Time Series Analysis in Health Research

Forecasting of Weekly Flu Positive Cases in Canada

Erfan Hoque
Dept. of Community Health and Epidemiology
University of Saskatchewan.

SEA 24th Annual Symposium Workshop, September 26, 2025

We will work in *fpp3* package. Load the library *fpp3*. If *fpp3* package is not installed, then we need to install it first and then load the library.

```
library(fpp3)
library(dplyr)
library(fable.prophet) # needed for Prophet model
```

This document is intended to serve as a rough template for what we covered in the last lectures on time series analysis in R.

Google Flu Data

Data Discription:

This is Google Flu trend data for Canada (2003-2015). These data are provided by the Statistical Society of Canada (SSC) in 2016 as part of the SSC case study competition. link: https://ssc.ca/en/case-study/can-google-flu-trends-predict-frequency-and-results-tests-influenza-and-other.

Google Flu Data Information:

In 2009-2010, a novel H1N1 influenza virus, often called *swine flu*, caused a global pandemic, which the WHO declared in June 2009 and ended in August 2010.

Google Flu Trends (GFT) was a service that used aggregated Google search query data to estimate the prevalence of influenza-like illness (ILI) in populations, aiming to provide real-time, early warnings for flu outbreaks. It provided estimates of influenza activity for more than 25 countries. The idea behind Google Flu Trends was that, by monitoring millions of users' health tracking behaviors online, the large number of Google search queries gathered can be analyzed to reveal if there is the presence of flu-like illness in a population.

While it showed initial promise, GFT was discontinued in 2015 after several years of inaccurate estimates, largely due to problems with its original algorithm, changes to Google's search algorithm, and biases in the data.

Source: https://en.wikipedia.org/wiki/Google_Flu_Trends

Frequency: Weekly

Geography: Canada and Some Provinces

Set Directory

To check your current working directory in R, use: `getwd()`

This will return the path of the directory where R is currently reading and writing files like this

```
getwd()
```

Read Data

To read a different data you need to change the name of the data based on the data file name.

```
flu <- read.csv("flu_model.csv")
```

tisbble objects

We need to create the data in *tisbble* object to use the functions from *fpp3* package.

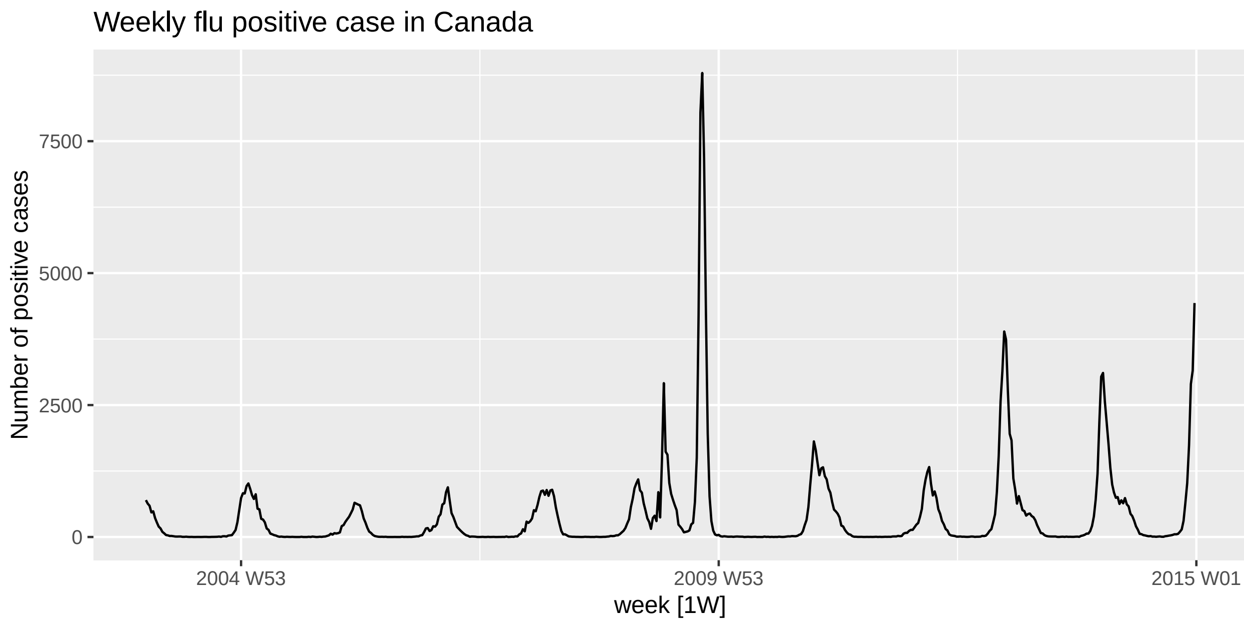
```
flu_ts <- flu |>
  mutate(week=yearweek(week),
         Date=as.Date(Date)) |>
  as_tsibble(index = week)
flu_ts |>
  head(10) # check first few rows of the data
```

```
## # A tsibble: 10 x 12 [1W]
##    Date           week FluApos FluBpos FluPos FluTest RSVtest RSVpos tem_CA
##    <date>       <week>   <int>   <int>  <int>   <int>   <int>  <int>  <dbl>
##  1 2004-01-04 2004 W01     693       6    699    3691    3158    129  -21.7
##  2 2004-01-11 2004 W02     632       0    632    3335    2754    152  -24.3
##  3 2004-01-18 2004 W03     591       1    592    3160    2404    180  -23.8
##  4 2004-01-25 2004 W04     465       0    465    3182    2483    262  -24.9
##  5 2004-02-01 2004 W05     483       4    487    3284    2456    363  -25.5
##  6 2004-02-08 2004 W06     356       5    361    2977    2437    449  -18.3
##  7 2004-02-15 2004 W07     273       1    274    2699    2138    387  -20.3
##  8 2004-02-22 2004 W08     190       7    197    2812    2485    544  -19.6
##  9 2004-02-29 2004 W09     145      17    162    2511    2299    489  -15.9
## 10 2004-03-07 2004 W10      89       8     97    2459    2479    486  -19.0
## # i 3 more variables: pre_CA <dbl>, hits_CA <int>, GFT.Canada <int>
```

Time series plot, patterns and decomposition

Time plot

```
flu_ts %>% autoplot(FluPos) +
  labs(title = "Weekly flu positive case in Canada",
         y="Number of positive cases")
```

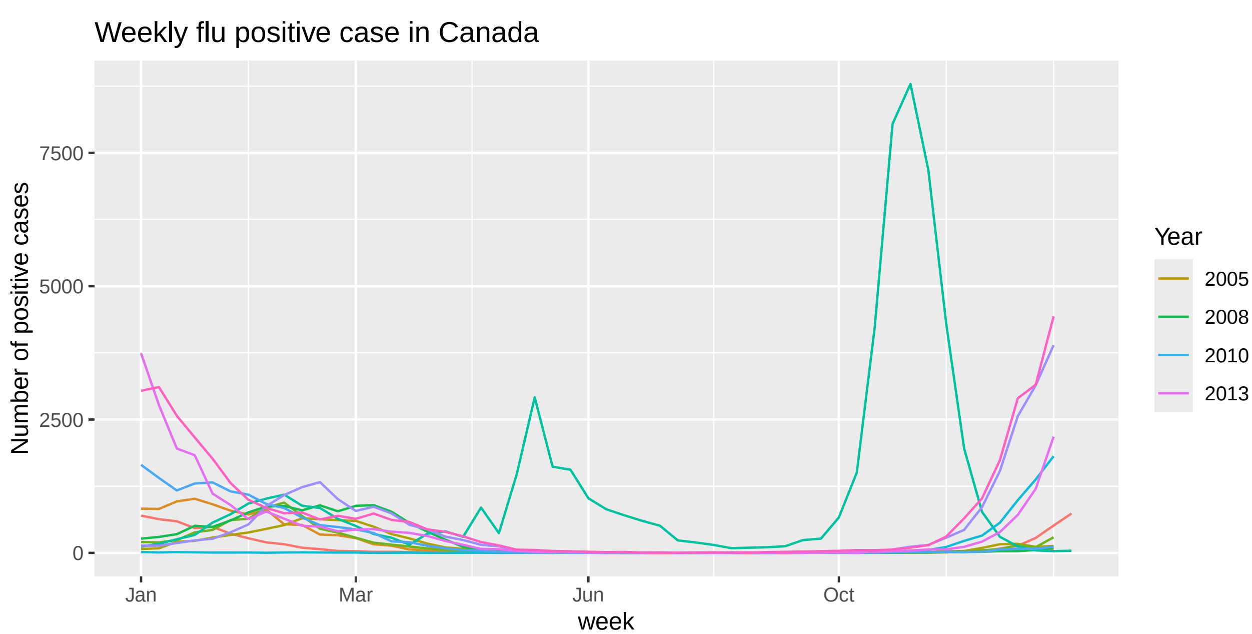


We can see some short trend and seasonality in the mortality rate.

Seasonal plots

A seasonal plot is similar to a time plot except that the data are plotted against the individual "seasons" in which the data were observed. This enables the underlying seasonal pattern to be seen more clearly, and also allows any substantial departures from the seasonal pattern to be easily identified.

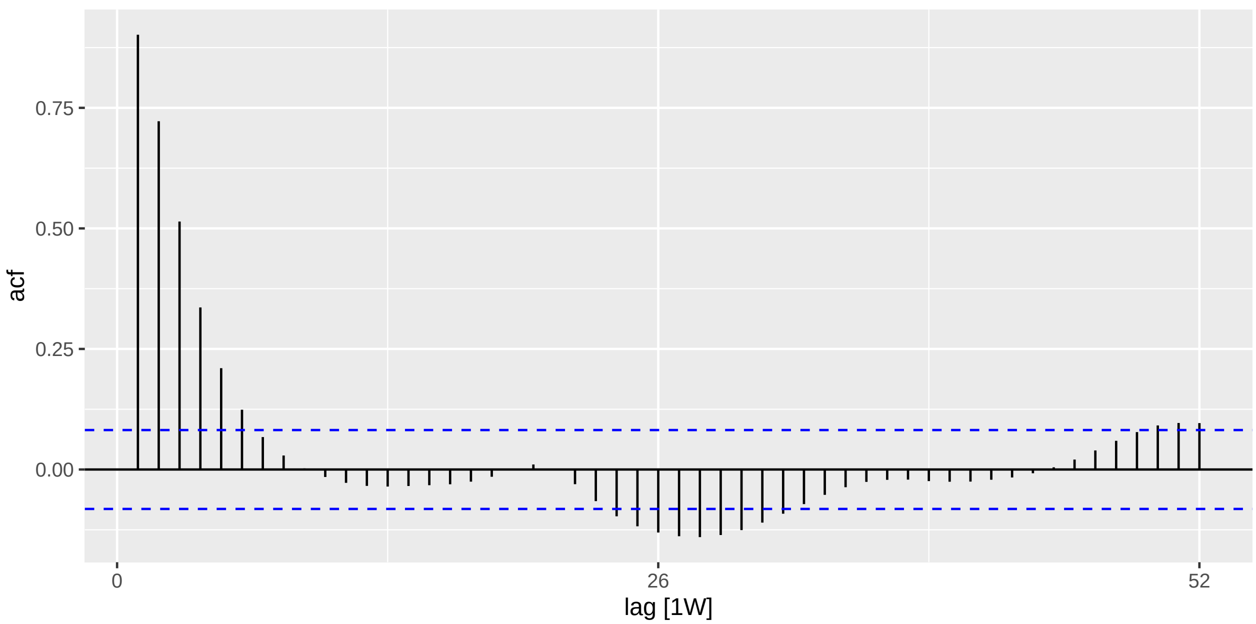
```
flu_ts |>
  gg_season(FluPos)+
      labs(title = "Weekly flu positive case in Canada",
         y="Number of positive cases")+
  guides(color = guide_legend(title = "Year"))
```



Autocorrelation

When data have a trend, the autocorrelations for small lags tend to be large and positive. When data are seasonal, the autocorrelations will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency). When data are trended and seasonal, we see a combination of these effects.

```
flu_ts %>%
  ACF(FluPos, lag_max = 52) %>% autoplot()
```



Time series decomposition

STL (Seasonal and Trend decomposition using Loess) is a versatile and robust method for decomposing time series.

