both sexes trend_model 2025 W05 N(13, 1.1)  13.4
##  6 Canada Both sexes trend_model 2025 W06 N(13, 1.1)  13.4
##  7 Canada Both sexes trend_model 2025 W07 N(13, 1.1)  13.4
##  8 Canada Both sexes trend_model 2025 W08 N(13, 1.1)  13.4
##  9 Canada Both sexes trend_model 2025 W09 N(13, 1.1)  13.4
```

Visualising forecasts

```r
fit %>% forecast(h = "1 years") %>%
  autoplot(mortality_ts_canada, level = NULL) +
    labs(title = "Weekly mortality in Canada", y = "Rate")
```

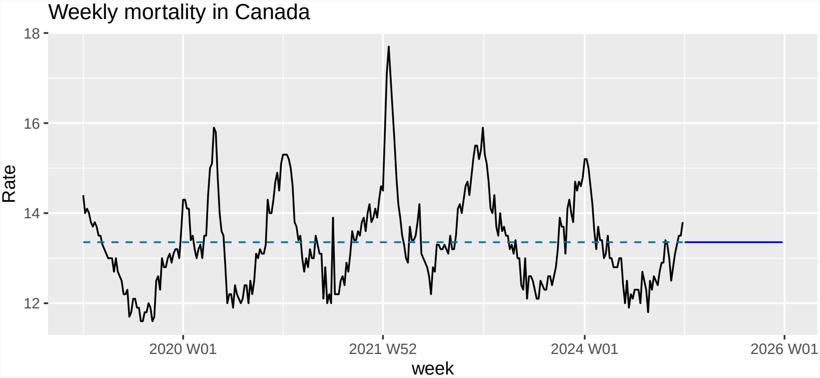


Some simple forecasting methods

Some simple forecasting methods

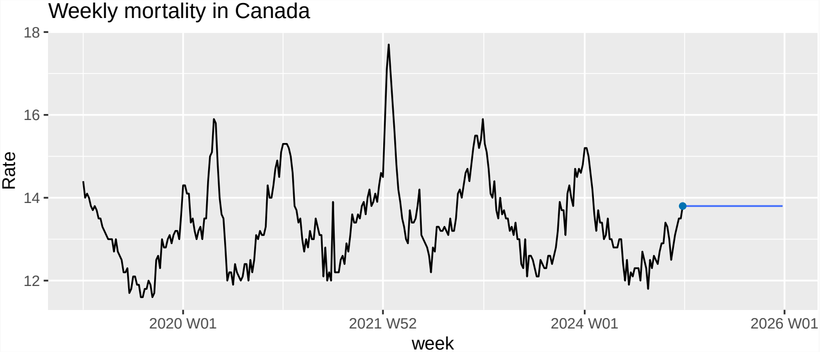
`MEAN(y)`: Average method

- Forecast of all future values is equal to mean of historical data $\{y_1, \dots, y_T\}$.
- Forecasts: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$



`NAIVE(y)`: Naïve method

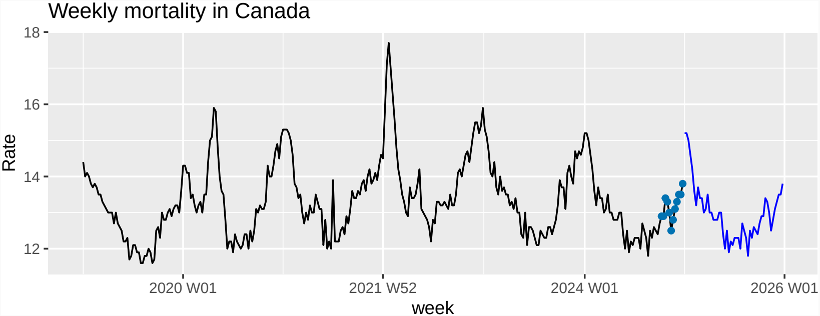
- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{T+h|T} = y_T$.
- Consequence of efficient market hypothesis.



Some simple forecasting methods

`SNAIVE(y ~ lag(m))`: Seasonal naïve method

- Forecasts equal to last value from same season.
- Forecasts: $\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$, where $m =$ seasonal period and k is the integer part of $(h-1)/m$.



Model fitting

The `model()` function trains models to data.

```
mortality_fit <-  mortality_ts_canada %>%
  model(
    Seasonal_naive = SNAIVE(Rate),
    Naive = NAIVE(Rate),
    Mean = MEAN(Rate)
  )
```

```
## # A mable: 1 x 5
## # Key:     GEO, Sex [1]
##   GEO    Sex          Seasonal_naive   Naive    Mean
##   <chr>  <chr>               <model> <model> <model>
## 1 Canada Both sexes         <SNAIVE> <NAIVE>  <MEAN>
```

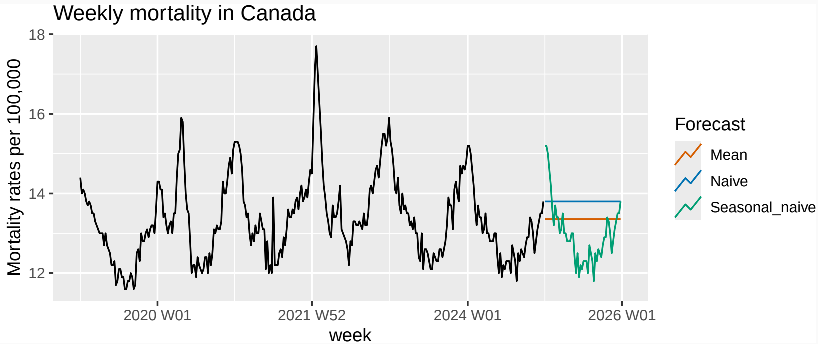
Producing forecasts

