TS_Project_HTS

Chinta

2023-03-03

0. Import dependencies.

1. Load the chickenpox data.

Load the data, get the region-county data, join the chickenpox cases with region-county data, and make it a monthly structure.

```
hch <- read.csv("hungary_chickenpox.csv")  #read the data

hch_r_c <- read.csv("hungary_rgn_county.csv") #get the region-counties info

names(hch_r_c) <- c("region", "rgn","county") #rename the columns

hch.df <- data.frame(hch) #Change to data frame

hch.df <- hch.df  %>%
  pivot_longer (cols =! Date,names_to = "county",
              values_to ="cases_count") #Unpivot the data

hch.df <- hch.df %>% left_join(hch_r_c) #Join the cases data with region-counties data

hch.df <- hch.df %>% select ( Date, region, rgn, county, cases_count) #arrange the columns

hch.df$Date <- as.Date(hch.df$Date, format="%d/%m/%Y") #format the date using as.Date

#hch.df <- hch.df %>% unite ("rgn", rgn:county, sep="") #combine the region and county columns

hch.df <- hch.df %>% select ( Date, region,  county, cases_count) #arrange the columns

hch.df$Date <- yearmonth(hch.df$Date) #format the Date columns to change the date from weekly to monthl

hch.df <- hch.df %>% group_by(Date, region, county) %>% summarise (cases_count = sum(cases_count))
```

2. Create a training and a test tsibble.

Creating the training and test tsibble. Dates less than December 2013 are part of training dataset, while above 12/2013 is test dataset. Additionally, use aggregate_key function to create country-level cases count.

```
hch.tsb <- as_tsibble(hch.df, key=c(region, county), index = Date) #create a tsibble object

#Check for gaps in the data
scan_gaps(hch.tsb) %>%
  count(Date)
```

```
## # A tibble: 0 x 3
## # Groups:   Date, region [0]
## # ... with 3 variables: Date <mth>, region <chr>, n <int>
```

```
hch.tsb_agg <- tsibble(hch.df, key = c(region, county), index = Date) %>%
  aggregate_key(region/county, cases_count = sum(cases_count))

hch.tsb_agg_train <- hch.tsb_agg %>% filter(year(Date) <= 2012)
hch.tsb_agg_test <- hch.tsb_agg %>% filter(year(Date) > 2012)
```

2.1 Process Steps

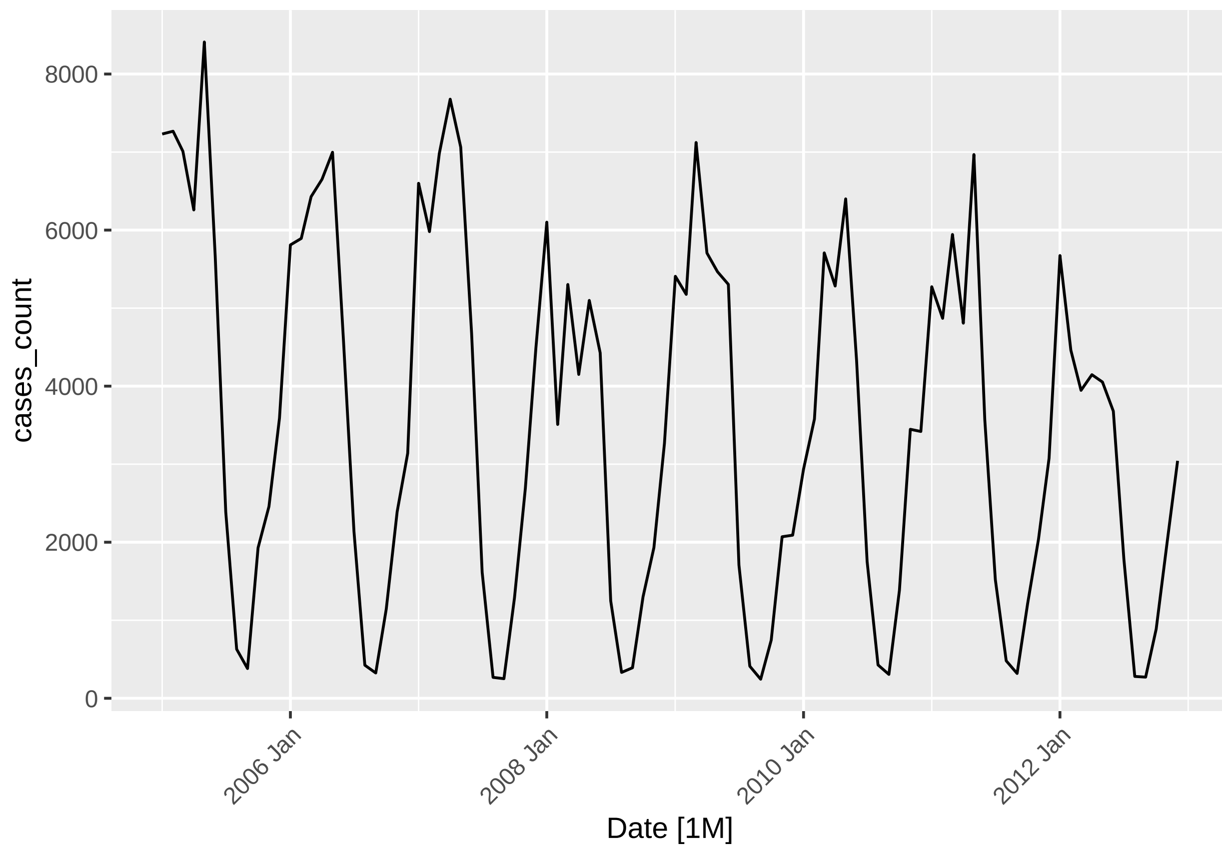
- *data %>% aggregate_key() %>% model() %>% reconcile() %>% forecast()*

3. Plot aggregate (country level) and Central Hungary training data.

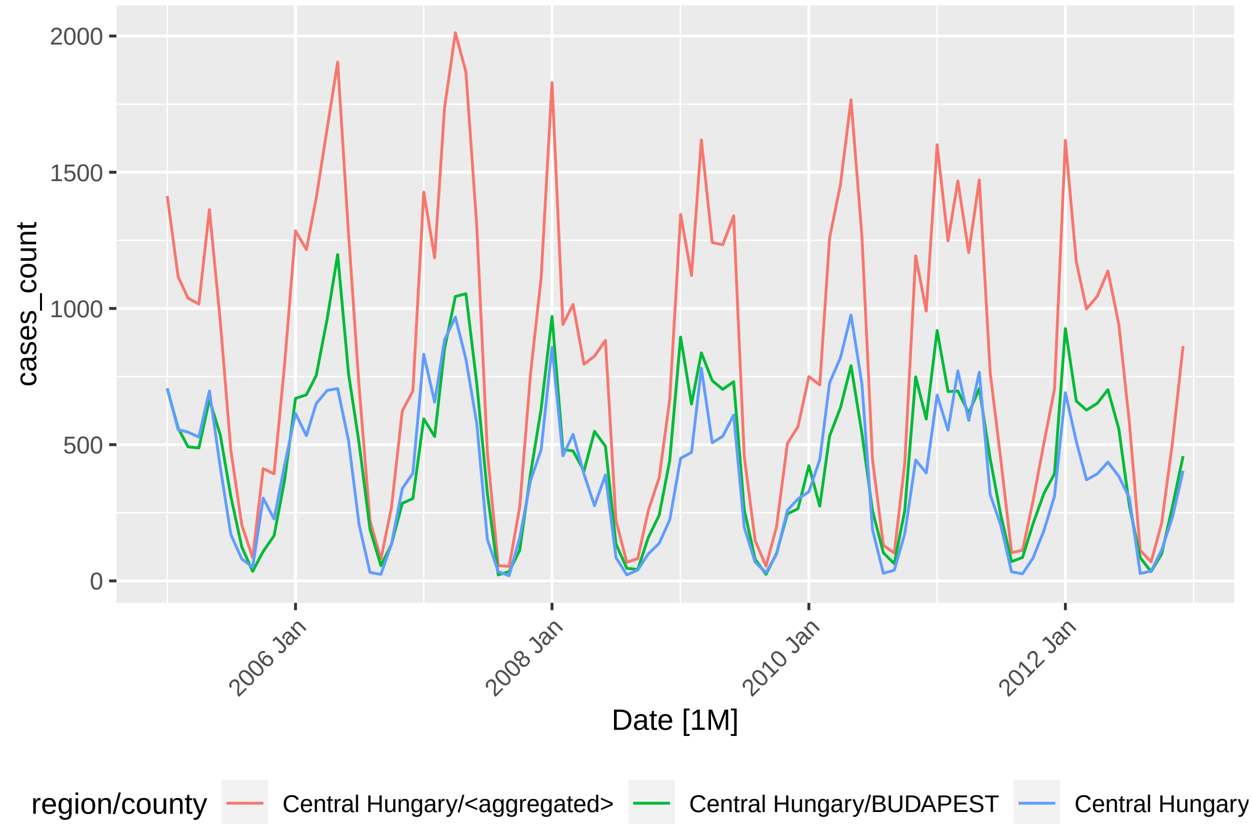
Plot the aggregate and Central Hungary training data to understand data. We see that Central Hungary has pretty heavy influence on the country-level data.