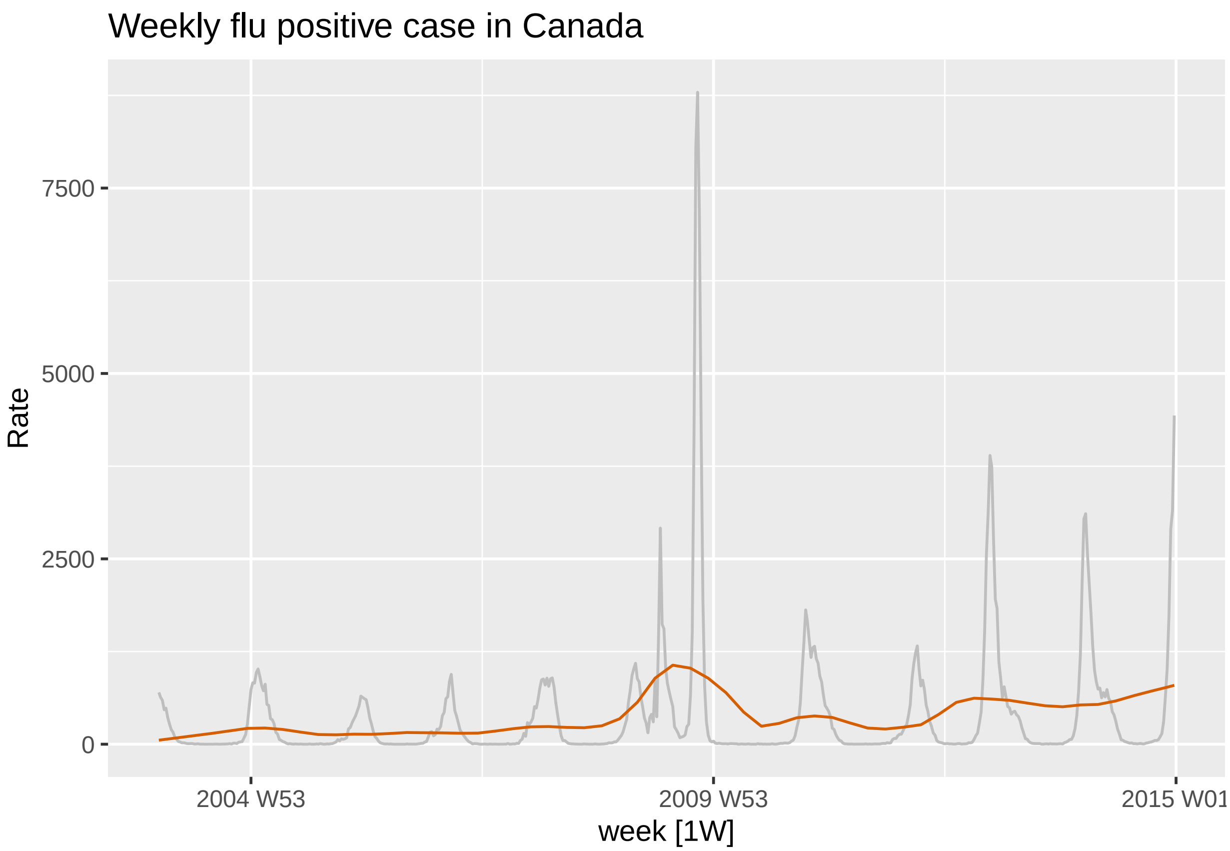
```
dcmp <- flu_ts %>%
  model(stl = STL(FluPos))
components(dcmp)
```

```
## # A dable: 574 x 7 [1W]
## # Key:      .model [1]
## # :         FluPos = trend + season_year + remainder
##    .model     week FluPos trend season_year remainder season_adjust
##    <chr>    <week>  <int> <dbl>       <dbl>     <dbl>         <dbl>
##  1 stl    2004 W01    699  54.7        138.     506.          561.
##  2 stl    2004 W02    632  57.7        175.     399.          457.
##  3 stl    2004 W03    592  60.8        168.     364.          424.
##  4 stl    2004 W04    465  63.8        194.     207.          271.
##  5 stl    2004 W05    487  66.9        245.     175.          242.
##  6 stl    2004 W06    361  70.0        252.      39.5         109.
##  7 stl    2004 W07    274  73.0        283.     -82.1          -9.08
##  8 stl    2004 W08    197  76.1        304.    -183.         -107.
##  9 stl    2004 W09    162  79.1        376.    -293.         -214.
```

```
## 10 stl     2004 W10      97   82.2         355.     -340.        -258.
## # i 564 more rows
```

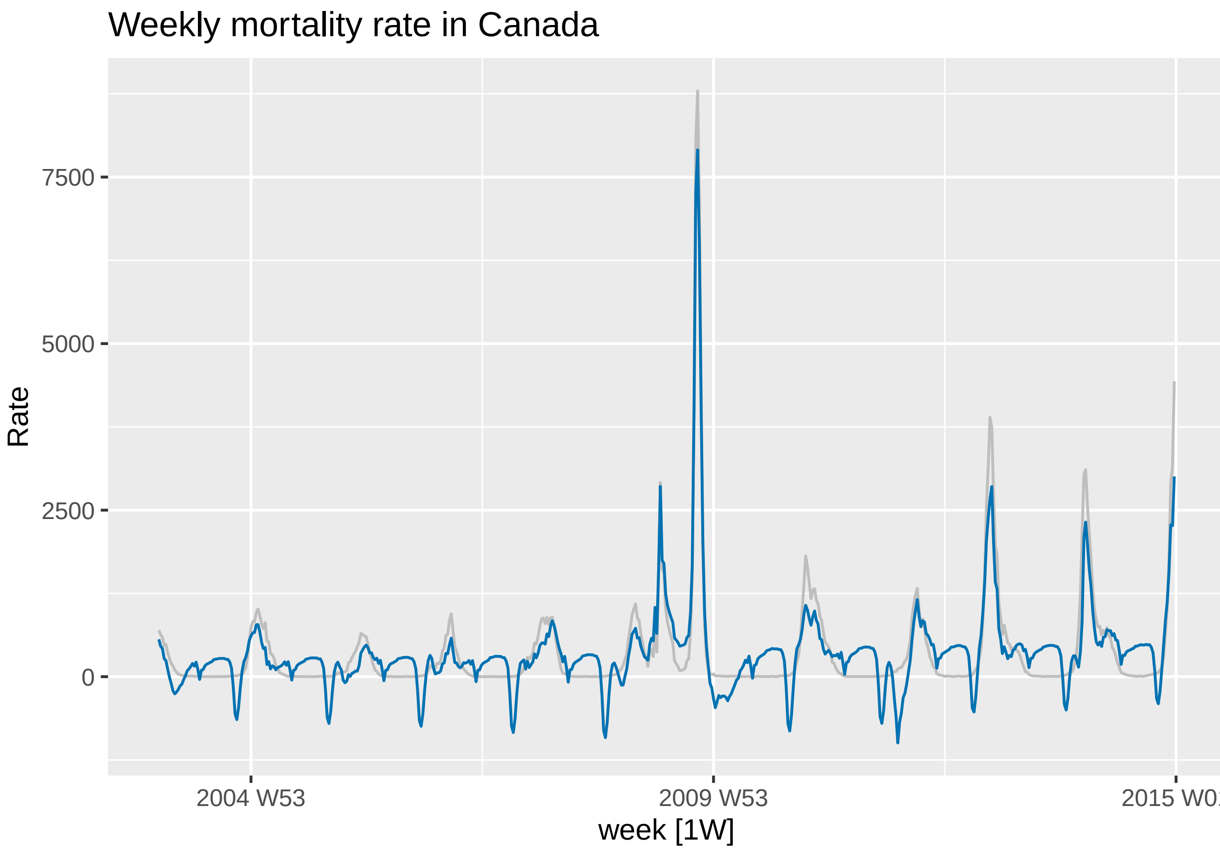
Trend-adjustment

```
flu_ts %>%
  autoplot(FluPos, color='gray') +
  autolayer(components(dcmp), trend, color='#D55E00') +
  labs(y = "Rate", title = "Weekly flu positive case in Canada")
```



Seasonal-adjustment

```
flu_ts %>%
  autoplot(FluPos, color='gray') +
  autolayer(components(dcmp), season_adjust, color='#0072B2') +
  labs(y = "Rate", title = "Weekly mortality rate in Canada")
```

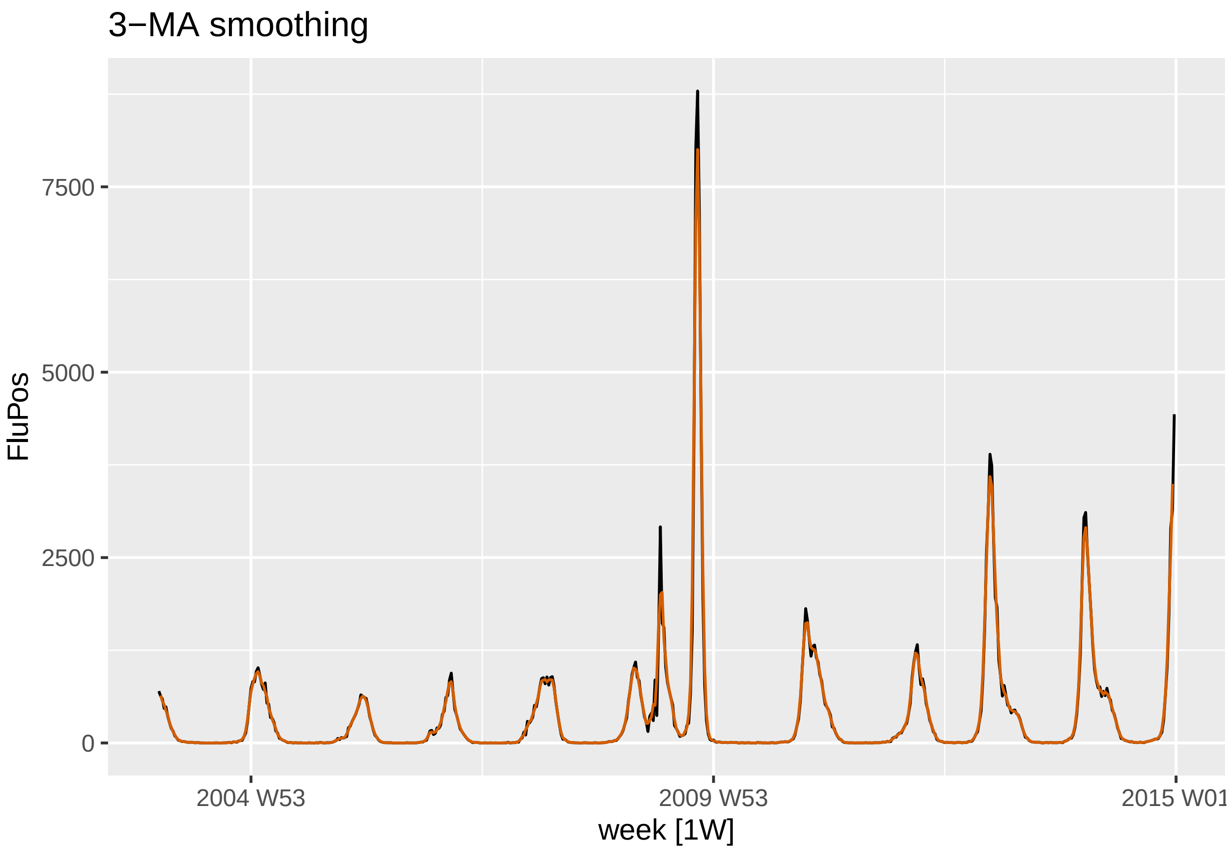


Classical Decomposition

The traditional way to do time series decomposition is called `Classical decomposition`. The simplest estimate of the trend-cycle uses *moving averages* which is an average of nearby points.