```r
mortality_fc <- mortality_fit %>%
  forecast(h = "1 years")
```

```
## # A fable: 156 x 6 [1W]
## # Key:     GEO, Sex, .model [3]
##   GEO    Sex        .model                 week       Rate .mean
##   <chr>  <chr>      <chr>                <week>     <dist> <dbl>
## 1 Canada Both sexes Seasonal_naive 2025 W01 N(15, 0.9)  15.2
## 2 Canada Both sexes Seasonal_naive 2025 W02 N(15, 0.9)  15.2
## 3 Canada Both sexes Seasonal_naive 2025 W03 N(15, 0.9)    15
## 4 Canada Both sexes Seasonal_naive 2025 W04 N(15, 0.9)  14.6
## # i 152 more rows
```

Visualising forecasts

```r
mortality_fc %>%
  autoplot(mortality_ts_canada, level = NULL) +
  labs(title = "Weekly mortality in Canada",
       y = "Mortality rates per 100,000") +
  guides(colour = guide_legend(title = "Forecast"))
```



Residual diagnostics

Fitted values

- $\hat{y}_{t|t-1}$ is the forecast of y_t based on observations $y_1, \ldots, y_{t-1}$ (we call these "fitted values").
- Sometimes we write it as: $\hat{y}_t \equiv \hat{y}_{t|t-1}$.
- Often not true forecasts since parameters are estimated on all data.

For example:

- $\hat{y}_t = \bar{y}$ for average method.

Forecasting residuals

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

Assumptions:

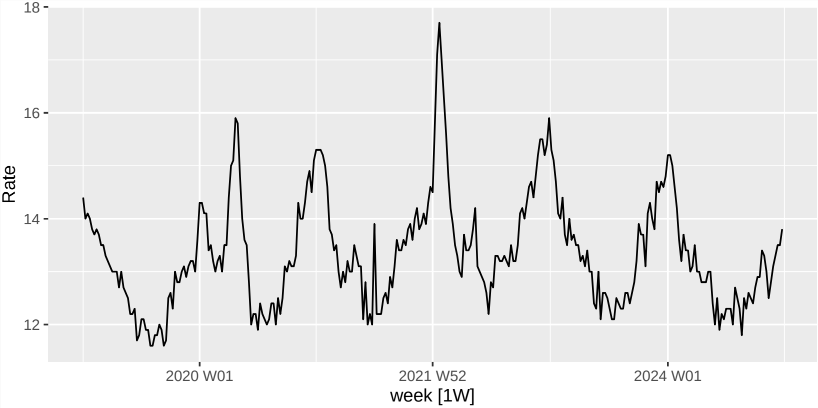
1. e_t's are uncorrelated. If they aren't, then information left in residuals that should be used in computing forecasts.
2. e_t's have mean zero. If they don't, then forecasts are biased.

Properties (for distributions & prediction intervals):

3. e_t's have constant variance.
4. e_t's are normally distributed.

Mortality rate in Canada

```r
mortality_ts_canada %>% autoplot(Rate)
```



Mortality rate in Canada

```
fit <- mortality_ts_canada %>% model(NAIVE(Rate))
augment(fit)
```

```
## # A tsibble: 313 x 8 [1W]
## # Key:       GEO, Sex, .model [1]
##    GEO    Sex   .model      week  Rate .fitted .resid .innov
##    <chr>  <chr> <chr>     <week> <dbl>   <dbl>  <dbl>  <dbl>
##  1 Canada Both ~ NAIVE~ 2019 W01  14.4    NA   NA     NA
##  2 Canada Both ~ NAIVE~ 2019 W02  14      14.4 -0.400 -0.400
##  3 Canada Both ~ NAIVE~ 2019 W03  14.1    14    0.100  0.100
##  4 Canada Both ~ NAIVE~ 2019 W04  14      14.1 -0.100 -0.100
##  5 Canada Both ~ NAIVE~ 2019 W05  13.8    14   -0.200 -0.200
##  6 Canada Both ~ NAIVE~ 2019 W06  13.7    13.8 -0.100 -0.100
##  7 Canada Both ~ NAIVE~ 2019 W07  13.8    13.7  0.100  0.100
##  8 Canada Both ~ NAIVE~ 2019 W08  13.7    13.8 -0.100 -0.100
##  9 Canada Both ~ NAIVE~ 2019 W09  13.5    13.7 -0.200 -0.200
## 10 Canada Both ~ NAIVE~ 2019 W10  13.5    13.5  0      0
## # i 303 more rows
```

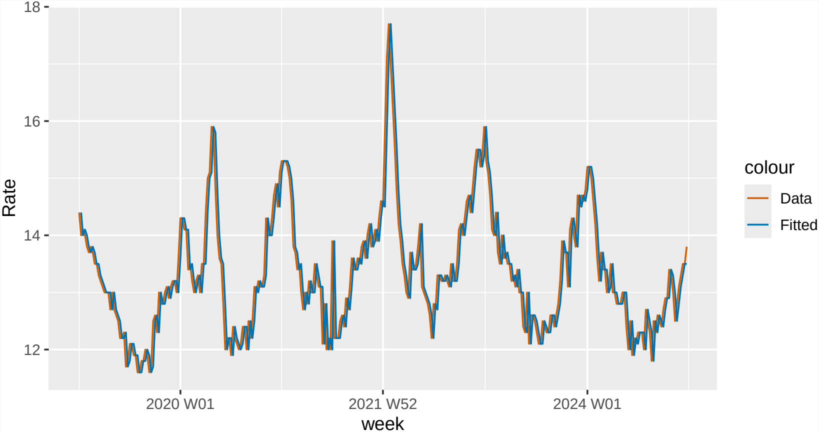
Naïve forecasts:

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

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

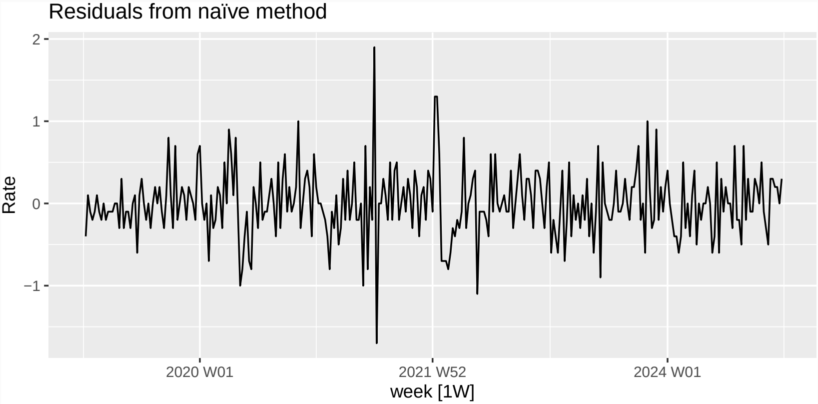
Mortality rate in Canada