- The variable that we'd like to estimate is the number of cases represented by the 'cases_count' variable. The plot reveals that weak trends and high seasonality are apparent.

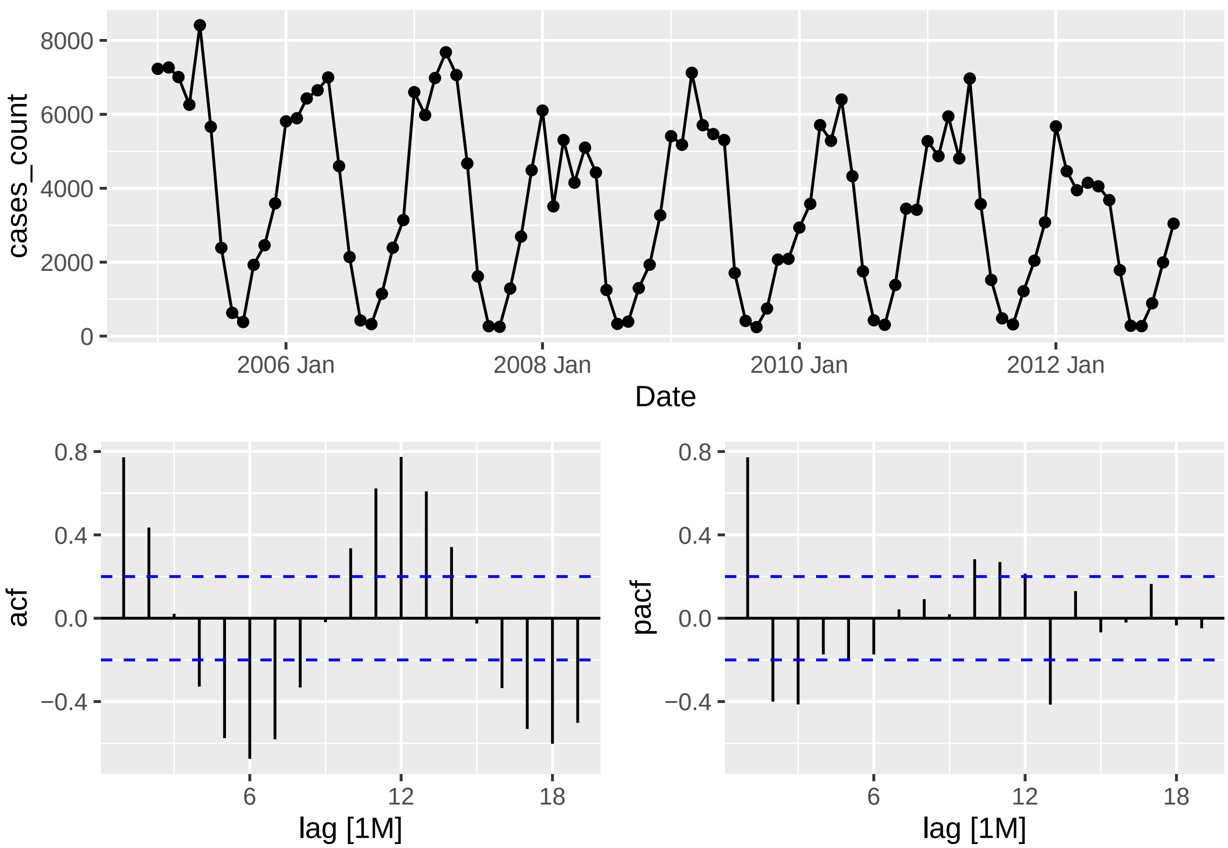
```
#Country-level training data
hch.tsb_agg_train %>%
  filter(is_aggregated(region)) %>%
  autoplot(cases_count) +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  ) #plot the output
```



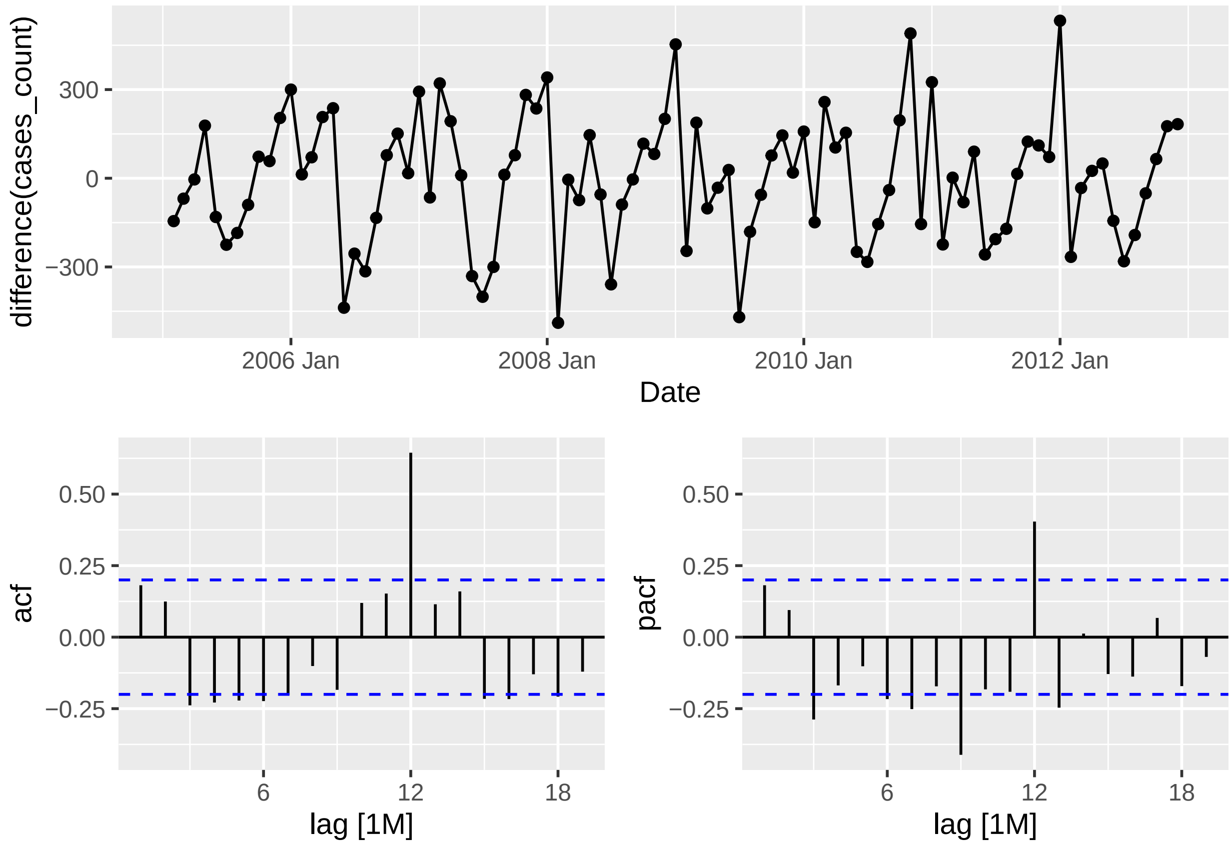

```r
#Central Hungary region training data
hch.tsb_agg_train %>%
  filter(region == 'Central Hungary') %>%
  autoplot(cases_count) +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  ) #plot the output
```




```
#Plot ACF and PACF at country and county levels.
hch.tsb_agg_train %>% filter(is_aggregated(region)) %>%  gg_tsdisplay(cases_count, plot_type='partial')
```




```
hch.tsb_agg_train %>% filter(county == 'BUDAPEST') %>%  gg_tsdisplay(difference(cases_count), plot_type=
```



4. Model ETS, ARIMA, TSLM, and reconcile using bottom_up, top_down, and minimum trace methodlogies.

```
fit_all <- hch.tsb_agg_train %>%
  model(ets = ETS(cases_count),
        arima = ARIMA((cases_count) ~ pdq(1,0,0) + PDQ(2,1,0)),
        lm = TSLM((cases_count) ~ season()),
        `Seasonal naïve` = SNAIVE(cases_count)) %>%
  reconcile(bu_ets = bottom_up(ets),
            td_ets = top_down(ets),
            min_trace_ets = min_trace(ets, "mint_shrink"),
            bu_arima = bottom_up(arima),
            td_arima = top_down(arima),
            min_trace_arima = min_trace(arima, "mint_shrink"),
            bu_tslm = bottom_up(lm),
            td_tslm = top_down(lm),
            min_trace_tslm = min_trace(lm, "mint_shrink")
            )
```

```
fit_all  %>%
  filter(is_aggregated(region)|county=='BUDAPEST') %>%
  select(region,county,ets,arima,lm) %>%
  pivot_longer(-c(region,county), names_to = "Model name",
                         values_to = "cases_count") %>%
```

```
  kable()
```