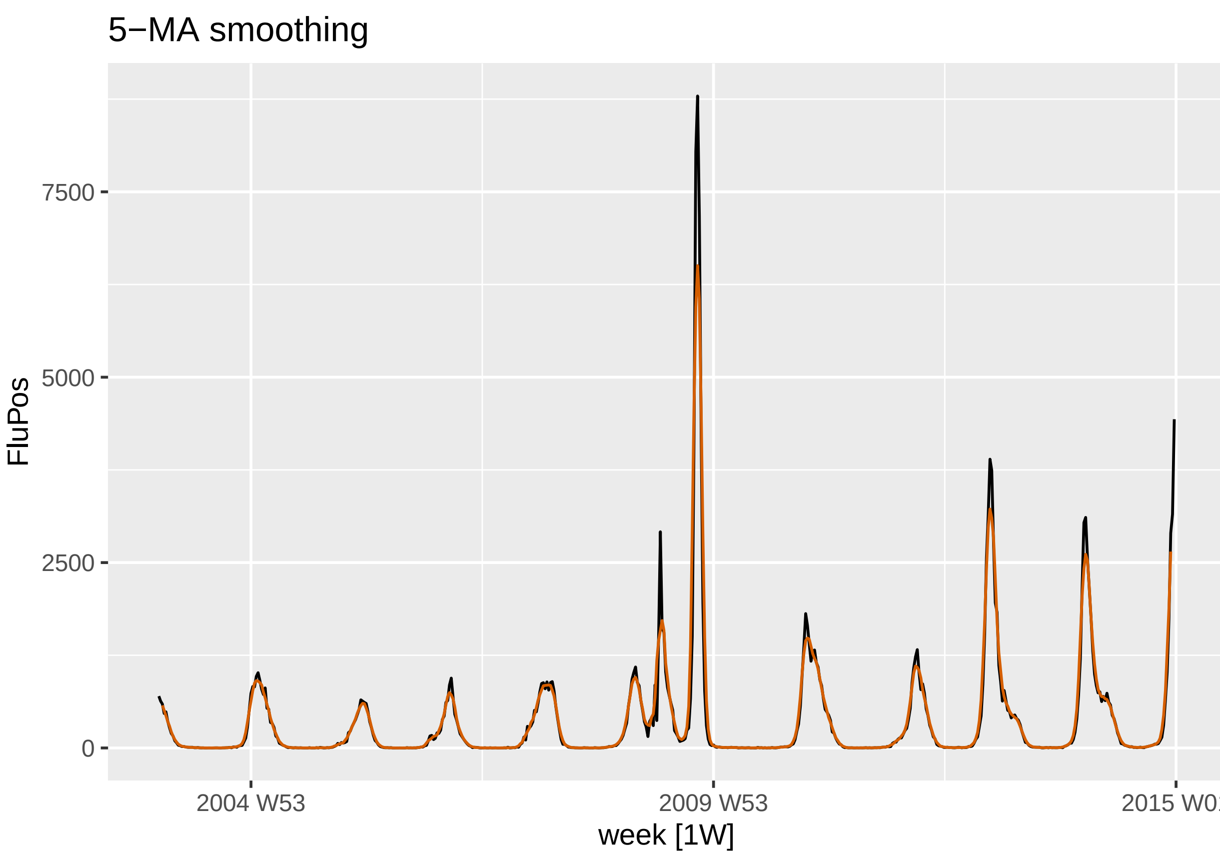
```
flu_ts_decom <- flu_ts |>
  mutate(
    `3-MA` = slider::slide_dbl(FluPos, mean,
                .before = 1, .after = 1, .complete = TRUE),
    `5-MA` = slider::slide_dbl(FluPos, mean,
                .before = 2, .after = 2, .complete = TRUE)
  )
# plot of 3-MA
flu_ts_decom |>
  autoplot(FluPos) +
  geom_line(aes(y = `3-MA`), colour = "#D55E00")+
  labs(title = "3-MA smoothing")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```r
# plot of 5-MA
flu_ts_decom |>
  autoplot(FluPos) +
  geom_line(aes(y = `5-MA`), colour = "#D55E00")+
  labs(title = "5-MA smoothing")
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```



Forecasting Models

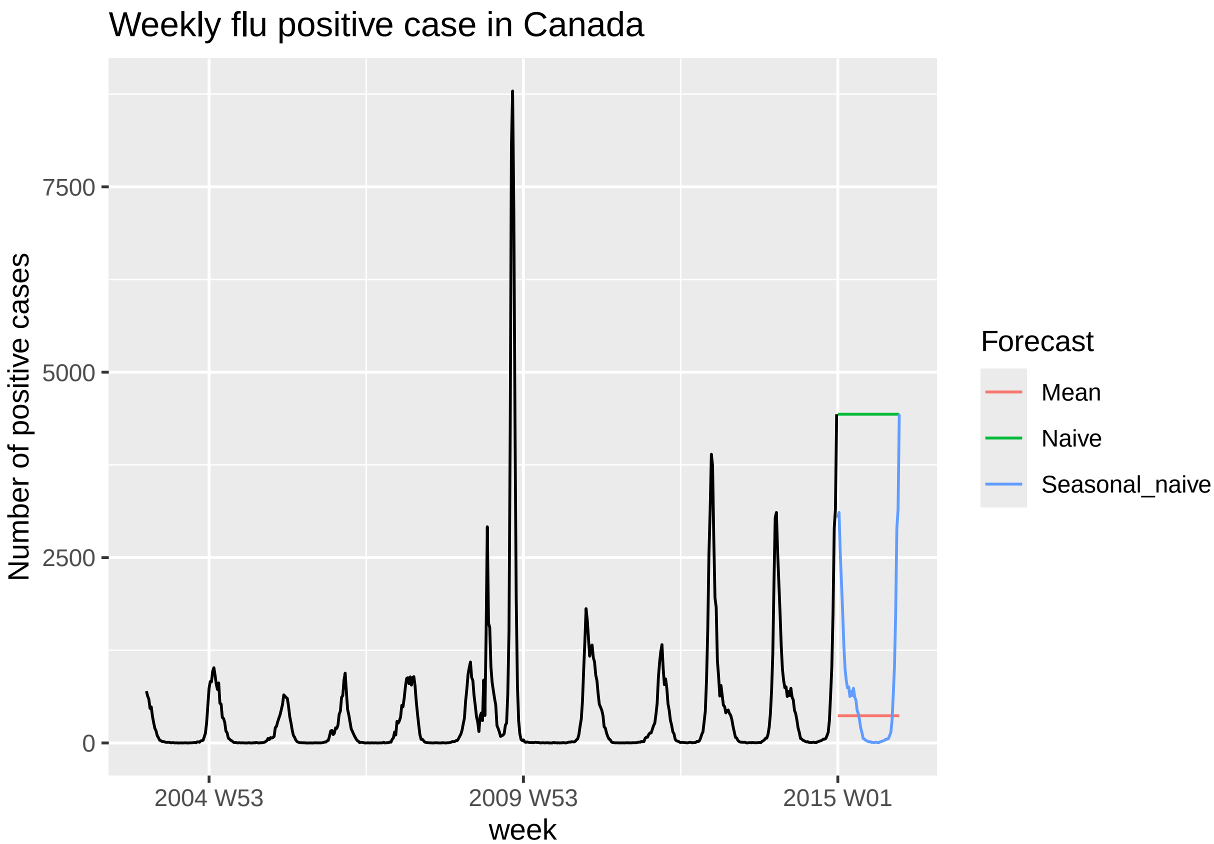
Simple forecasting methods

The `model()` function trains models to data. We are going to forecast using Mean, Naive and Seasonal Naive method.

```
flu_fit <-  flu_ts %>%
  model(
    Seasonal_naive = SNAIVE(FluPos),
    Naive = NAIVE(FluPos),
    Mean = MEAN(FluPos)
  )
```

We can now produce forecasts using the fitted models.

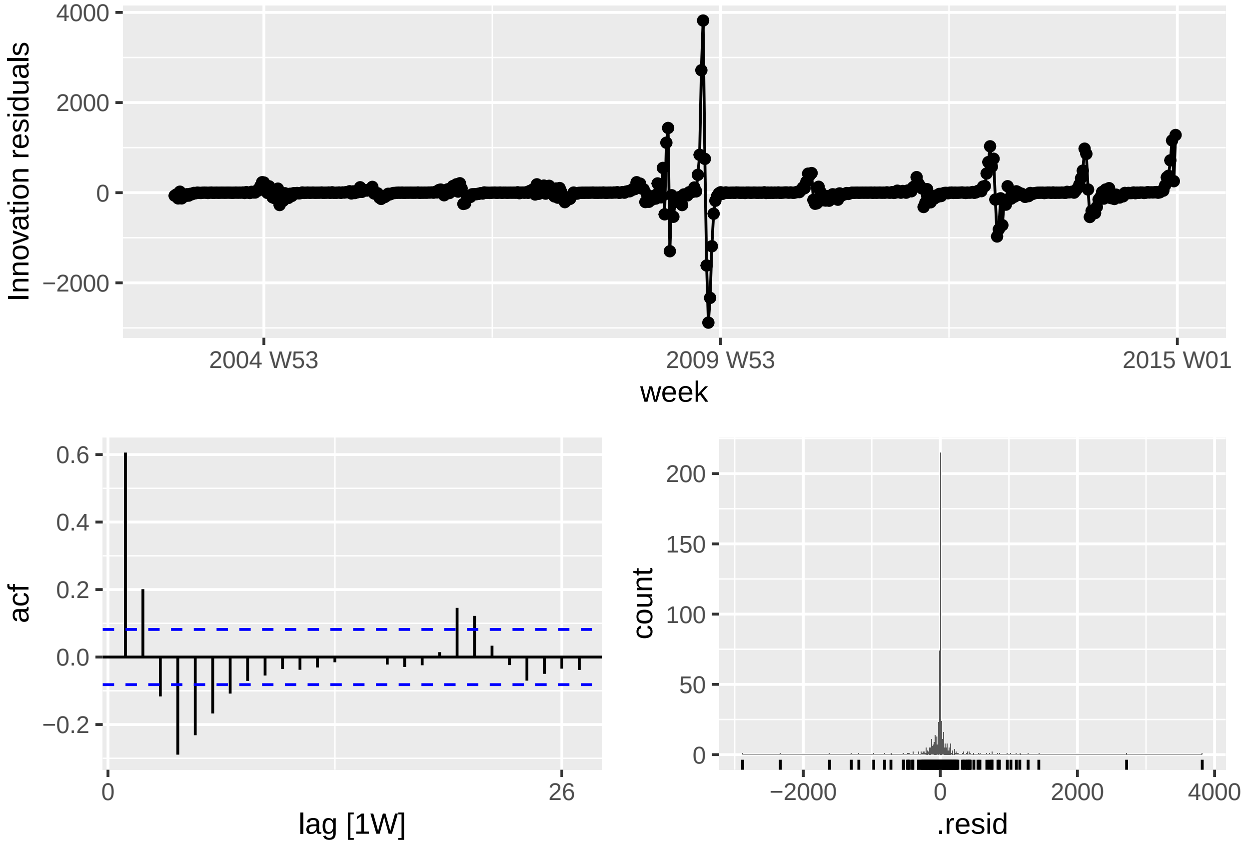
```
flu_fc <- flu_fit %>%
  forecast(h = "1 years")

flu_fc %>%
  autoplot(flu_ts, level = NULL) +
  labs(title = "Weekly flu positive case in Canada",
       y = "Number of positive cases") +
  guides(colour = guide_legend(title = "Forecast"))
```



Residual diagnostics

It is very important to do the residual diagnostic after fitting any model to check whether the residual assumptions have been satisfied or not.

```
fit.naive <- flu_ts %>% model(NAIVE(FluPos))
gg_tsresiduals(fit.naive)
```



Here, based on the plots, we can say the residual assumptions (uncorrelated, mean zero, constant variance) have not been satisfied for Naive model.

Tests for autocorrelation

Moreover, we can do Box-Pierce or Ljung-Box test to see whether the residuals are significantly different from a zero set.

H_0: the series is white noise vs H_1: the series is not white noise.

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

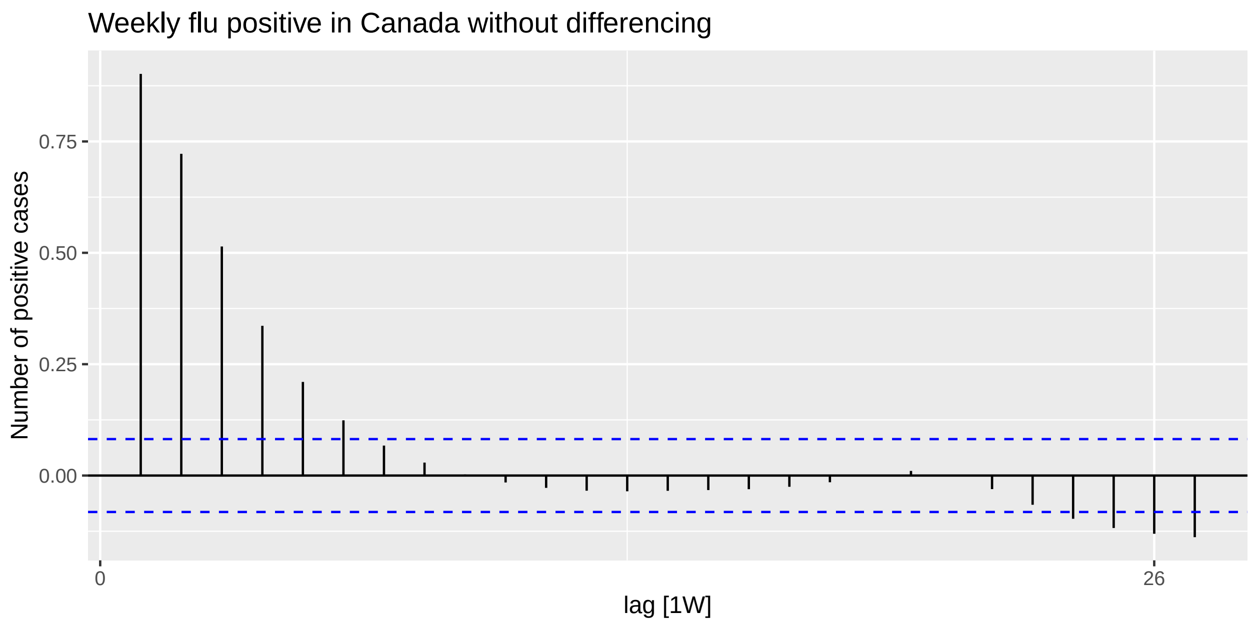
```
## # A tibble: 1 x 3
##   .model        lb_stat lb_pvalue
##   <chr>           <dbl>     <dbl>
## 1 NAIVE(FluPos)    351.         0
```

The results are significant (i.e., the p-values are relatively small < 0.05). Thus, we can conclude that the residuals are distinguishable from a white noise series.

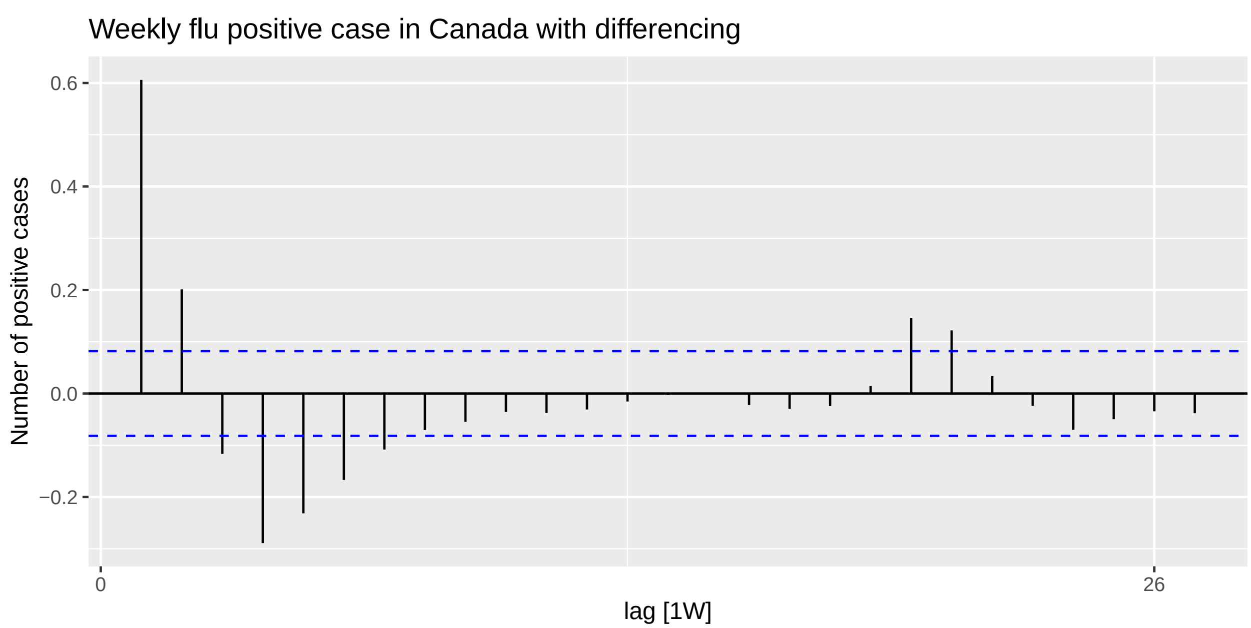
ARIMA / SARIMA models

Before fitting the ARIMA model, it is important to check the stationary of the data. If the data is non-stationary, then we need to make it stationary using the idea of differencing. Differencing helps to stabilize the mean (one way to make a non-stationary time series stationary).