```r
augment(fit) %>%
  ggplot(aes(x = week)) +
  geom_line(aes(y = Rate, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted"))
```



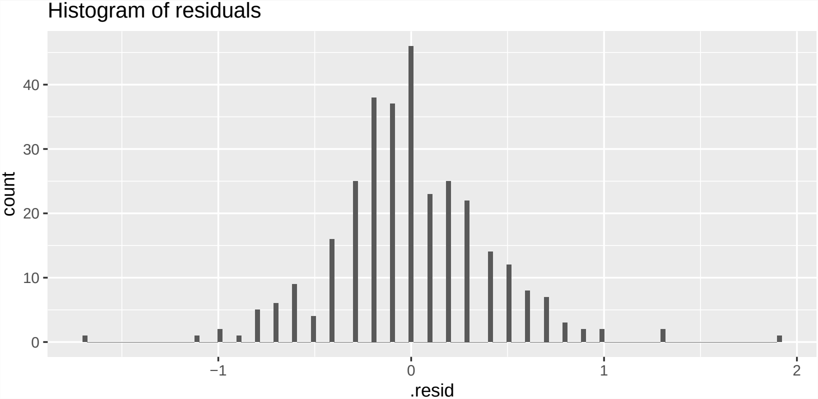
Mortality rate in Canada

```r
augment(fit) %>%
  autoplot(.resid) +
  labs(y = "Rate",
       title = "Residuals from naïve method")
```



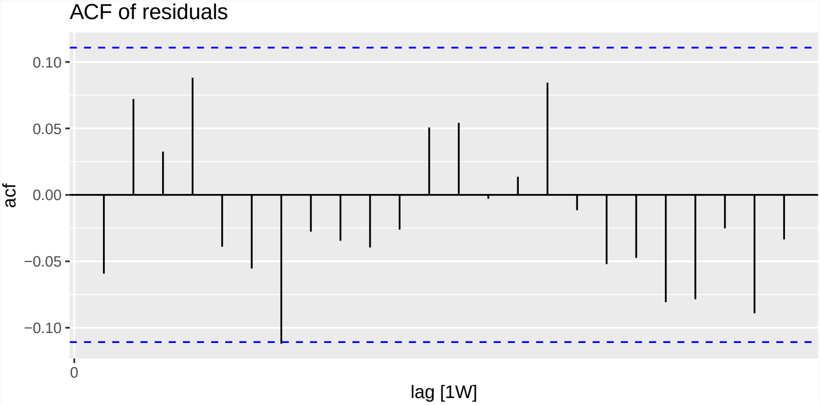
Mortality rate in Canada

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



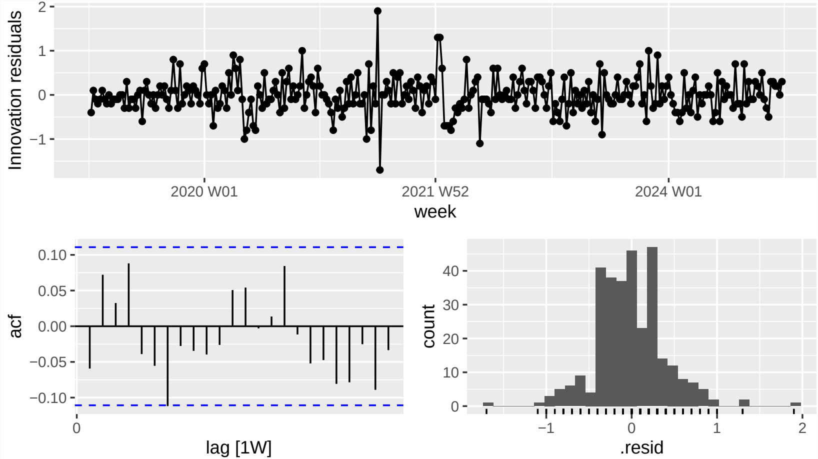
Mortality rate in Canada

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



gg_tsresiduals() function

```r
gg_tsresiduals(fit)
```



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 for autocorrelation

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

- Box-Pierce test
- Ljung-Box test

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