4.1 Base Models selected.

region	county	Model name	cases_count
Central Hungary	BUDAPEST	ets	<ETS(M,N,M)>
Central Hungary	BUDAPEST	arima	<ARIMA(1,0,0)(2,1,0)[12]>
Central Hungary	BUDAPEST	lm	
		ets	<ETS(M,N,M)>
		arima	<ARIMA(1,0,0)(2,1,0)[12] w/ drift>
		lm	

5. Analyze the IC metrics of ETS, ARIMA, and related reconcile() functions

- Country-level IC metrics:

```
fit_all %>%
  filter(is_aggregated(region))%>%
  select(region, county,ets, arima,`Seasonal naïve`,bu_ets, bu_arima, td_ets, td_arima,lm,bu_tslm,td_tsl
  glance() %>%
  transmute(.model,
            region = if_else(is_aggregated(region), 'country-level',as.character(region)),
            county, AICc, AIC, BIC )  %>%
  arrange(AICc) %>%
  kable()
```

.model	region	county	AICc	AIC	BIC
lm	country-level		1319.532	1315.093	1348.430
bu_tslm	country-level		1319.532	1315.093	1348.430
td_tslm	country-level		1319.532	1315.093	1348.430
min_trace_tslm	country-level		1319.532	1315.093	1348.430
arima	country-level		1370.271	1369.502	1381.656
bu_arima	country-level		1370.271	1369.502	1381.656
td_arima	country-level		1370.271	1369.502	1381.656
ets	country-level		1644.731	1638.731	1677.196
bu_ets	country-level		1644.731	1638.731	1677.196
td_ets	country-level		1644.731	1638.731	1677.196
Seasonal naïve	country-level		NA	NA	NA

- Budapest IC metrics:

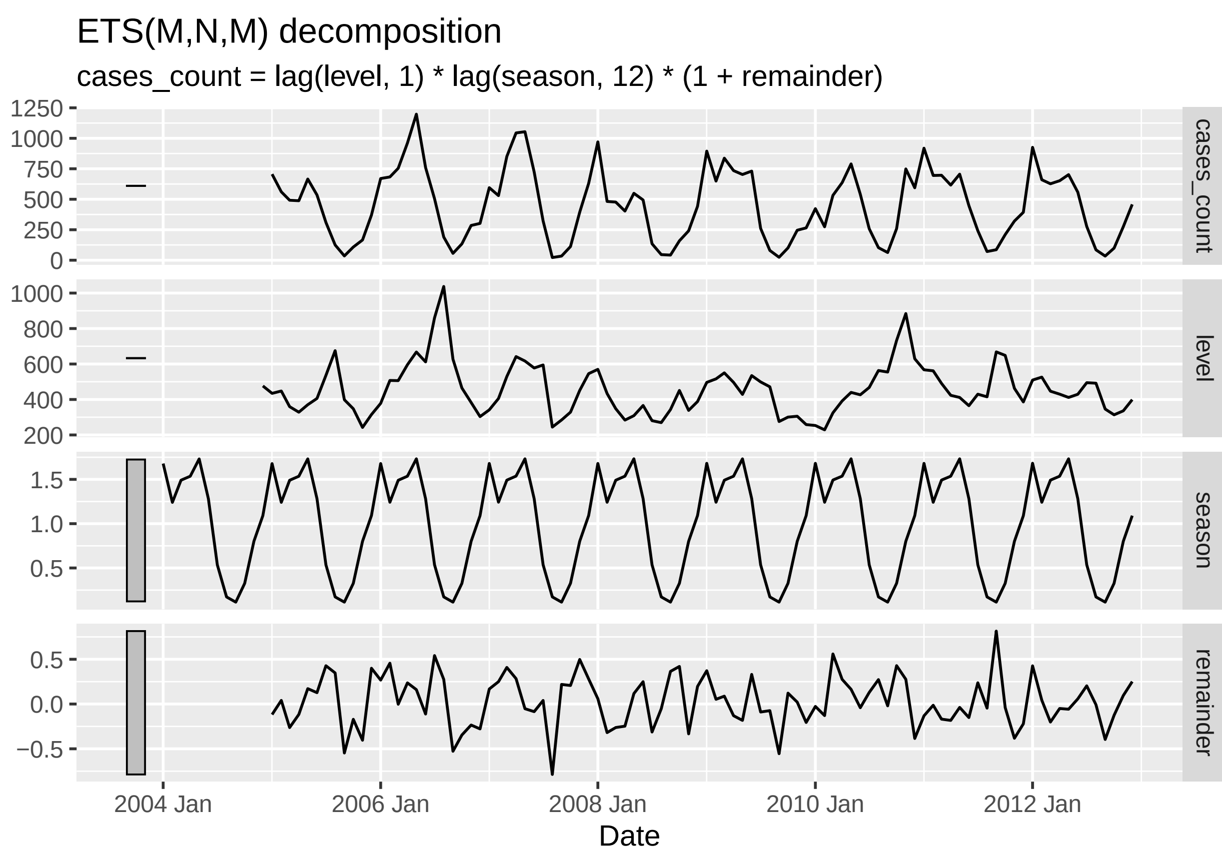
```
fit_all %>%
  filter(county=='BUDAPEST')%>%
  select(region, county,ets, arima,bu_ets, bu_arima, td_ets,
         td_arima,lm,bu_tslm,td_tslm,min_trace_tslm ) %>%
  glance() %>%
  transmute(.model,
            region = if_else(is_aggregated(region), 'country-level',as.character(region)),
            county, AICc, AIC, BIC )  %>%
  arrange(AICc) %>%
  kable()
```

.model	region	county	AICc	AIC	BIC
lm	Central Hungary	BUDAPEST	974.739	970.300	1003.637
bu_tslm	Central Hungary	BUDAPEST	974.739	970.300	1003.637
td_tslm	Central Hungary	BUDAPEST	974.739	970.300	1003.637
min_trace_tslm	Central Hungary	BUDAPEST	974.739	970.300	1003.637
arima	Central Hungary	BUDAPEST	1052.653	1052.146	1061.870
bu_arima	Central Hungary	BUDAPEST	1052.653	1052.146	1061.870
td_arima	Central Hungary	BUDAPEST	1052.653	1052.146	1061.870
ets	Central Hungary	BUDAPEST	1344.975	1338.975	1377.440
bu_ets	Central Hungary	BUDAPEST	1344.975	1338.975	1377.440
td_ets	Central Hungary	BUDAPEST	1344.975	1338.975	1377.440

6. Get the decomposition of the ets

- Budapest (county level) decomposition:

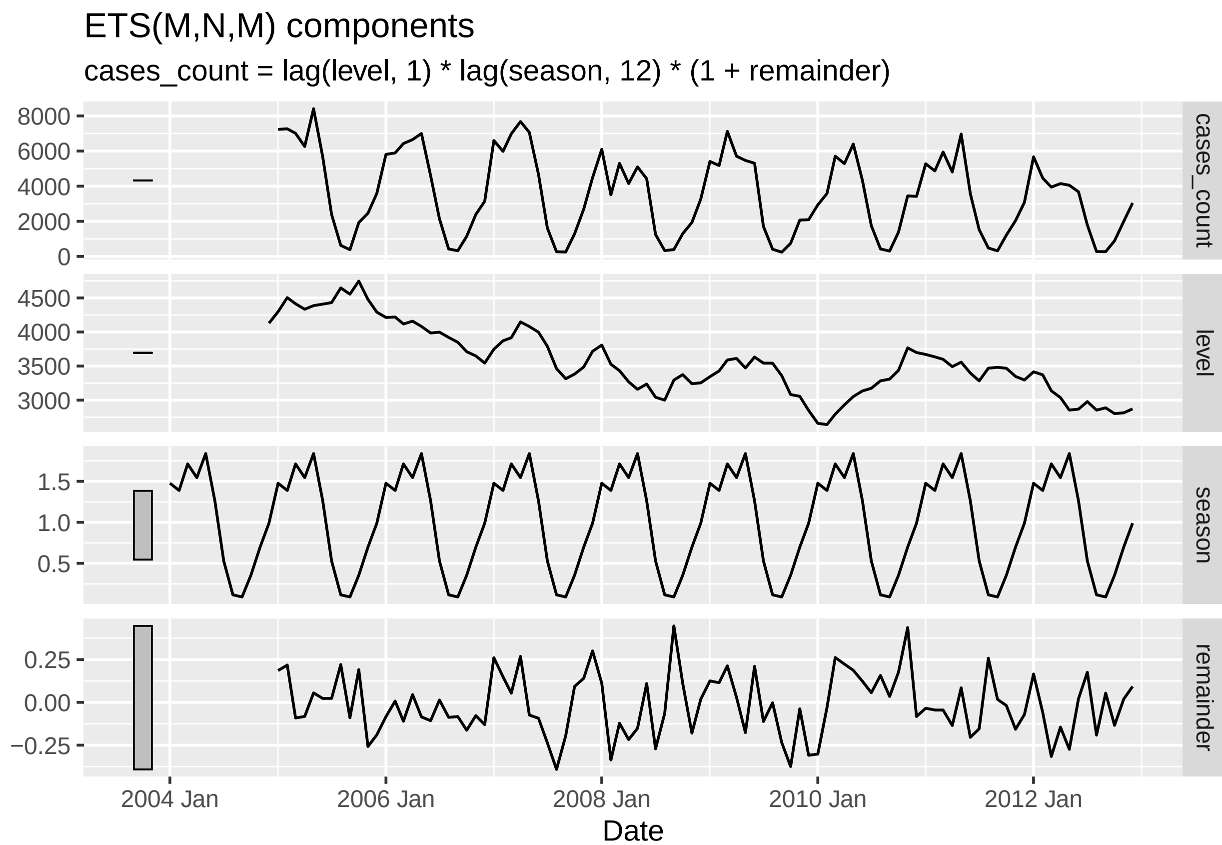
```
fit_all %>%
  filter(county=='BUDAPEST') %>%
  select(td_ets) %>%
  components() %>%
  autoplot()
```