```r
flu_ts %>% ACF(FluPos) %>% autoplot() +
    labs(title = "Weekly flu positive in Canada without differencing",
       y = "Number of positive cases")
```




```r
flu_ts %>% ACF(difference(FluPos)) %>% autoplot() +
    labs(title = "Weekly flu positive case in Canada with differencing",
       y = "Number of positive cases")
```



First order differencing seems make the mortality data stationary. But still there are some pattern left over.

Now let's fit the ARIMA/SARIMA model. The *ARIMA* function form *fpp3* package does everything together (checking for stationarity of the data, doing any differencing if needed and then fitting the model using ARIMA or SARIMA based on data)

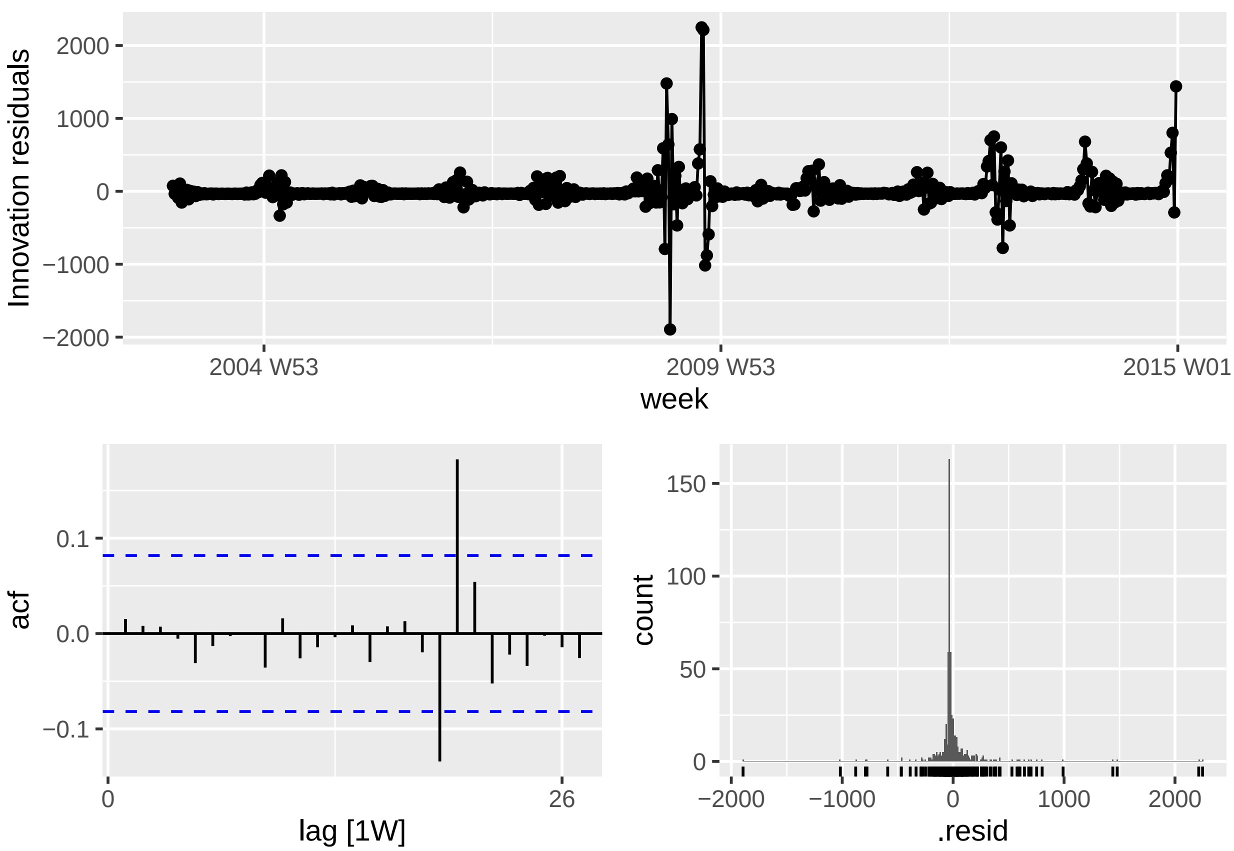
```r
fit.arima <- flu_ts |>
  model(ARIMA(FluPos))
report(fit.arima)
```

```
## Series: FluPos
## Model: ARIMA(1,0,3)(0,0,1)[52] w/ mean
##
## Coefficients:
##          ar1     ma1     ma2     ma3    sma1  constant
##       0.7499  0.9108  0.6418  0.3873  0.0657  102.0171
## s.e.  0.0357  0.0444  0.0549  0.0436  0.0391   30.2154
##
## sigma^2 estimated as 55249:  log likelihood=-3947.27
## AIC=7908.54   AICc=7908.74   BIC=7939.01
```

We have SARIMA(1,0,3)(0,0,1)[52] model for this data.

Let's check the residual.

```r
gg_tsresiduals(fit.arima)
```



Here, based on the plots, we can say the residual assumptions (uncorrelated, mean zero, constant variance) have been satisfied for ARIMA model (still some pick is there!).

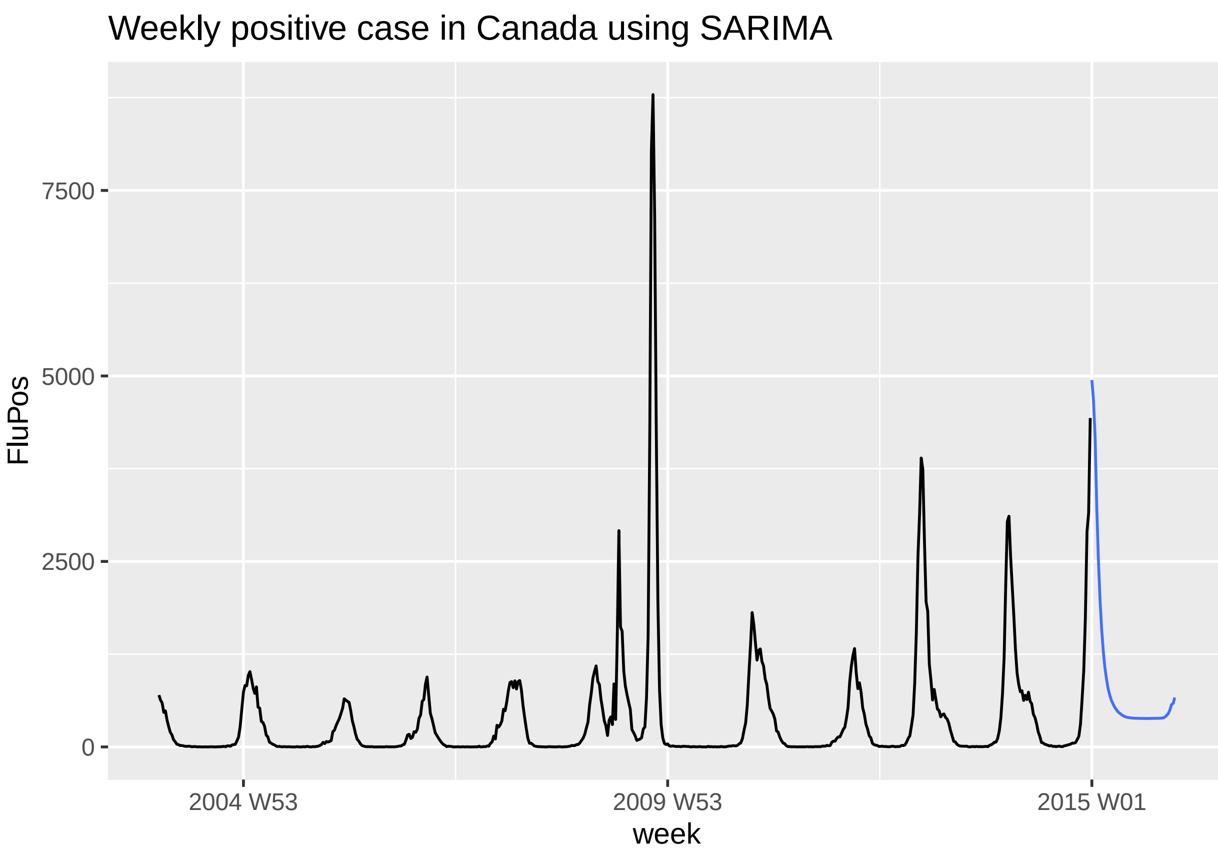
```r
augment(fit.arima) %>%
  features(.innov, ljung_box, lag = 36, dof=4)
```

```
## # A tibble: 1 x 3
##   .model        lb_stat lb_pvalue
##   <chr>           <dbl>     <dbl>
## 1 ARIMA(FluPos)    40.4     0.147
```

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.

Let's forecast mortality rate for next one year (52 weeks).

```r
fit.arima %>% forecast(h = 52) %>%  # h = "1 years"
  autoplot(flu_ts, level=NULL) +
  labs(y = "FluPos", title = "Weekly positive case in Canada using SARIMA")
```



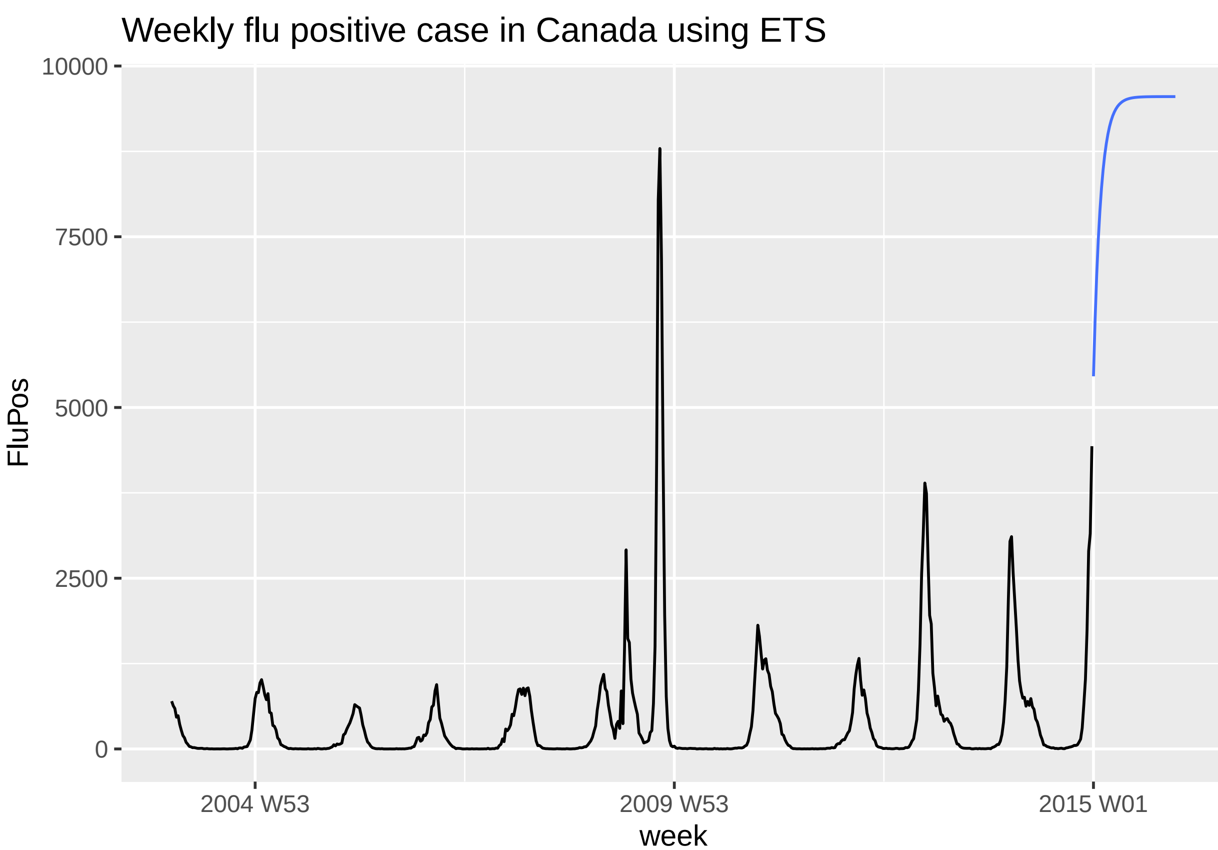
Simple Exponential Smoothing

```r
fit.ets <- flu_ts %>% model(ETS(FluPos))
report(fit.ets)
```

```
## Series: FluPos
```

```
## Model: ETS(A,Ad,N)
##   Smoothing parameters:
##     alpha = 0.9998995
##     beta  = 0.9998831
##     phi   = 0.8000001
##
##   Initial states:
##       l[0]      b[0]
##  775.7788 -57.87618
##
##   sigma^2:  71534.02
##
##      AIC     AICc      BIC
## 10069.52 10069.67 10095.63
```