```
## # A tibble: 1 x 5
##   GEO    Sex         .model       lb_stat lb_pvalue
##   <chr>  <chr>       <chr>          <dbl>     <dbl>
## 1 Canada Both sexes NAIVE(Rate)    12.2     0.272
```

The results are not significant (i.e., the p-values are relatively large). Thus, we can conclude that the residuals are not distinguishable from a white noise series.

Evaluating forecast accuracy

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

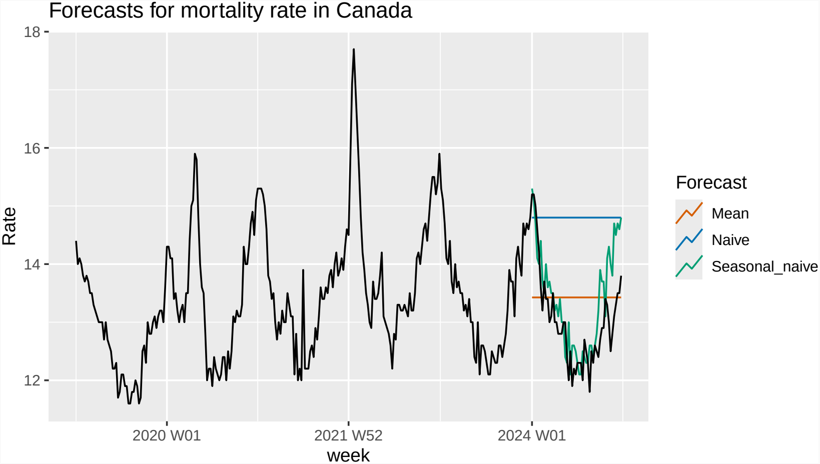
Forecast "error": the difference between an observed value and its forecast.

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

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

- 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

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

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

where Q is a stable measure of the scale of the time series y_t.

- MAE, MSE, RMSE are all scale dependent.
- MAPE, MASE 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

```r
train <- mortality_ts_canada %>%
  filter(year(week) <= 2023)

mortality_fit <- train %>%
  model(
    Mean = MEAN(Rate),
    Naive = NAIVE(Rate),
    Seasonal_naive = SNAIVE(Rate)
  )
mortality_fc <- mortality_fit %>%
  forecast(h = 52)
```

Measures of forecast accuracy

```r
fabletools::accuracy(mortality_fit)
```

```
## # A tibble: 3 x 6
##   .model         .type     RMSE   MAE  MAPE  MASE
##   <chr>          <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 Mean           Training 1.06  0.828  6.11 1.07
## 2 Naive          Training 0.418 0.301  2.22 0.388
## 3 Seasonal_naive Training 1.00  0.776  5.57 1
```

```r
fabletools::accuracy(mortality_fc, mortality_ts_canada)
```

```
## # A tibble: 3 x 6
##   .model         .type  RMSE   MAE  MAPE  MASE
##   <chr>          <chr> <dbl> <dbl> <dbl> <dbl>
## 1 Mean           Test  0.891 0.742  5.78 0.956
## 2 Naive          Test  1.97  1.84  14.5  2.37
## 3 Seasonal_naive Test  0.668 0.521  4.03 0.672
```

ARIMA models

ARIMA models

- ARIMA models provide another approach to time series forecasting.
- Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting.
- While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

ARIMA models

AR: autoregressive (lagged observations as inputs)
I: integrated (differencing to make series stationary)
MA: moving average (lagged errors as inputs)

Stationarity and differencing

Stationarity

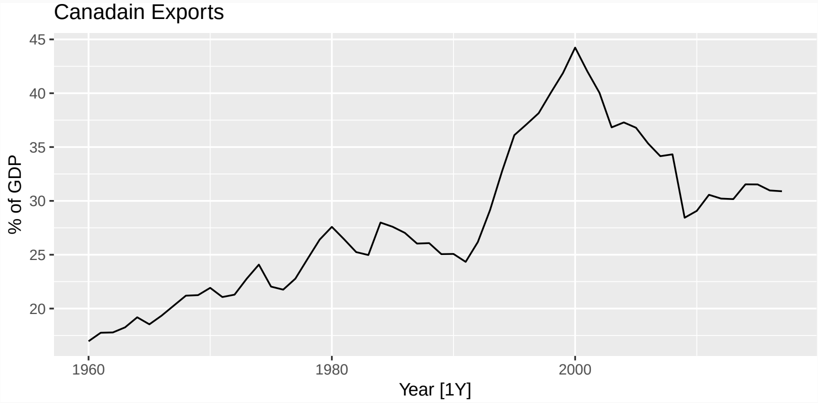
Definition
If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \ldots, y_{t+s})$ does not depend on t (not a function of time).

A **stationary series** is:

- roughly horizontal
- constant mean and variance (not a function of time)
- no patterns predictable in the long-term

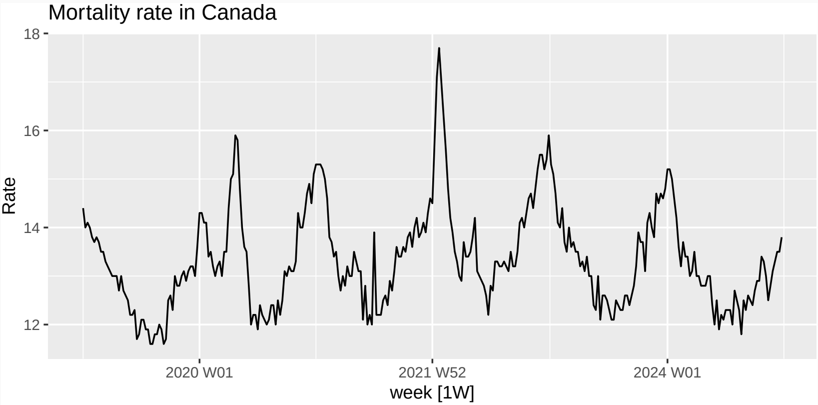
Stationary?