- Country-level decomposition:

```
fit_all %>%
  filter(is_aggregated(region)) %>%
```

```
  select(ets) %>%
  components() %>%
  autoplot() +
  labs(title = "ETS(M,N,M) components")
```



7. Run the forecast of next 2 years on the model:

```
fc_all <- fit_all %>%
  fabletools::forecast(h=24)
```

7.1 Get the accuracy metrics for comparison:

- Country-level accuracy metrics:

```
fc_all %>%
  fabletools::accuracy(
    data = hch.tsb_agg,
    measures = list(rmse = RMSE, mase = MASE, mape = MAPE, mae=MAE)
  ) %>%
  filter(is_aggregated(region)) %>%
  arrange(rmse) %>%
  transmute(.model,
            region = if_else(is_aggregated(region),
                             'country-level',
                             as.character(region)),
            county, rmse, mase, mape, mae) %>%
```

```
  kable()
```

.model	region	county	rmse	mase	mape	mae
ets	country-level		632.3692	0.6092038	21.35646	451.4708
td_ets	country-level		632.3692	0.6092038	21.35646	451.4708
min_trace_ets	country-level		641.9707	0.6257824	21.92544	463.7569
bu_ets	country-level		651.9269	0.6409759	22.25146	475.0166
arima	country-level		667.4608	0.7509349	41.25541	556.5053
td_arima	country-level		667.4608	0.7509349	41.25541	556.5053
min_trace_arima	country-level		792.5440	0.7547412	27.81123	559.3261
bu_arima	country-level		836.7337	0.7924584	27.18954	587.2777
Seasonal naïve	country-level		864.3824	0.8280108	25.48327	613.6250
lm	country-level		1201.2503	1.2215928	34.73856	905.3021
min_trace_tslm	country-level		1201.2503	1.2215928	34.73856	905.3021
td_tslm	country-level		1201.2503	1.2215928	34.73856	905.3021
bu_tslm	country-level		1201.2503	1.2215928	34.73856	905.3021

- Budapest (county-level) accuracy metrics:

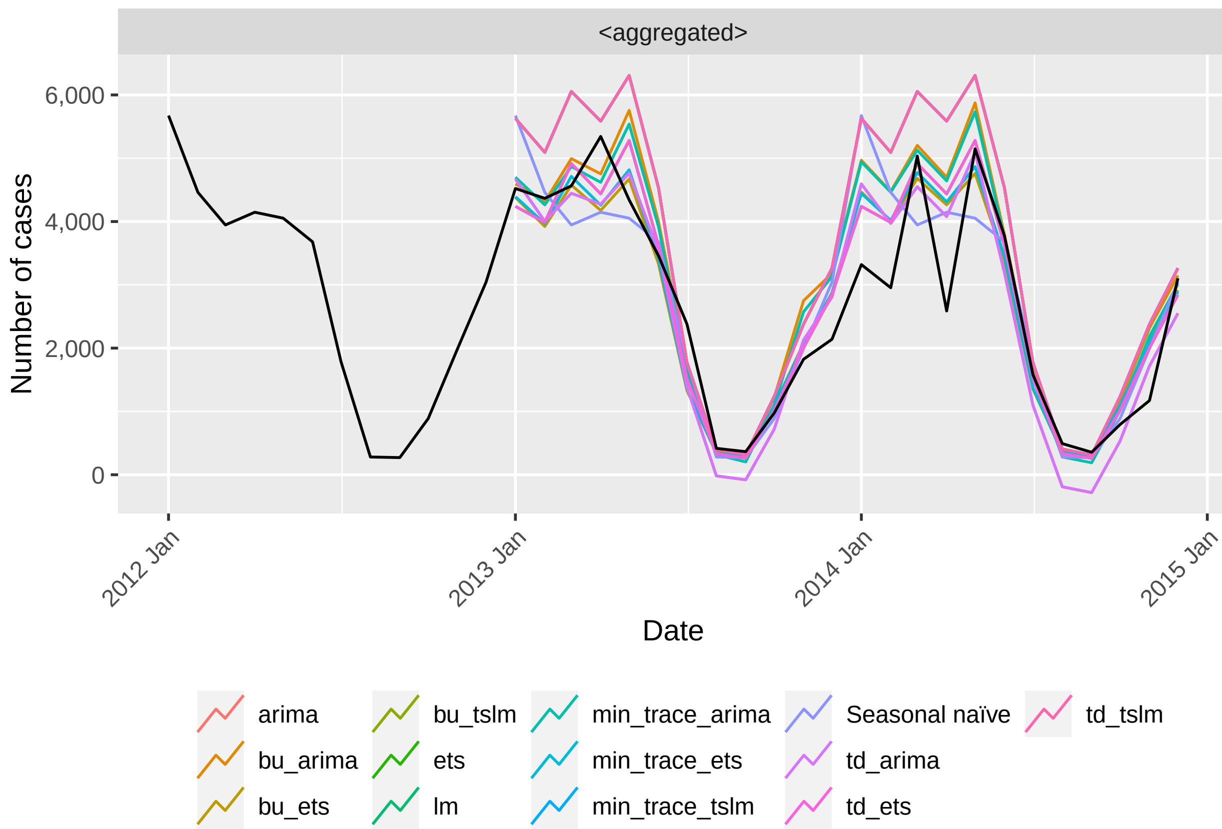
```
fc_all %>%
  fabletools::accuracy(
    data = hch.tsb_agg,
    measures = list(rmse = RMSE, mase = MASE, mape = MAPE, mae=MAE)
  ) %>%
  filter(county=='BUDAPEST') %>%
  arrange(rmse) %>%
  transmute(.model,
            region = if_else(is_aggregated(region),
                             'country-level',
                             as.character(region)),
            county, rmse, mase, mape, mae) %>%
  kable()
```

.model	region	county	rmse	mase	mape	mae
td_ets	Central Hungary	BUDAPEST	136.3707	0.6644929	33.75123	105.0927
min_trace_ets	Central Hungary	BUDAPEST	141.3894	0.6747999	34.33903	106.7228
bu_ets	Central Hungary	BUDAPEST	142.6083	0.6789840	34.88513	107.3846
ets	Central Hungary	BUDAPEST	142.6083	0.6789840	34.88513	107.3846
td_tslm	Central Hungary	BUDAPEST	150.6835	0.6920963	39.59377	109.4583
bu_tslm	Central Hungary	BUDAPEST	150.6835	0.6920963	39.59377	109.4583
lm	Central Hungary	BUDAPEST	150.6835	0.6920963	39.59377	109.4583
min_trace_tslm	Central Hungary	BUDAPEST	150.6835	0.6920963	39.59377	109.4583
td_arima	Central Hungary	BUDAPEST	171.4101	0.8742857	61.82152	138.2725
Seasonal naïve	Central Hungary	BUDAPEST	178.2706	0.7603312	37.63705	120.2500
min_trace_arima	Central Hungary	BUDAPEST	185.8736	0.8356473	50.90250	132.1616
arima	Central Hungary	BUDAPEST	186.2432	0.8359951	50.80321	132.2166
bu_arima	Central Hungary	BUDAPEST	186.2432	0.8359951	50.80321	132.2166