```r
fit.ets %>%
  forecast(h = "1 years") %>%
  autoplot(flu_ts, level=NULL)+
  labs(title="Weekly flu positive case in Canada using ETS", y="FluPos")
```



Neural Netwrok Model

Now, we will use Neural network autoregression model to forecast mortality rate using `NNETAR()` function that fits an NNAR$(p, P, k)_m$ model. If p and P are not specified, they are automatically selected.

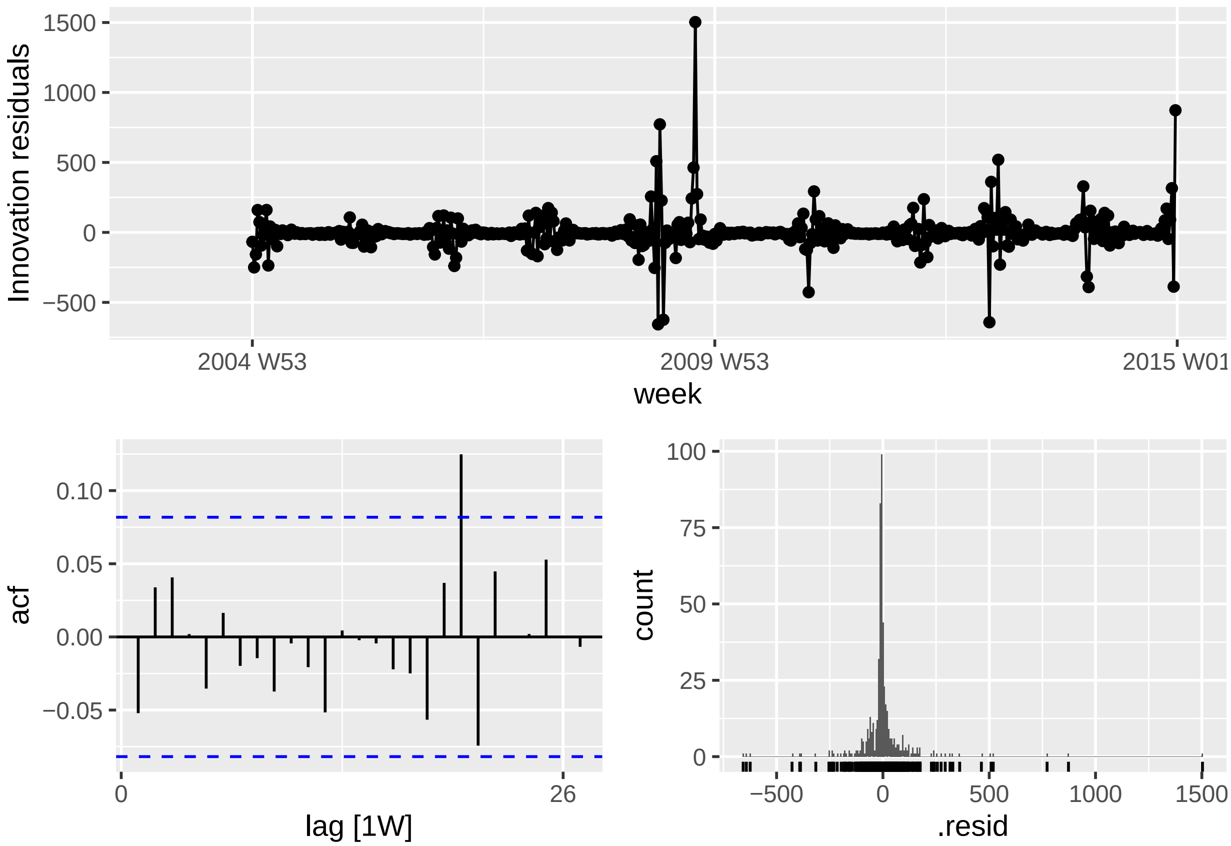
```r
fit.nn <- flu_ts %>% model(NNETAR(FluPos))
fit.nn
```

```
## # A mable: 1 x 1
##     `NNETAR(FluPos)`
##              <model>
## 1 <NNAR(6,1,4)[52]>
```

The result provides a NNAR(6,1,4)[52] model for mortality rate. Here, the last 6 observations are used as predictors, and there are 4 neurons in the hidden layer.

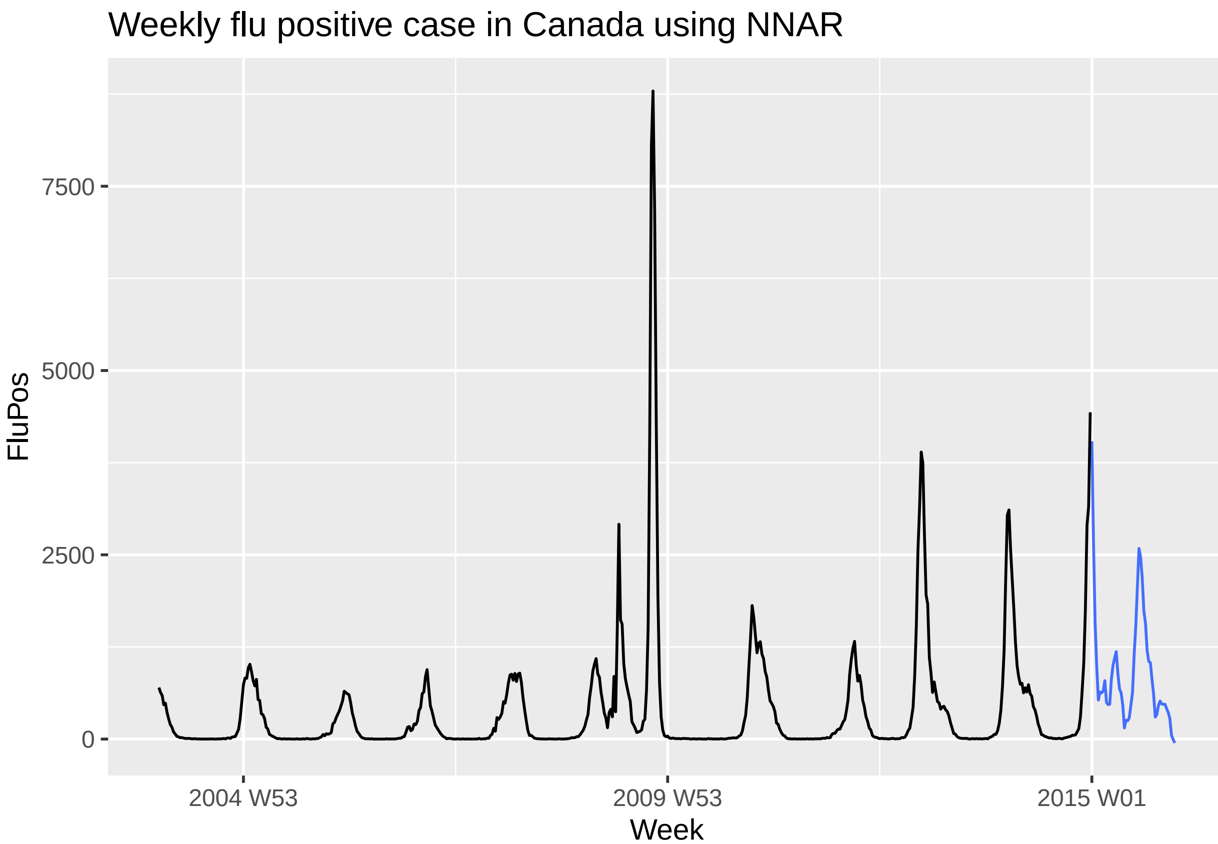
Let's check the residual.

```r
gg_tsresiduals(fit.nn)
```



Let's forecast mortality rate for next one year (52 weeks).

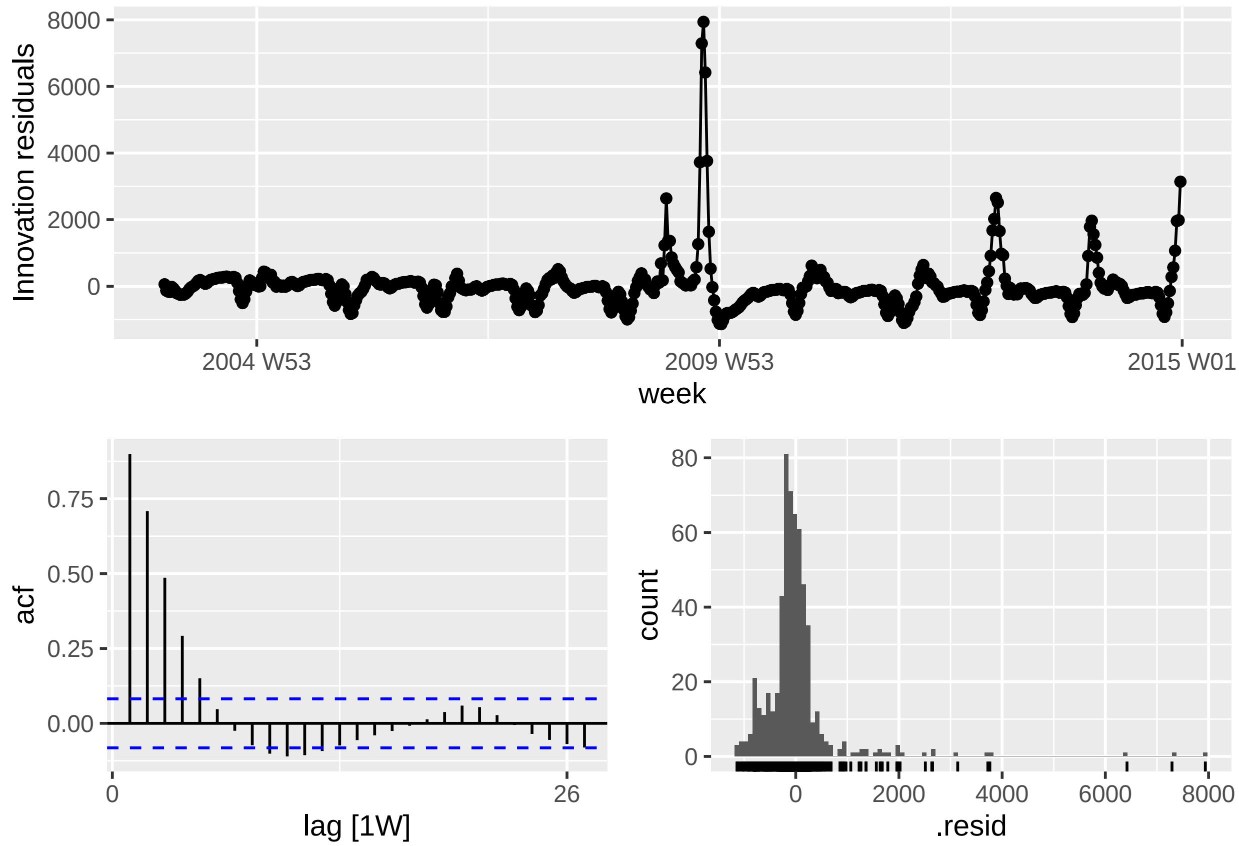
```r
fit.nn %>% forecast(h="1 years", times=1) %>%
  autoplot(flu_ts, level=NULL) +
  labs(x = "Week", y = "FluPos", title = "Weekly flu positive case in Canada using NNAR")
```



Prophet Model

Prophet is an open-source forecasting tool developed by Facebook for time series forecasting. Prophet model is available via the `fable.prophet` package.

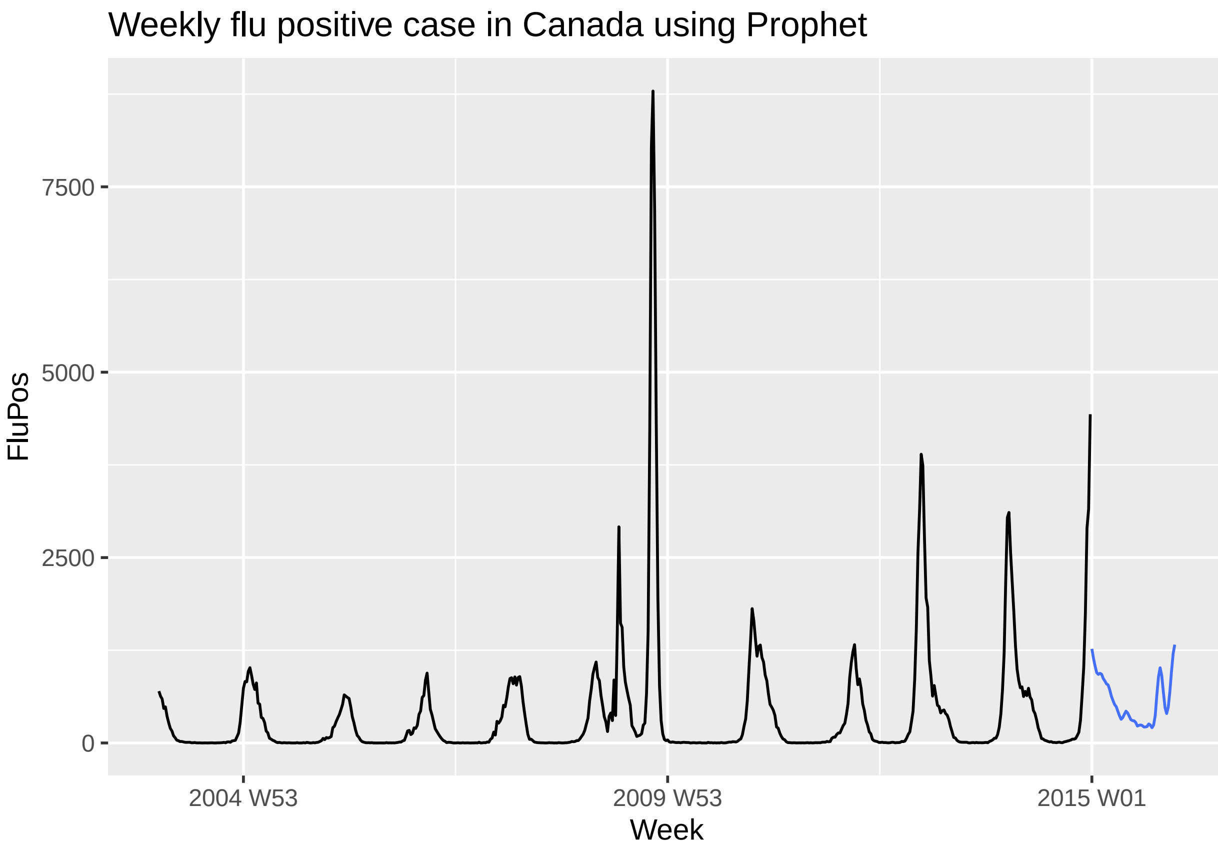
```
fit.ph <- flu_ts %>%
  model(prophet(FluPos ~ season(period = 52, order = 10)))
fit.ph |> gg_tsresiduals() # check residual
```




```
#components(fit.ph) %>% autoplot()
```

Let's forecast mortality rate for next one year (52 weeks).