```
global_economy %>%
  filter(Country == "Canada") %>%
  autoplot(Exports) +
  labs(y = "% of GDP", title = "Canadain Exports")
```



Stationary?

```
mortality_ts_canada %>%
  autoplot(Rate) +
  labs(y = "Rate", title = "Mortality rate in Canada")
```



Stationarity and Non-stationarity

For ARIMA modelling, we need to **stabilize the mean**.

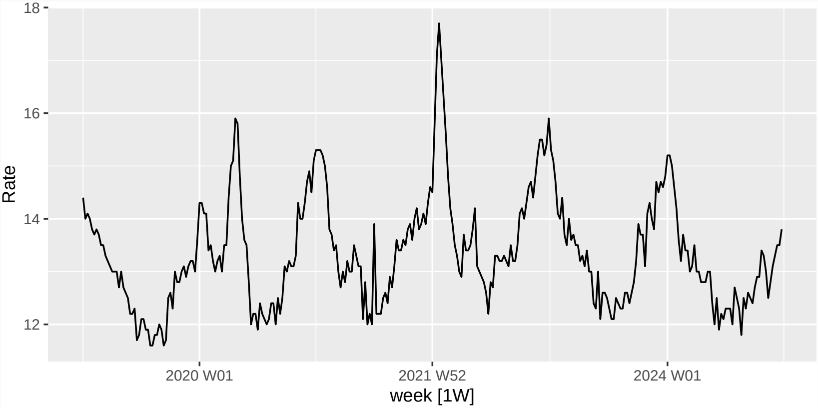
Transformations also help to **stabilize the variance**.

Identifying non-stationary series:

- time plot.
- The ACF of stationary data drops to zero relatively quickly
- The ACF of non-stationary data decreases slowly.
- For non-stationary data, the value of ρ_1 is often large and positive.

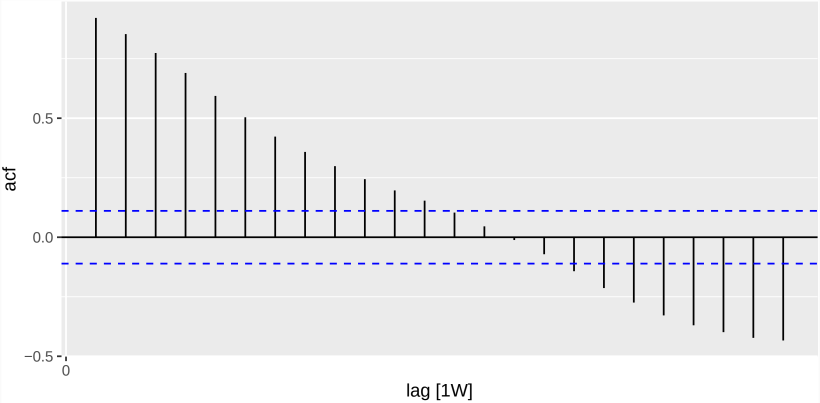
Example: Mortality rate in Canada

```r
mortality_ts_canada %>%
  autoplot(Rate) +
  labs(y = "Rate")
```



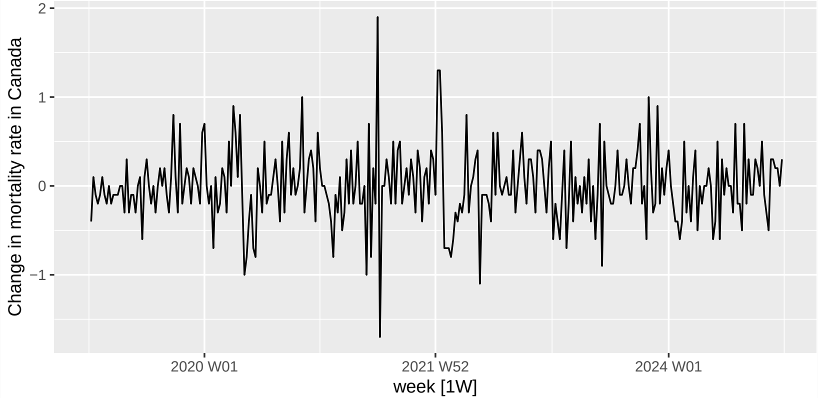
Example: Mortality rate in Canada

```r
mortality_ts_canada %>% ACF(Rate) %>% autoplot()
```



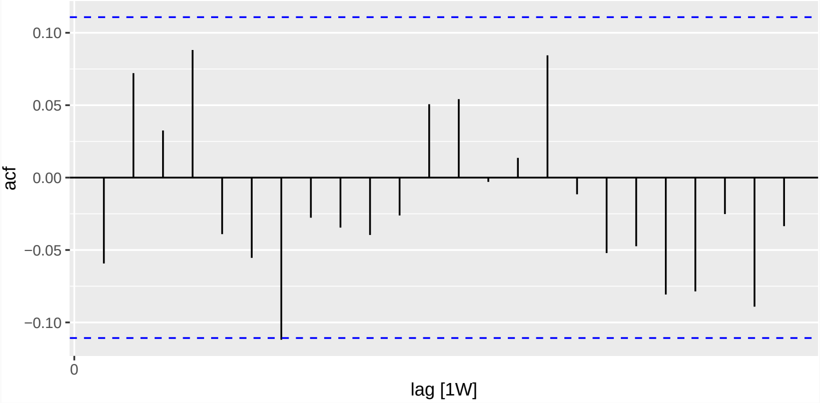
Example: Mortality rate in Canada