7.2 Plot the forecast at the country-level:

```
autoplot(
  fc_all %>%
```

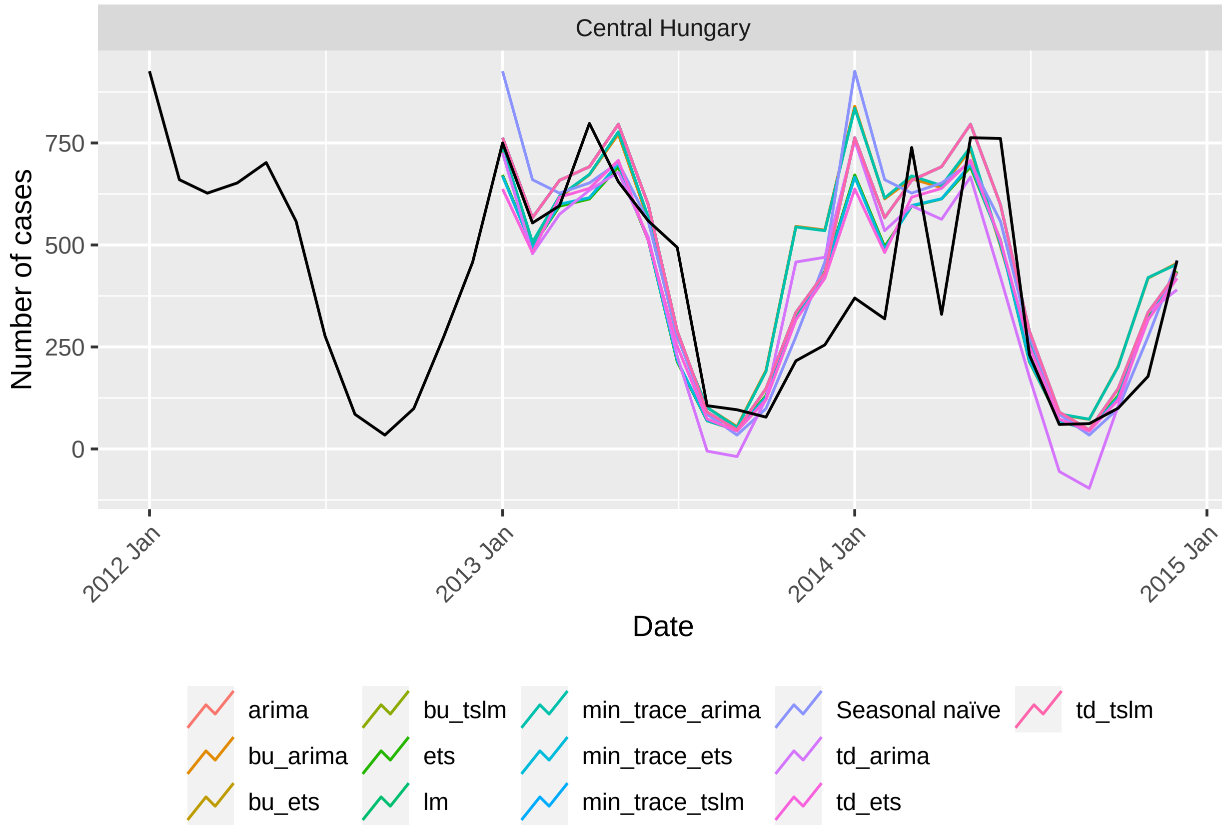
```
    filter( is_aggregated(region)),
  hch.tsb_agg %>%
    ungroup() %>%
    filter(year(Date) >2011), level = NULL) +
  facet_wrap(~region, scales = "free_y") +
  scale_y_continuous(labels = scales::comma_format()) +
  labs(color = "", x = "Date" ,y = "Number of cases") +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  )
```



Plot the forecast at the country-level and all the regions

```
autoplot(
  fc_all %>%
    filter(county=='BUDAPEST' ),
  hch.tsb_agg %>%
    ungroup() %>%
    filter(year(Date) >2011), level = NULL) +
  facet_wrap(~region, scales = "free_y") +
  scale_y_continuous(labels = scales::comma_format()) +
  labs(color = "", x = "Date" ,y = "Number of cases") +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
```