```
fit.ph %>% forecast(h = "1 years") %>%
  autoplot(flu_ts, level=NULL) +
  labs(x = "Week", y = "FluPos", title = "Weekly flu positive case in Canada using Prophet")
```



Forecast accuracy: Comparing all models

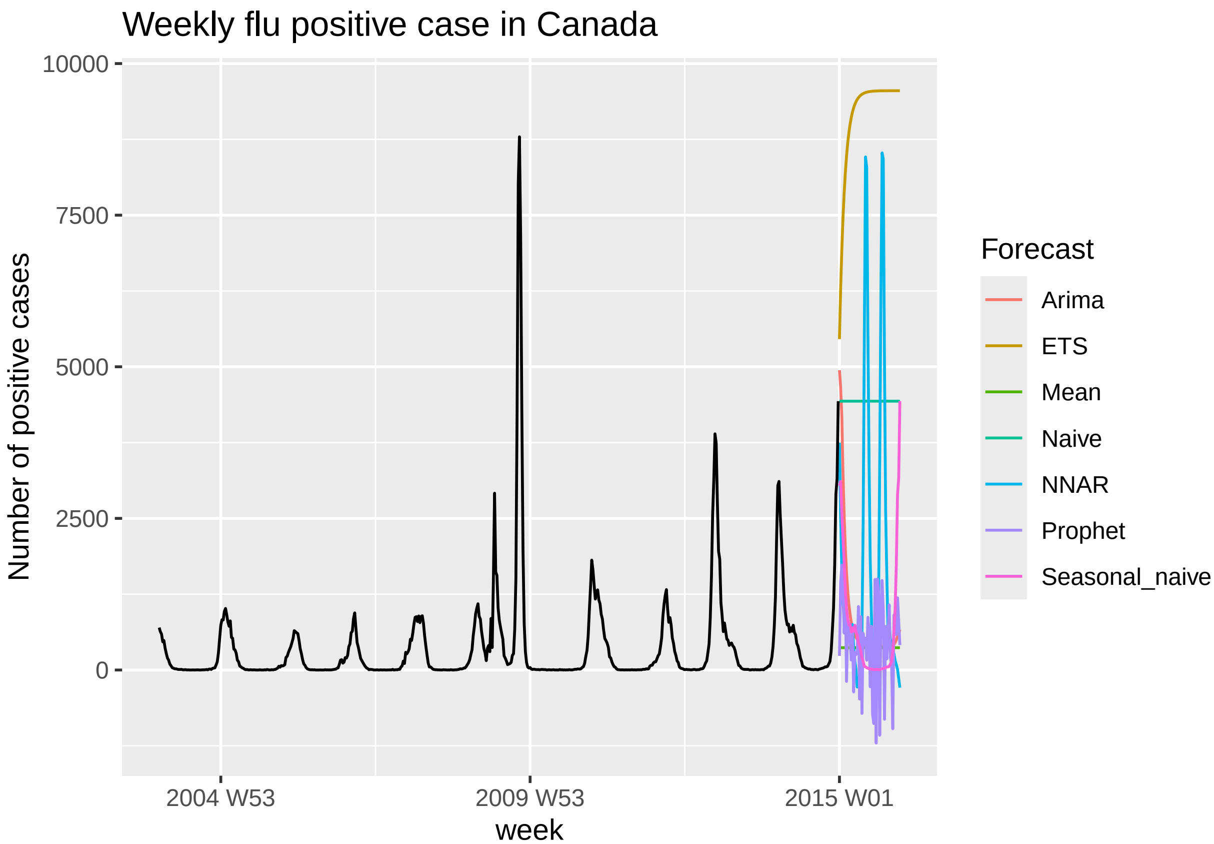
We can fit all model together in one function and compare them based on some accuracy measures. Then choose the best one and forecast based on the best model.

```
flu_fit <-  flu_ts %>%
  model(
    Seasonal_naive = SNAIVE(FluPos),
    Naive = NAIVE(FluPos),
    Mean = MEAN(FluPos),
    Arima = ARIMA(FluPos),
    ETS = ETS(FluPos),
    NNAR = NNETAR(FluPos),
    #Prophet = prophet(FluPos)
    Prophet = prophet(FluPos ~ season(period = 52, order = 10))
  )
```

We can now produce forecasts for next 1 years (52 weeks) using the fitted models.

```
flu_fc <- flu_fit %>%
  forecast(h = "1 years", times=1)

flu_fc %>%
  autoplot(flu_ts, level = NULL) +
  labs(title = "Weekly flu positive case in Canada",
```

```r
      y = "Number of positive cases") +
  guides(colour = guide_legend(title = "Forecast"))
```



Let's compare all models.

```r
accuracy(flu_fit)
```

```
## # A tibble: 7 x 10
##   .model         .type          ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
##   <chr>          <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 Seasonal_naive Training  6.19e+ 1 1122. 436.   -Inf   Inf 1     1       0.912
## 2 Naive          Training  6.52e+ 0  330. 107.   -Inf   Inf 0.245 0.294   0.606
## 3 Mean           Training -8.38e-13  836. 451.   -Inf   Inf 1.03  0.745   0.902
## 4 Arima          Training  2.38e- 1  234.  97.1   NaN   Inf 0.223 0.208   0.0153
## 5 ETS            Training  3.14e+ 0  266.  87.3   NaN   Inf 0.200 0.237   0.0776
## 6 NNAR           Training -9.18e- 1  126.  54.6  -Inf   Inf 0.125 0.112  -0.0455
## 7 Prophet        Training -9.12e- 1  755. 367.    NaN   Inf 0.840 0.673   0.899
```

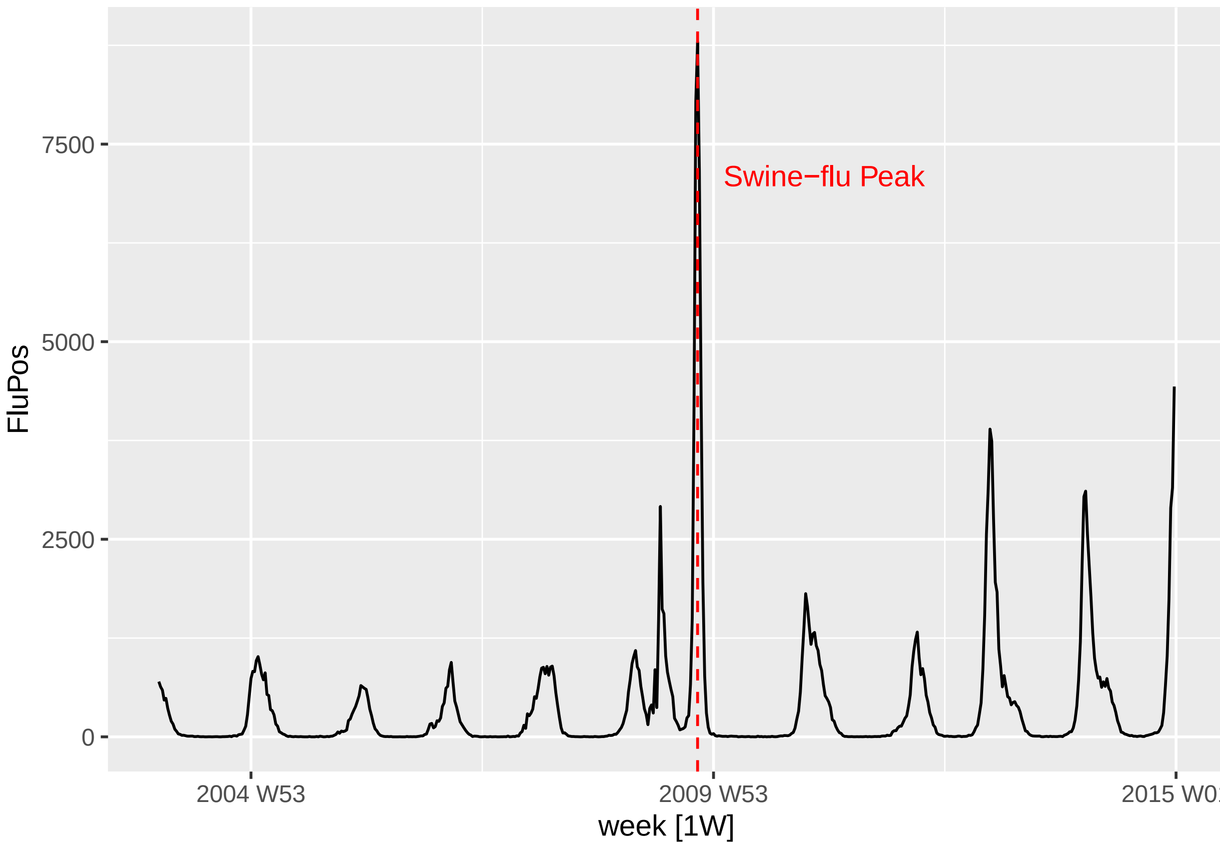
Based on the accuracy measures (say, RMSE, MAE, MAPE, MASE), it is found that Neural Net and then ARIMA models provide better forecasting performance compare to other models.

Time Series Regression

Now, we are going to do the time series regression models. We will forecast the weekly mortality rate based on some other predictor, say mean temperature of week, Google flu tend, trend of the data, any intervention effect (e.g. swine-flu effect) and so on. We need to assume that morality rate has a linear relationship with these predictors.

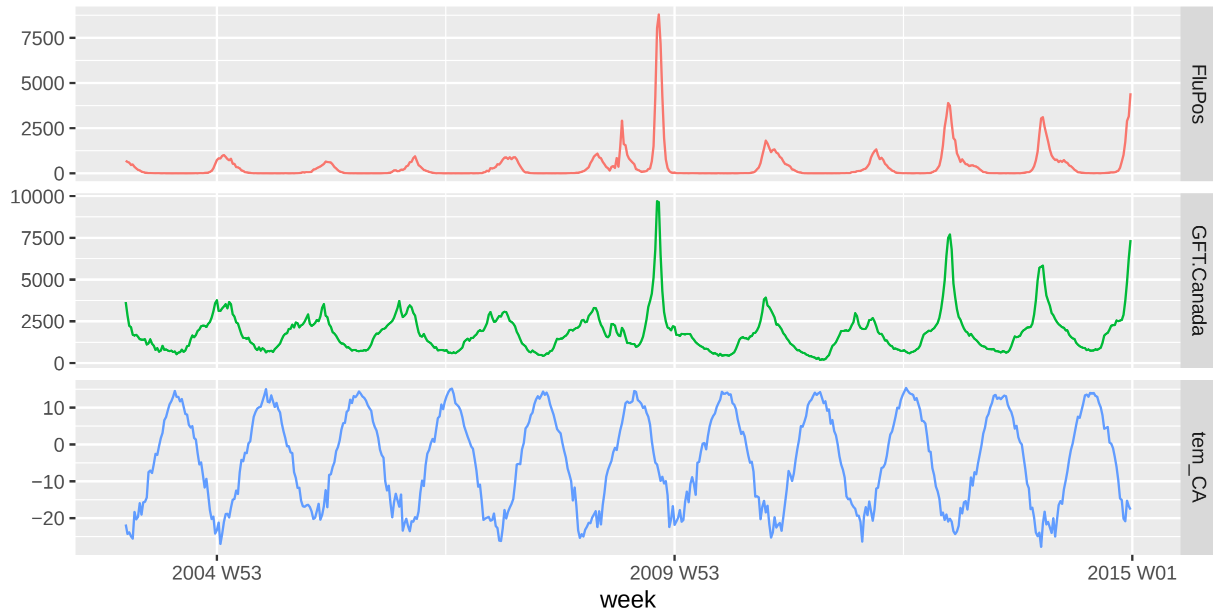
First let's look at the time plot again.

```
peak_time <- flu_ts |>
  filter(FluPos == max(FluPos, na.rm = TRUE)) |>
  pull(index(flu_ts)) |>
  as.Date()
flu_ts %>% autoplot(FluPos) +
  geom_vline(xintercept = peak_time, linetype = "dashed", color = "red")+
  annotate("text", x = peak_time + 500, y = 7500, label = "Swine-flu Peak",
           vjust = 2, color = "red", angle = 0)
```



Now, let's look at the relationship among variables.

```
flu_ts %>%
  select(FluPos, tem_CA, GFT.Canada, week) %>% pivot_longer(-week) |>
  ggplot(aes(week, value, colour = name)) +
  geom_line() + facet_grid(name ~ ., scales = "free_y") +
  guides(colour = "none") + labs(y=" ")
```



Let's see the relation between weekly mortality rate and weekly mean temperature.