```r
mortality_ts_canada %>%
  autoplot(difference(Rate)) +
  labs(y = "Change in mortality rate in Canada")
```



Example: Mortality rate in Canada

```r
mortality_ts_canada %>% ACF(difference(Rate)) %>% autoplot()
```



Differencing

- Differencing helps to **stabilize the mean** (this shows one way to make a non-stationary time series stationary).
- The differenced series is the *change* between each observation in the original series: $y'_t = y_t - y_{t-1}$.
- The differenced series will have only $T - 1$ values since it is not possible to calculate a difference y'_1 for the first observation.

Second-order differencing

Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time:

$$y''_t = y'_t - y'_{t-1}$$

- y''_t will have $T - 2$ values.
- In practice, it is almost never necessary to go beyond second-order differences.

Seasonal differencing

A seasonal difference is the difference between an observation and the corresponding observation from the previous year. Seasonal differencing is used to remove seasonal patterns.

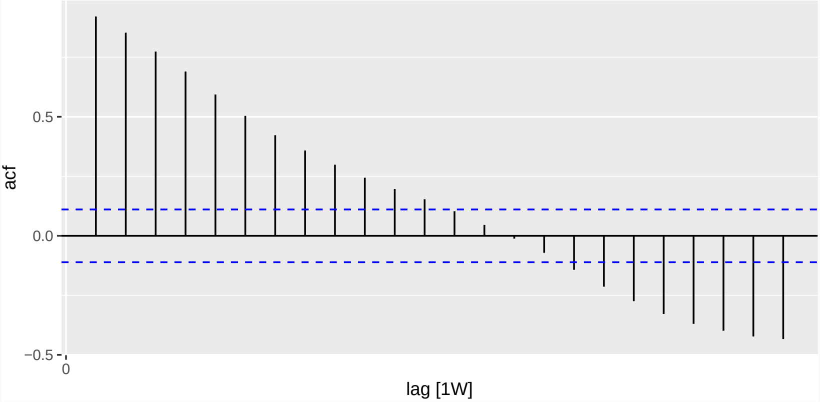
$$y'_t = y_t - y_{t-m}$$

where $m =$ number of seasons.

- For monthly data $m = 12$.
- For quarterly data $m = 4$.
- For weekly data $m = 52$.

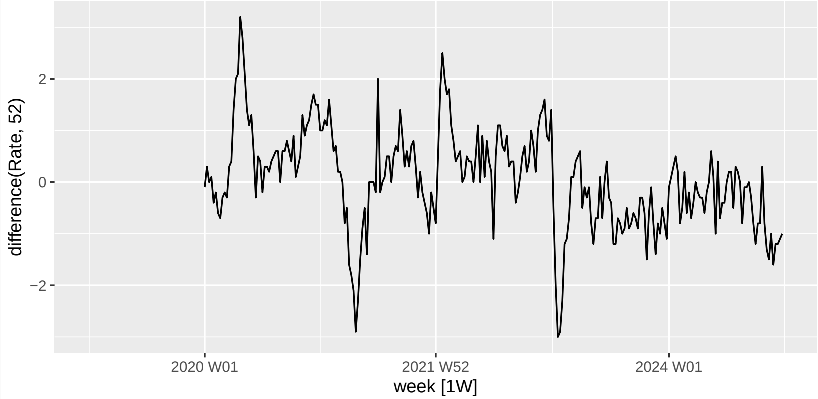
Mortality rate in Canada

```
mortality_ts_canada %>% ACF(Rate) %>% autoplot()
```



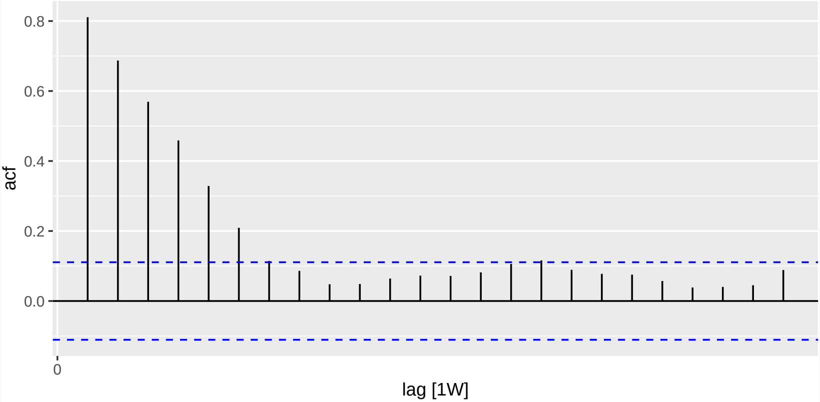
Mortality rate in Canada

```
mortality_ts_canada %>%
  autoplot(difference(Rate, 52))
```



Mortality rate in Canada