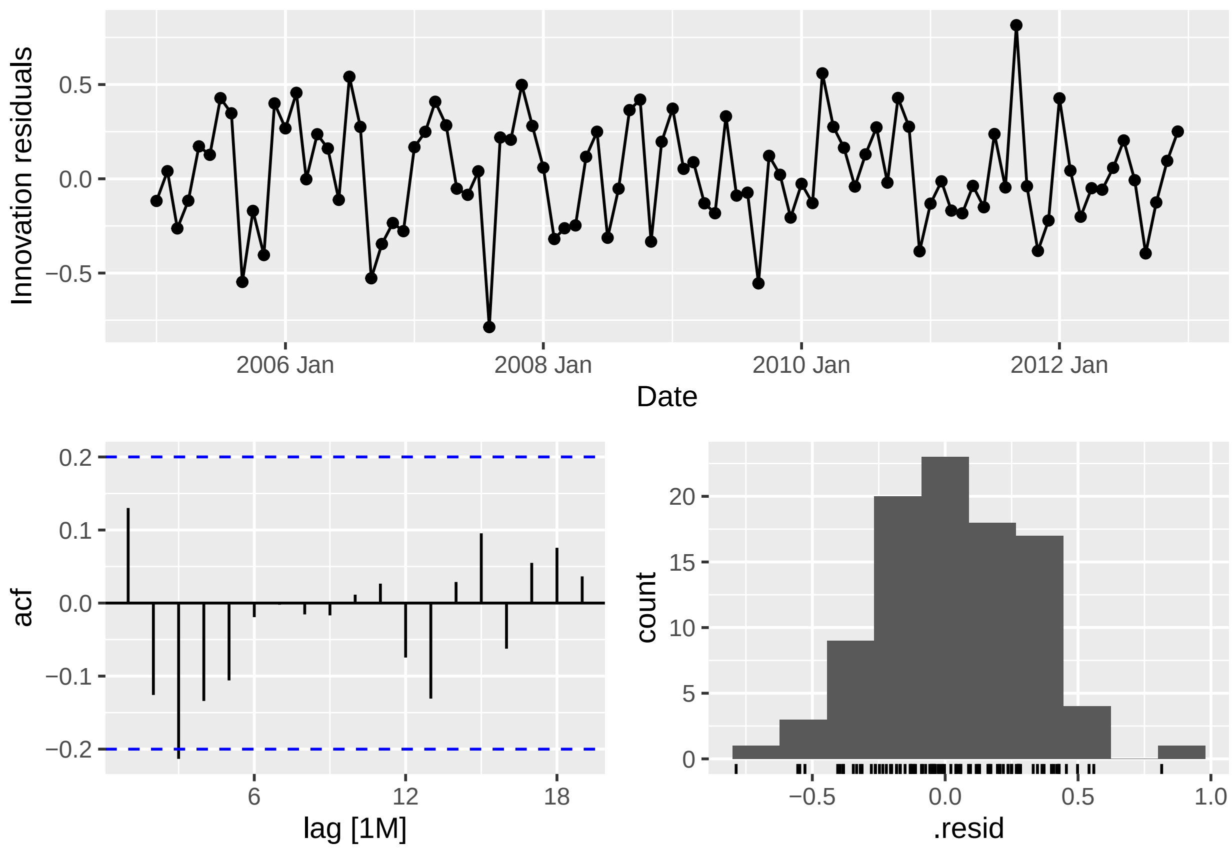
```
)
```



7.3 Get the ACF plot of the residuals of top two models based on RMSE at county level:

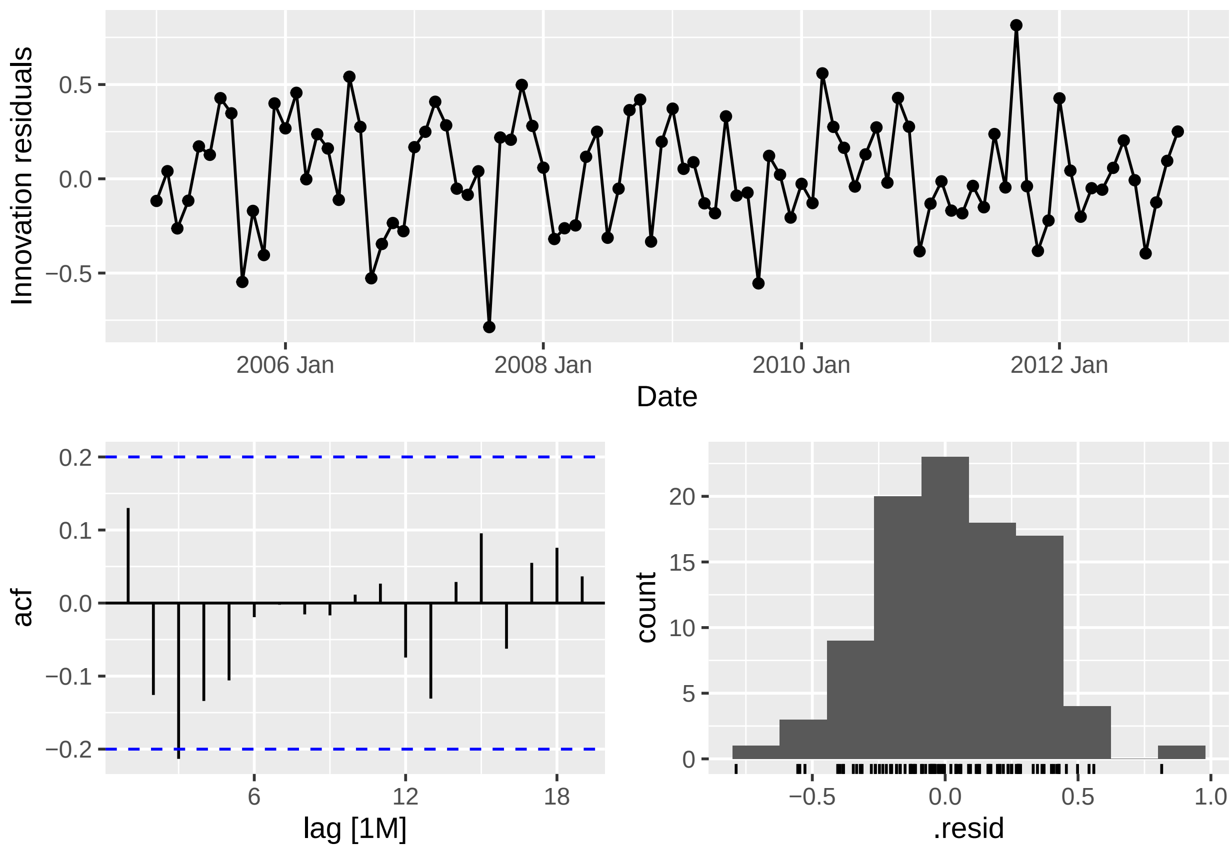
- ETS based on Top-down approach:

```
fit_all  %>%
   filter(county=='BUDAPEST' ) %>%
  select(td_ets)  %>%
  gg_tsresiduals()
```



- Base ETS plot:

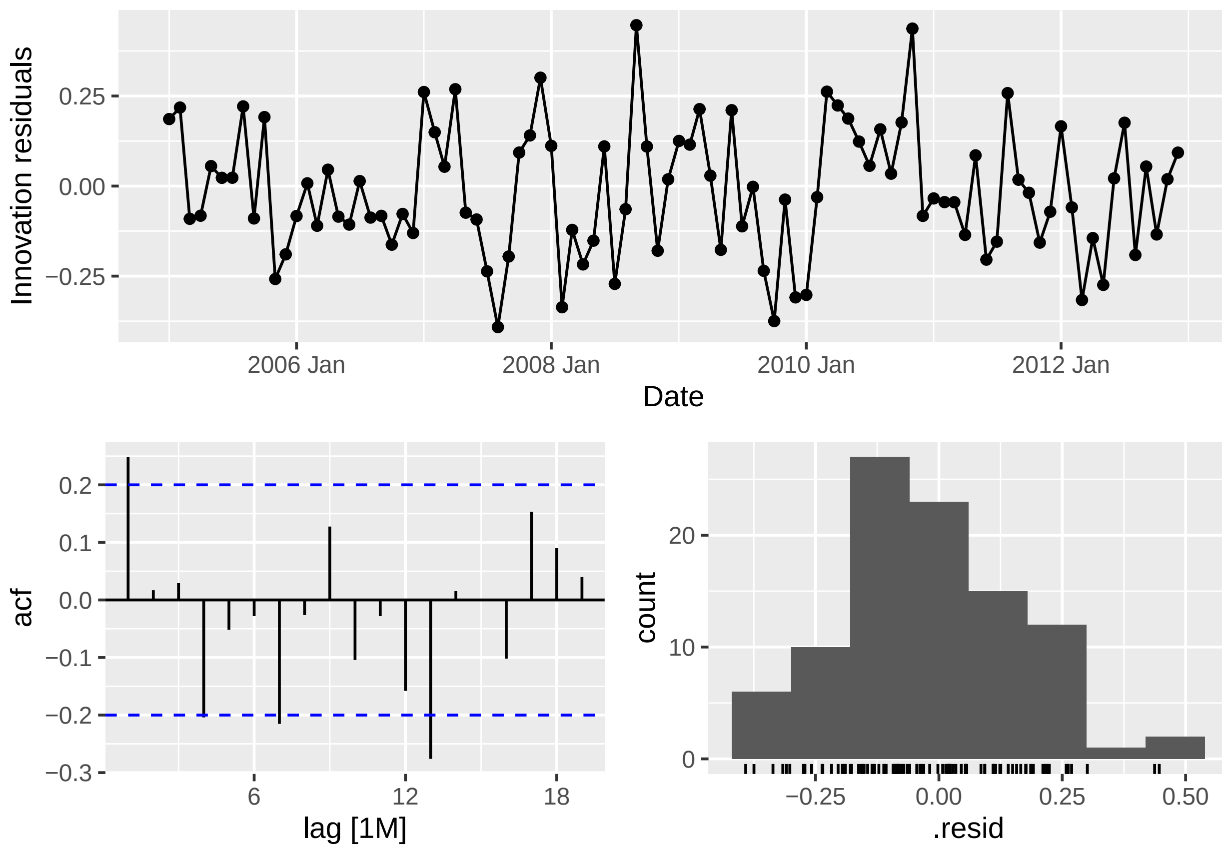
```
fit_all  %>%
   filter(county=='BUDAPEST' ) %>%
  select(min_trace_ets)  %>%
  gg_tsresiduals()
```



7.4 Get the ACF plot of the residuals of top two models based on RMSE at COUNTRY level:

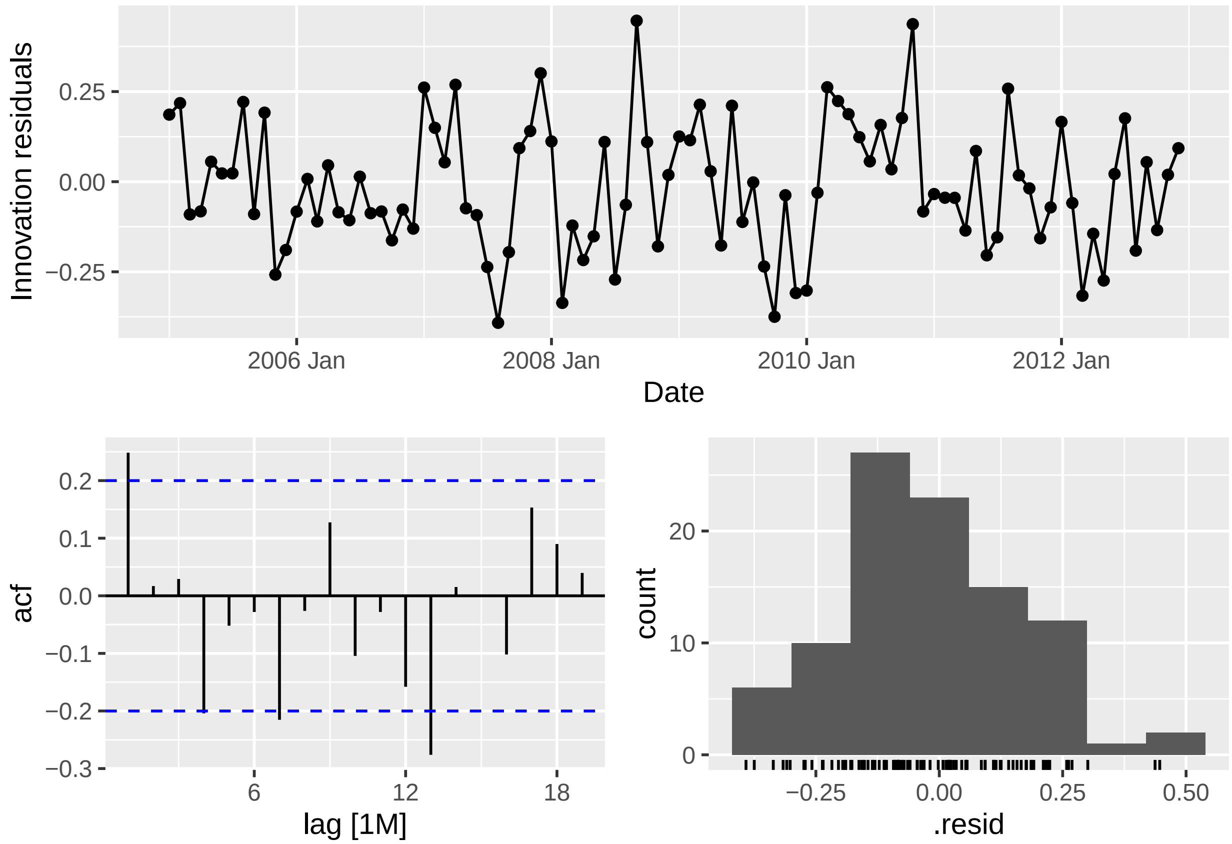
- Base ETS:

```
fit_all  %>%
   filter(is_aggregated(region) ) %>%
  select(ets)  %>%
  gg_tsresiduals()
```