```
flu_ts %>%
  ggplot(aes(y = FluPos, x = GFT.Canada)) +
  labs(x = "GFT Trend", y = "Flu positive case in Canada") +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

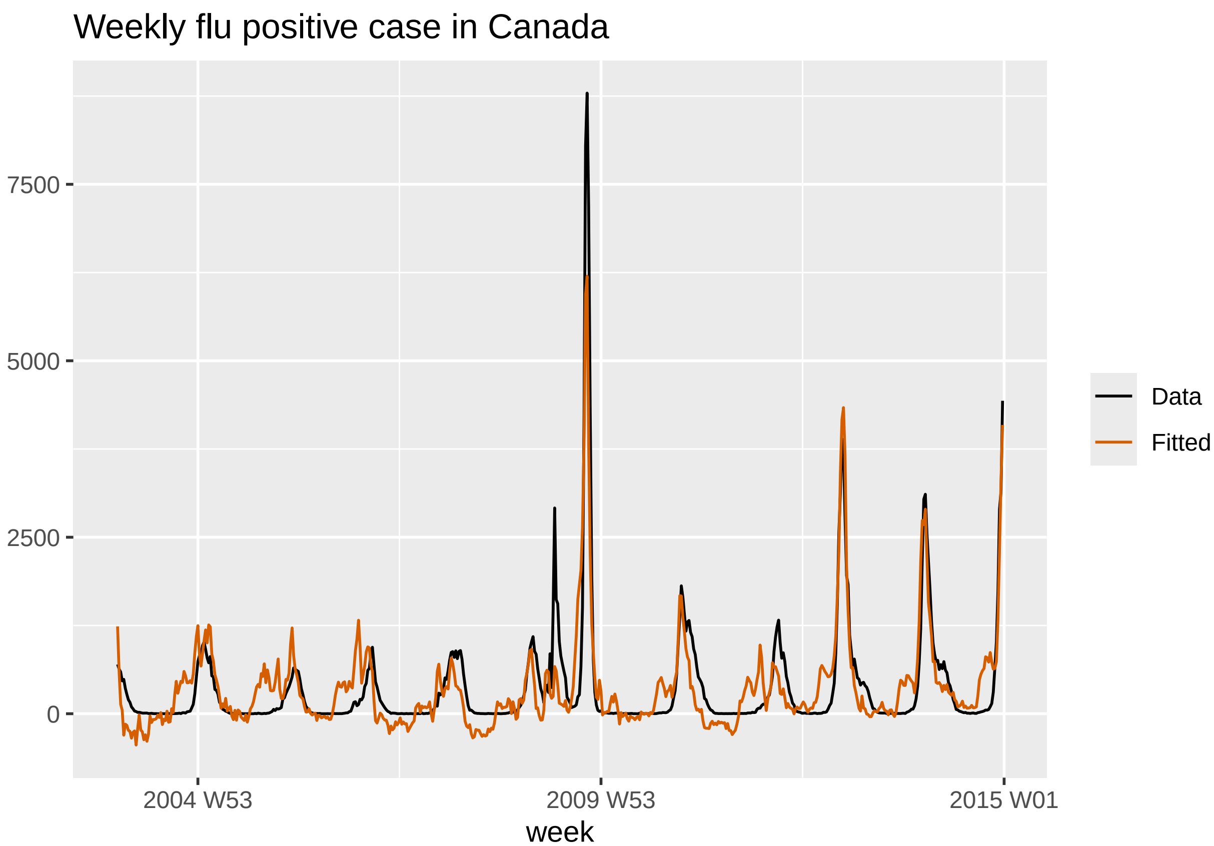
Let's fit the time series regression model using covariates/predictors, say swine-flu intervention effect, trend variable (already built in *fpp3* package as *trend()*)

```
# create dummy for swine-flu intervention effect
flu_ts <- flu_ts |>
  mutate(swineflu=ifelse(Date<as.Date("2009-10-26"),0,1))

fit_flu <- flu_ts |>
  model(TSLM(FluPos ~ GFT.Canada+ tem_CA+ swineflu +trend()))
report(fit_flu)
```

```
## Series: FluPos
## Model: TSLM
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1560.885  -227.953     0.134   201.351  3343.856
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -869.28451   47.35385 -18.357  < 2e-16 ***
## GFT.Canada     0.73645    0.02048  35.966  < 2e-16 ***
## tem_CA        26.74132    1.96668  13.597  < 2e-16 ***
## swineflu     363.51584   73.01782   4.978 8.51e-07 ***
## trend()       -0.61217    0.22224  -2.755  0.00606 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 427.1 on 569 degrees of freedom
## Multiple R-squared: 0.7414,  Adjusted R-squared: 0.7395
## F-statistic: 407.8 on 4 and 569 DF, p-value: < 2.22e-16
```

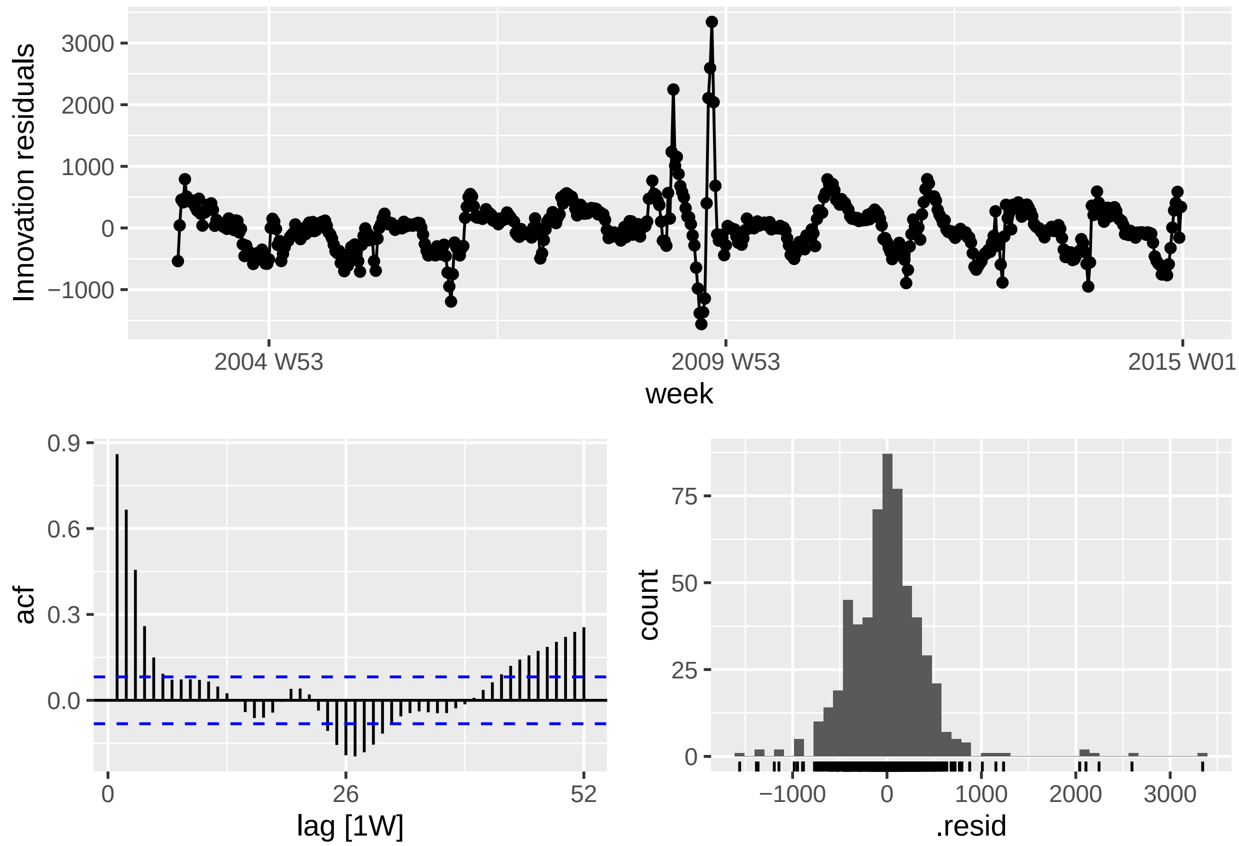
```
# Check the fitted values
augment(fit_flu) |>
  ggplot(aes(x = week)) +
  geom_line(aes(y = FluPos, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(y = NULL,
    title = "Weekly flu positive case in Canada"
  ) +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```



Residual diagnostics

It is very important to do residual diagnostic after fitting the regression model so that we can verify the assumptions (residual's are uncorrelated, zero mean, constant variance, uncorrelated with other predictor).

```
fit_flu |> gg_tsresiduals(lag=52)
```



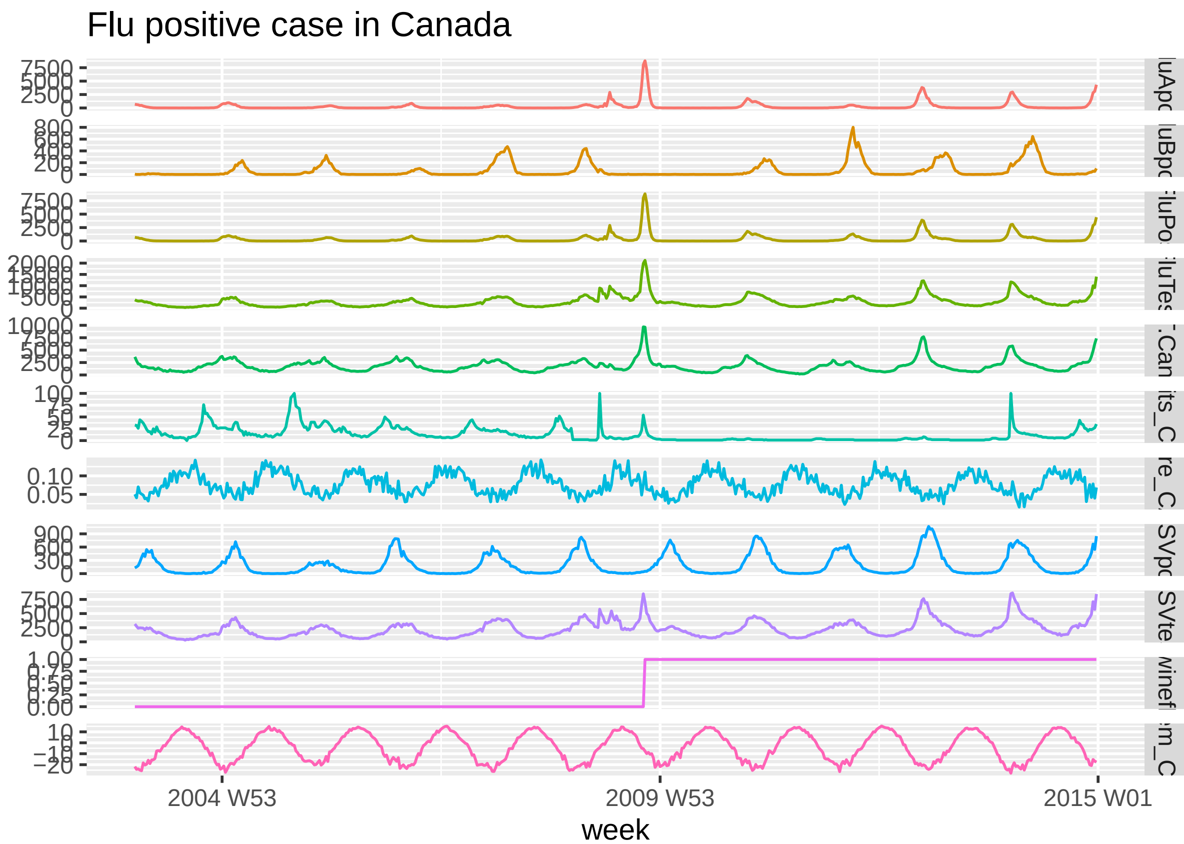
By looking at the auto correlation function of residuals, we can see still some correlation left in the residuals!

What to do then??

Solution: we can use Dynamic regression model.

Dynamic regression models

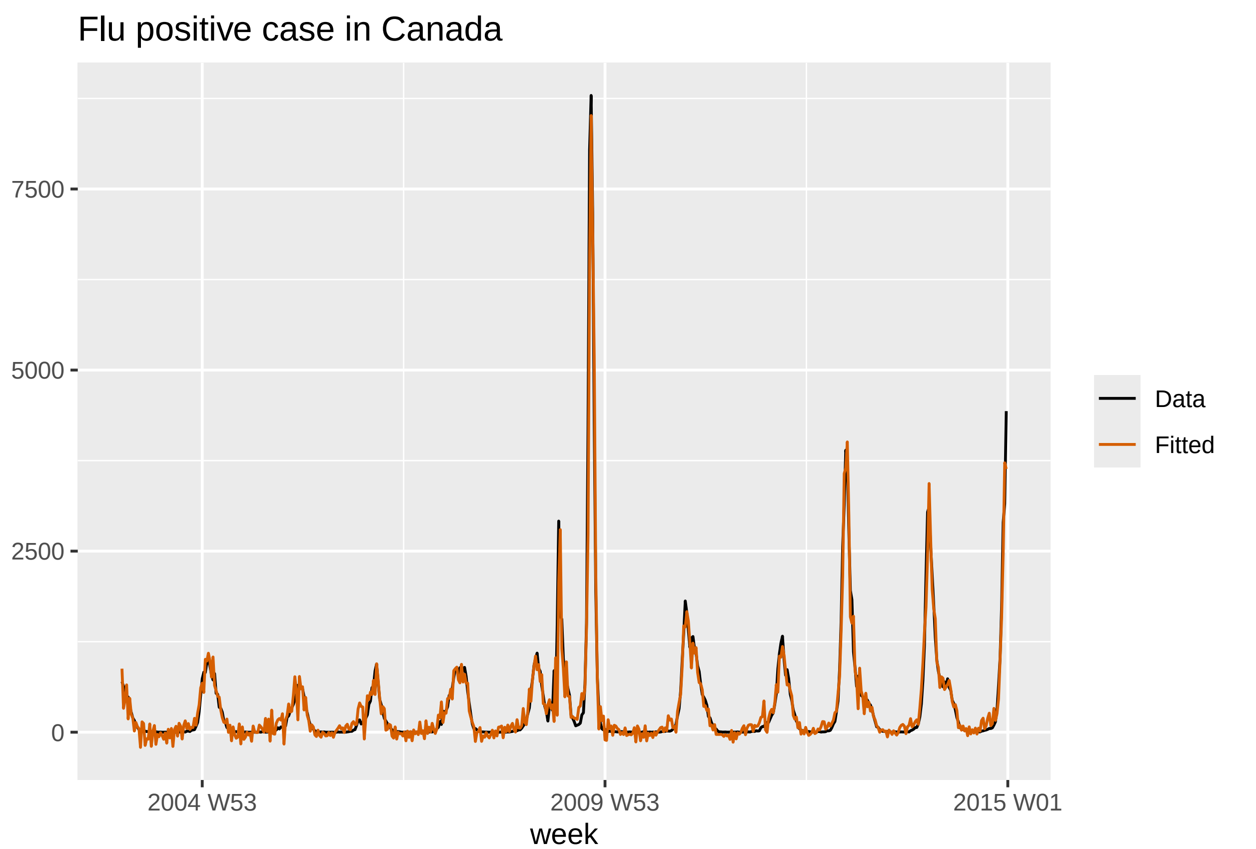
```
flu_ts %>%
  select(-Date) %>%
  gather(key='variable', value='value') %>%
  ggplot(aes(y=value, x=week, group=variable, colour=variable)) +
  geom_line() + facet_grid(variable ~ ., scales='free_y') +
  labs(y="",title ="Flu positive case in Canada") +
  guides(colour="none")
```



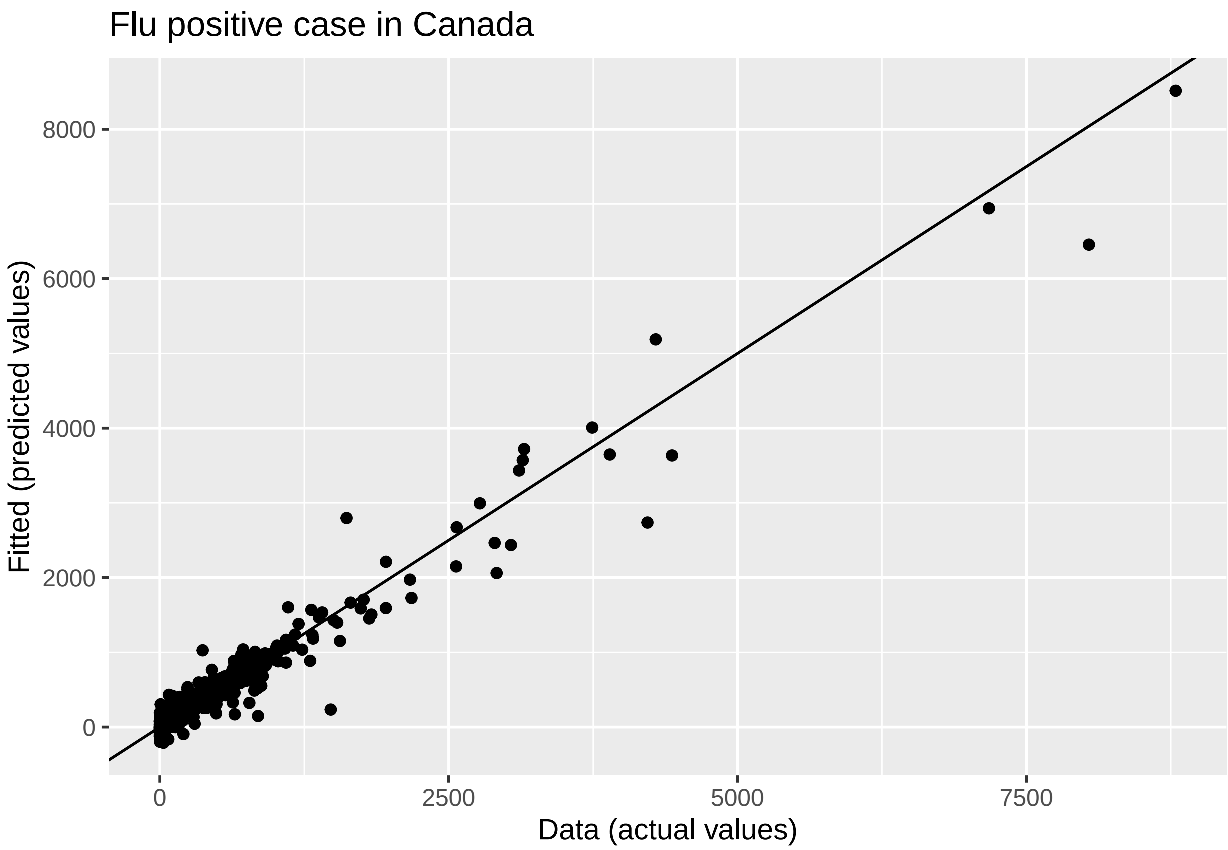