```
mortality_ts_canada %>%
  ACF(difference(Rate, 52)) %>% autoplot()
```



ARIMA models

Autoregressive Integrated Moving Average models (ARIMA)

ARIMA(p, d, q) model

AR: $p =$ order of the autoregressive part

I: $d =$ degree of first differencing involved

MA: $q =$ order of the moving average part.

- White noise model: ARIMA(0,0,0)
- Random walk: ARIMA(0,1,0) with no constant
- AR(p): ARIMA(p,0,0)
- MA(q): ARIMA(0,0,q)

Autoregressive (AR) models

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an AR model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself.

Autoregressive (AR) models:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + a_t$$

where a_t ($a_t \sim N(0, \sigma_a^2)$) is white noise. This is a multiple regression with **lagged values** of y_t as predictors. We refer to this as an AR(p) model, an autoregressive model of order p.

Moving Average (MA) models:

$$y_t = \mu + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}$$

where a_t is white noise and $a_t \overset{\text{iid}}{\sim} N(0, \sigma_a^2)$ a_t's are independent normally distributed with mean zero and constant variance σ_a^2.

This is a multiple regression with **past *errors*** as predictors. We refer to this as an MA(q) model, a moving average model of order q.

Don't confuse this with moving average smoothing!

Automatic modelling procedure with `ARIMA()`

1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. Use `ARIMA` to automatically select a model.
4. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
5. Once the residuals look like white noise, calculate forecasts.

Seasonal ARIMA models

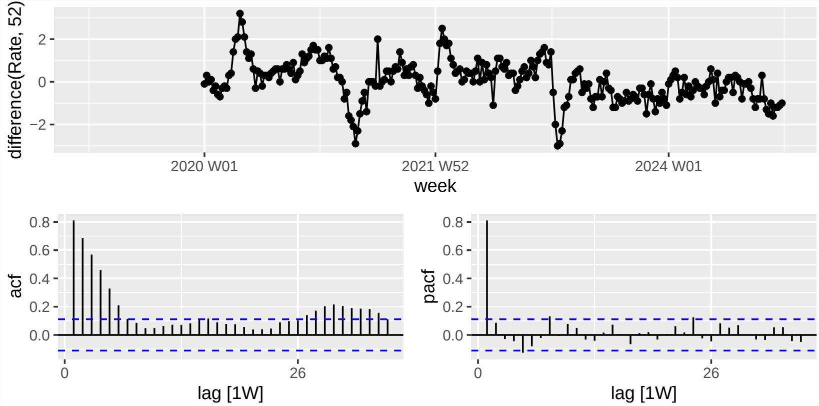
Seasonal ARIMA models

ARIMA	$\underbrace{(p, d, q)}$	$\underbrace{(P, D, Q)_s}$
	$\uparrow$ Non-seasonal part of the model	$\uparrow$ Seasonal part of of the model

where $s =$ number of observations per year.

Example: Mortality rate in Canada

```
mortality_ts_canada %>% gg_tsdisplay(difference(Rate,52),
                    lag_max = 36, plot_type = 'partial')
```



Mortality rate in Canada

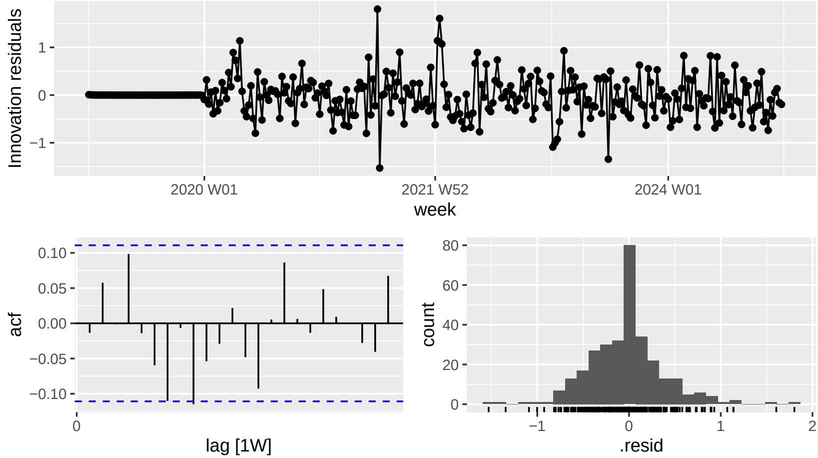
```r
fit <- mortality_ts_canada |>
  model(ARIMA(Rate))
report(fit)
```

```
## Series: Rate
## Model: ARIMA(1,1,2)(1,1,0)[52]
##
## Coefficients:
##          ar1      ma1    ma2     sar1
##       0.8391  -1.0746  0.100  -0.5548
## s.e.  0.0486   0.0748  0.069   0.0502
##
## sigma^2 estimated as 0.2067:  log likelihood=-172
## AIC=354   AICc=354   BIC=372
```

It's SARIMA(1,1,2)(1,1,0)[52] model.

Mortality rate in Canada

```
gg_tsresiduals(fit)
```



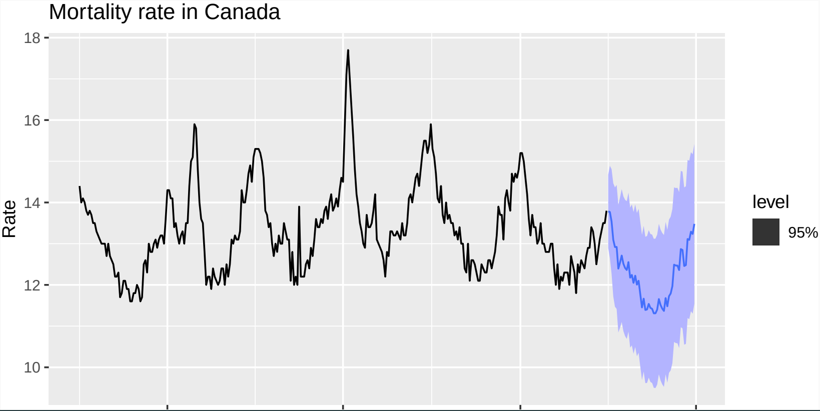
Mortality rate in Canada

```r
augment(fit) %>%
  features(.innov, ljung_box, lag = 36, dof=4)
```

```
## # A tibble: 1 x 5
##   GEO    Sex         .model      lb_stat lb_pvalue
##   <chr>  <chr>       <chr>         <dbl>     <dbl>
## 1 Canada Both sexes ARIMA(Rate)    34.2     0.363
```

Mortality rate in Canada