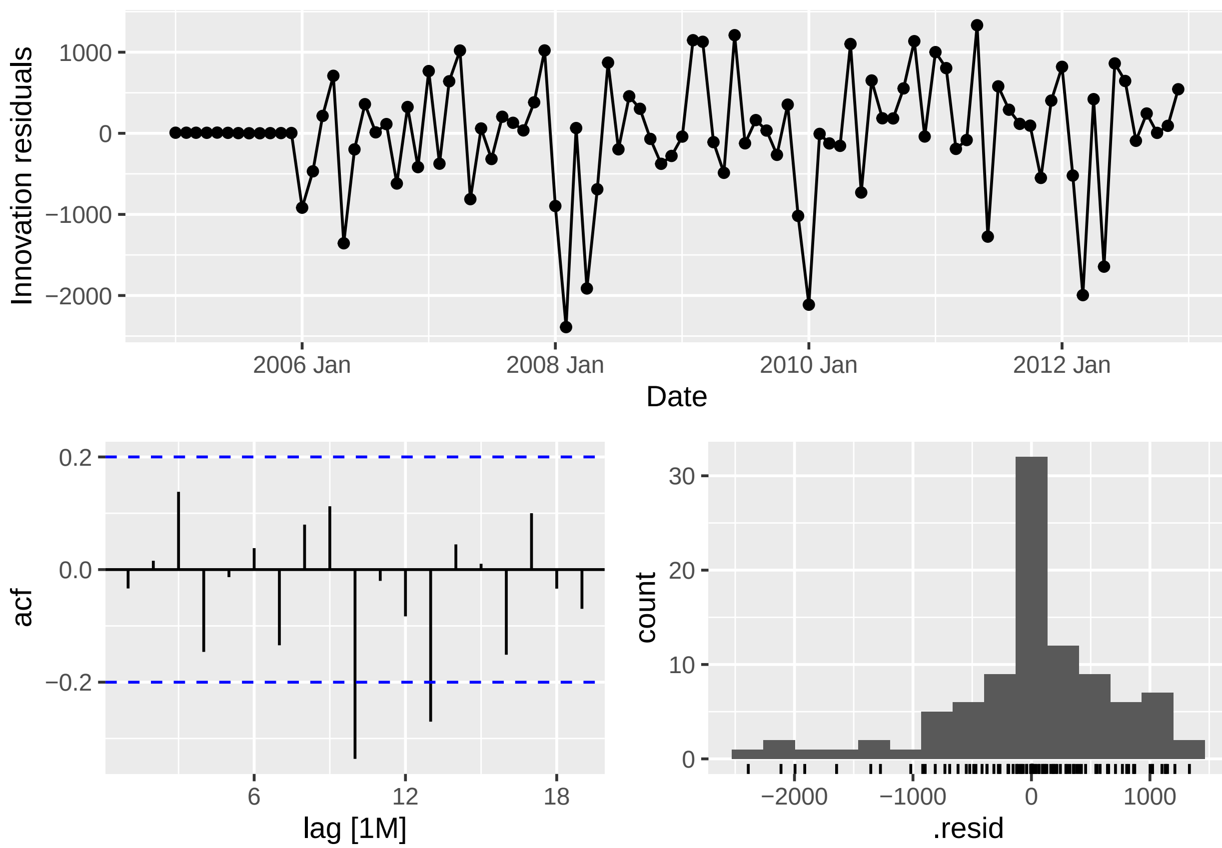
- ETS based on Top-down approach:

```
fit_all  %>%
   filter(is_aggregated(region) ) %>%
  select(td_ets)  %>%
  gg_tsresiduals()
```



- Base ARIMA:

```
fit_all  %>%
   filter(is_aggregated(region) ) %>%
  select(arima)  %>%
  gg_tsresiduals()
```



7.5 Conduct Ljung-Box test on the fit:

- Ljung-Box test: we see a p-value much smaller than 0.05, thus we can reject the null hypothesis, indicating the time series does contain an autocorrelation.

```
augment(fit_all)  %>%
  filter(is_aggregated(region) | county=='BUDAPEST') %>%
  filter(.model=='ets'|.model=='arima'|.model=='td_ets'|.model=='min_trace_ets')  %>%
  features(.resid, ljung_box,lag=12) %>%
  kable()
```

region	county	.model	lb_stat	lb_pvalue
Central Hungary	BUDAPEST	arima	8.66519	0.7312303
Central Hungary	BUDAPEST	ets	11.36668	0.4977826
Central Hungary	BUDAPEST	min_trace_ets	11.36668	0.4977826
Central Hungary	BUDAPEST	td_ets	11.36668	0.4977826
		arima	21.55678	0.0427981
		ets	22.16698	0.0356904
		min_trace_ets	22.16698	0.0356904
		td_ets	22.16698	0.0356904

8. Load the weekly chickenpox data.

Load the data, get the region-county data, join the chickenpox cases with region-county data, and keep it weekly data structure.

```r
# hch <- read.csv("hungary_chickenpox.csv")  #read the data
#
# hch_r_c <- read.csv("hungary_rgn_county.csv") #get the region-counties info
#
# names(hch_r_c) <- c("region", "rgn","county") #rename the columns

hch.df_w <- data.frame(hch) #Change to data frame

hch.df_w <- hch.df_w  %>%
  pivot_longer (cols =! Date,names_to = "county",
                values_to ="cases_count") #Unpivot the data

hch.df_w <- hch.df_w %>% left_join(hch_r_c) #Join the cases data with region-counties data

hch.df_w <- hch.df_w %>% select ( Date, region, rgn, county, cases_count) #arrange the columns

hch.df_w$Date <- as.Date(hch.df_w$Date, format="%d/%m/%Y") #format the date using as.Date

#hch.df_w <- hch.df_w %>% unite ("rgn", rgn:county, sep="") #combine the region and county columns

hch.df_w <- hch.df_w %>% select ( Date, region,  county, cases_count) #arrange the columns

#Considered to be multiple seasonal model, so keeping totals
hch.df_w$Date <- yearweek(hch.df_w$Date) #format the Date columns to change the date from weekly to mon

hch.df_w <- hch.df_w %>% group_by(Date, region, county) %>% summarise (cases_count = sum(cases_count))
```

9. Create a training and a test tsibble.

Creating the training and test tsibble. Dates less than December 2013 are part of training dataset, while above 12/2013 is test dataset. Additionally, use aggregate_key function to create country-level cases count.

```r
hch.tsb_w <- as_tsibble(hch.df_w, key=c(region, county), index = Date) #create a tsibble object

hch.tsb_w_agg <- tsibble(hch.df_w, key = c(region, county), index = Date) %>%
  aggregate_key(region/county, cases_count = sum(cases_count))

hch.tsb_w_agg_train <- hch.tsb_w_agg %>% filter(year(Date) <= 2012)
hch.tsb_w_agg_test <- hch.tsb_w_agg %>% filter(year(Date) > 2012)
```

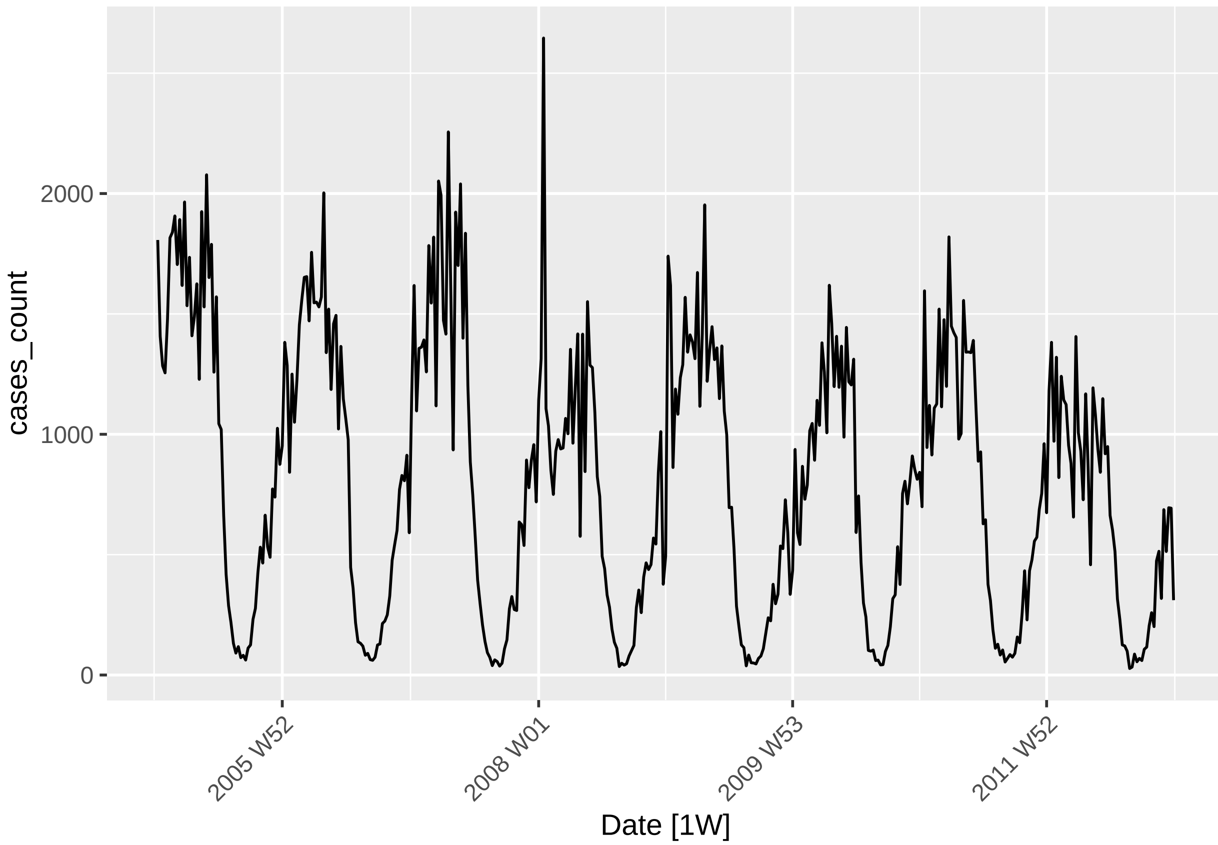
10. Plot aggregate (country level) and Central Hungary training data.