```r
fit.dyn <- flu_ts %>% model(ARIMA(FluPos ~ GFT.Canada+ tem_CA+ swineflu +trend()))
report(fit.dyn)
```

```
## Series: FluPos
## Model: LM w/ ARIMA(1,0,3)(0,0,1)[52] errors
##
## Coefficients:
##          ar1     ma1     ma2     ma3    sma1  GFT.Canada  tem_CA  swineflu
##       0.6026  0.6346  0.4918  0.3983  0.1040      0.5730  6.5610   36.5495
## s.e.  0.0490  0.0580  0.0549  0.0451  0.0399      0.0303  2.9456  148.1665
##       trend()  intercept
##        0.4732  -807.4670
## s.e.   0.5011   119.4874
##
## sigma^2 estimated as 35835:  log likelihood=-3820.54
## AIC=7663.08   AICc=7663.55   BIC=7710.96
```

```r
# Check the fitted values
augment(fit.dyn) |>
  ggplot(aes(x = week)) +
  geom_line(aes(y = FluPos, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(y = NULL,
    title = "Flu positive case in Canada"
  ) +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```



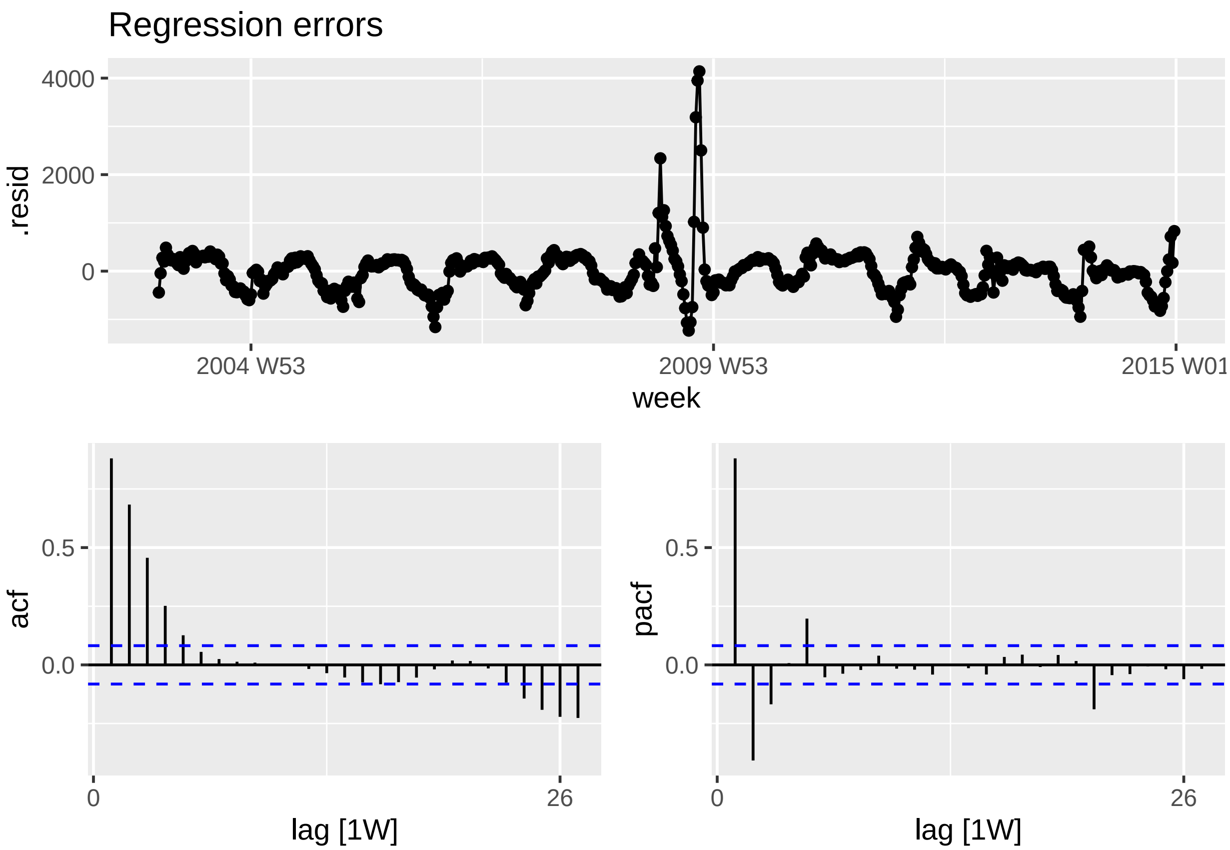

```r
augment(fit.dyn) |>
  ggplot(aes(x = FluPos, y = .fitted)) +
  geom_point() +
  labs(
    y = "Fitted (predicted values)",
    x = "Data (actual values)",
    title = "Flu positive case in Canada"
  ) +
  geom_abline(intercept = 0, slope = 1)
```



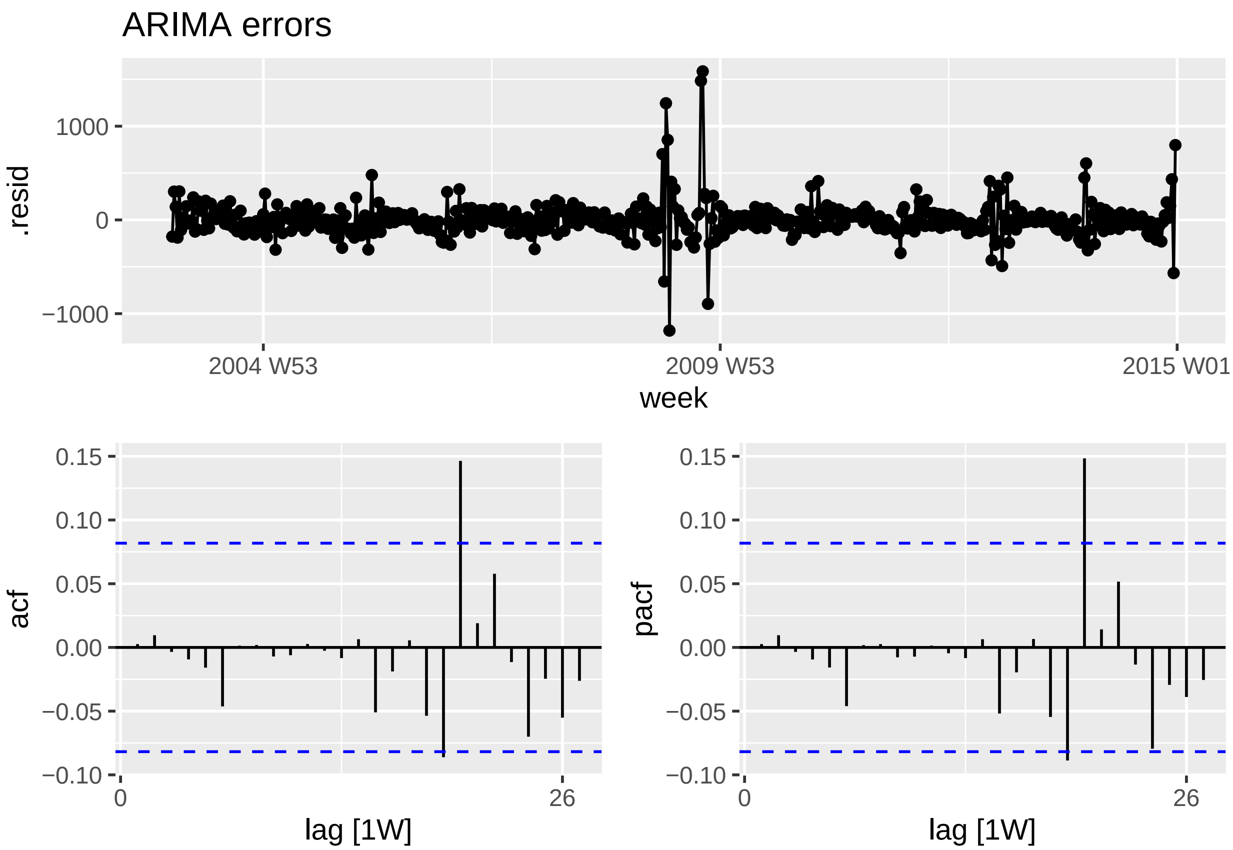
Residual diagnostics

Recall that, in dynamic regression we have two errors: regression error (η_t, will show some correlation) and Arima error (ϵ_t, will show no significant correlation).

```
# regression error
residuals(fit.dyn, type='regression') %>%
  gg_tsdisplay(.resid, plot_type = 'partial') +
  labs(title = "Regression errors")
```




```r
# Arima error
residuals(fit.dyn, type='innovation') %>%
  gg_tsdisplay(.resid, plot_type = 'partial') +
  labs(title = "ARIMA errors")
```



Residuals seems white noise, means no significant correlation left!

```
augment(fit.dyn) |>
  features(.innov, ljung_box,lag = 52)
```

```
## # A tibble: 1 x 3
##   .model                                              lb_stat lb_pvalue
##   <chr>                                                 <dbl>     <dbl>
## 1 ARIMA(FluPos ~ GFT.Canada + tem_CA + swineflu + trend())    43.7     0.787
```

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.

Comment: Overall, it shows that the dynamic regression perform better than the time series linear regression.

Interactive graphics for Time Series data:

We are trying to developed interactive shiny graphics to address these issues for several of the most common time series data analyses. The interactive shiny graphics is used to generate an interactive visualization environment that contains several distinct graphics, many of which are updated in response to user input. These visualizations reduce the burden of exploratory analyses and can serve as a useful tool for the communication of results to non-statisticians.

Link (still under working)

<https://syedrizvi05.shinyapps.io/TimeSeries/>

Acknowledgement:

This document was prepared with the help of a research student, *Syed Jafar Rizvi*, MSc Student, Dept. of Community Health and Epidemiology, University of Saskatchewan. Fell free to knock Rizbi at *jafar.rizvi@usask.ca* if you have any question.