```r
fit %>% forecast(h=52) %>%
  autoplot(mortality_ts_canada, level=95) +
  labs(y = "Rate", title = "Mortality rate in Canada")
```



Mortality rate in Canada (Training and test sets)

Training data: 2019 to 2023 and Test data: 2024

```
train <- mortality_ts_canada %>% filter(year(week) <= 2023)
fit <- train %>% model(ARIMA(Rate))

fit %>% forecast(h = "1 years") %>%
  fabletools::accuracy(mortality_ts_canada) %>%
  arrange(.model) %>%
  select(.model, .type, RMSE, MAE, MAPE, MASE)
```

```
## # A tibble: 1 x 6
##   .model      .type  RMSE   MAE  MAPE  MASE
##   <chr>       <chr> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(Rate) Test  0.867 0.738  5.66 0.951
```

Exponential Smoothing

Simple Exponential Smoothing

Time series $y_1, y_2, \ldots, y_t$.

Naive forecasts

$$\hat{y}_{t+h|t} = y_t$$

Mean forecasts

$$\hat{y}_{t+h|t} = \frac{1}{T}\sum_{t=1}^{T} z_t$$

- Want something in between these methods.
- Most recent data should have more weight.

For correlated data it is more appropriate to give more weight to the most recent observations and less to the observations in the distant past.

In simple exponential smoothing the more recent observation are given relatively more weight compare to older observations (weights decrease exponentially).

Simple Exponential Smoothing

Forecast equation
$\hat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots$

where $0 \leq \alpha \leq 1$ and called called the smoothing constant.

- This method is more suitable for forecasting data with no clear trend or seasonal pattern.

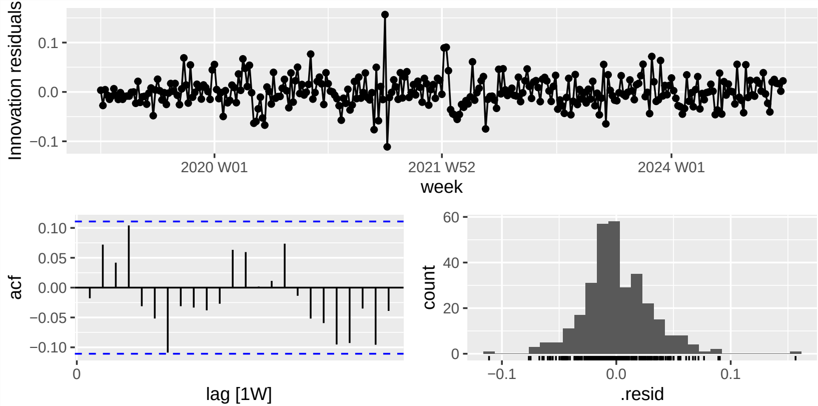
Example: Mortality rate in Canada

```r
fit <- mortality_ts_canada %>% model(ETS(Rate))
report(fit)
```

```
## Series: Rate
## Model: ETS(M,N,N)
##   Smoothing parameters:
##     alpha = 0.908
##
##   Initial states:
##  l[0]
##  14.4
##
##   sigma^2:  9e-04
##
##  AIC AICc  BIC
## 1229 1230 1241
```

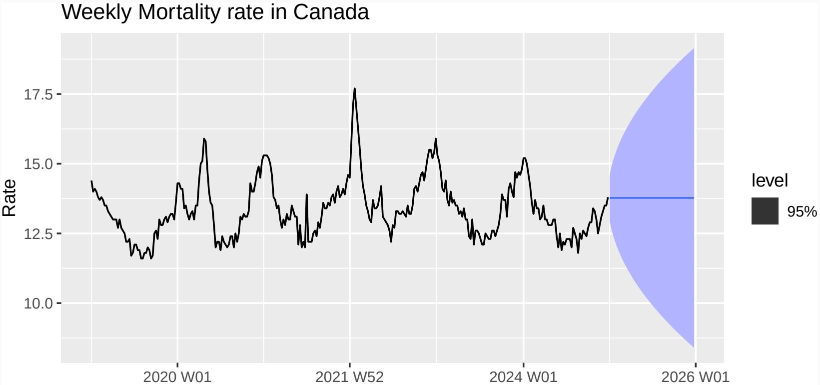
Mortality rate in Canada

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



Forecasting mortality rate in Canada

```r
fit |>
  forecast(h = 52) |>
  autoplot(mortality_ts_canada, level=95)+
  labs(title="Weekly Mortality rate in Canada",
       y="Rate")
```



Mortality rate in Canada (Training and test sets)

Training data: 2019 to 2023 and Test data: 2024

```
train <- mortality_ts_canada %>% filter(year(week) <= 2023)
fit <- train %>% model(ETS(Rate))

fit %>% forecast(h = "1 years") %>%
  fabletools::accuracy(mortality_ts_canada) %>%
  arrange(.model) %>%
  select(.model, .type, RMSE, MAE, MAPE, MASE)
```

```
## # A tibble: 1 x 6
##   .model     .type  RMSE   MAE  MAPE  MASE
##   <chr>      <chr> <dbl> <dbl> <dbl> <dbl>
## 1 ETS(Rate) Test   1.95  1.83  14.4  2.36
```

Final thoughts!

This lecture provides the benchmark and traditional methods to produce forecast, procedures for checking whether a forecasting method has adequately utilised the available information, and methods for evaluating forecast accuracy.

More details can be found in Chapters 5, 8 and 9 of Forecasting: Principles and Practice (3rd ed, `https://otexts.com/fpp3/`), as well as in many other time series forecasting resources.