Plot the aggregate and Central Hungary training data to understand data. We see that Central Hungary has pretty heavy influence on the country-level data.

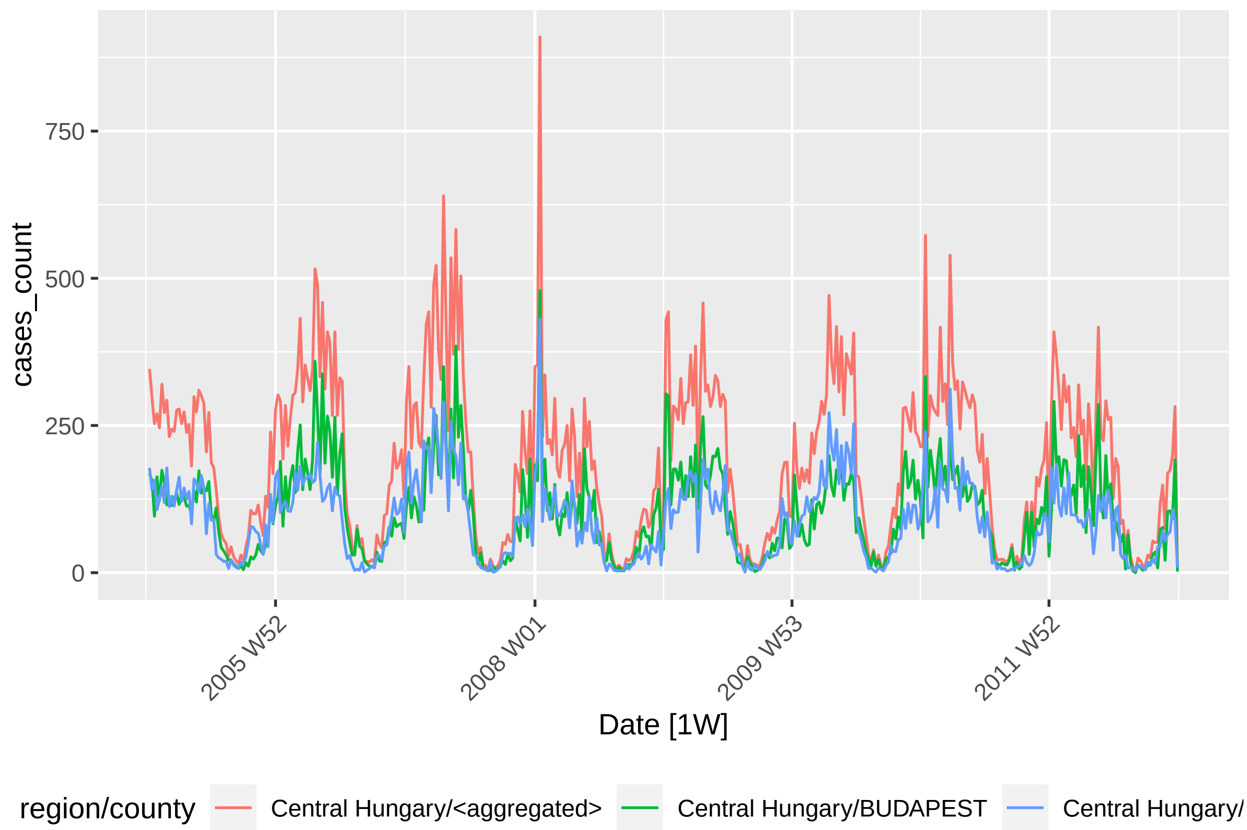
- The variable that we'd like to estimate is the number of cases represented by the 'cases_count' variable. The plot reveals that weak trends and high seasonality are apparent.

```r
#Country-level training data
hch.tsb_w_agg_train %>%
  filter(is_aggregated(region)) %>% #filter(Date>=yearweek('2005 W01') & Date<=yearweek('2005 W52')) %>%
  autoplot(cases_count) +
  theme(
    legend.position = 'bottom',
```

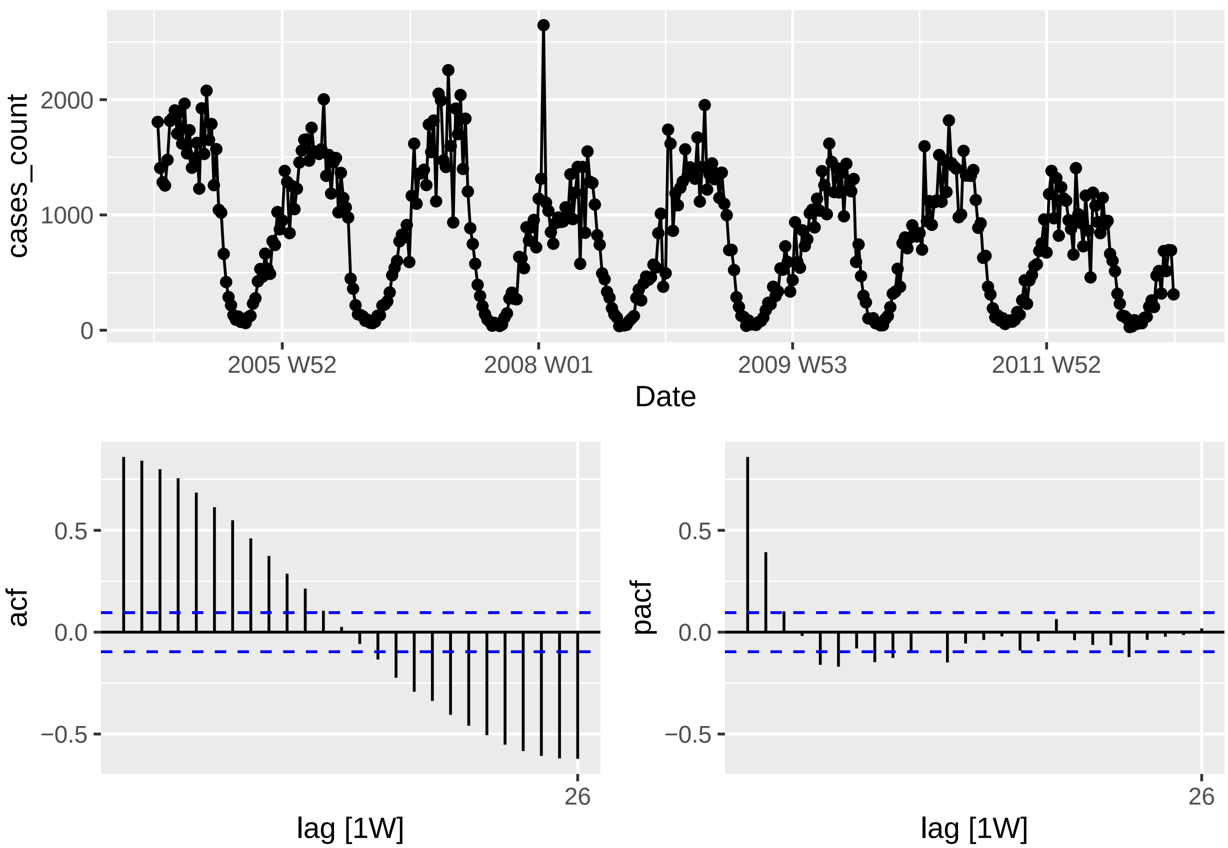
```
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  ) #plot the output
```



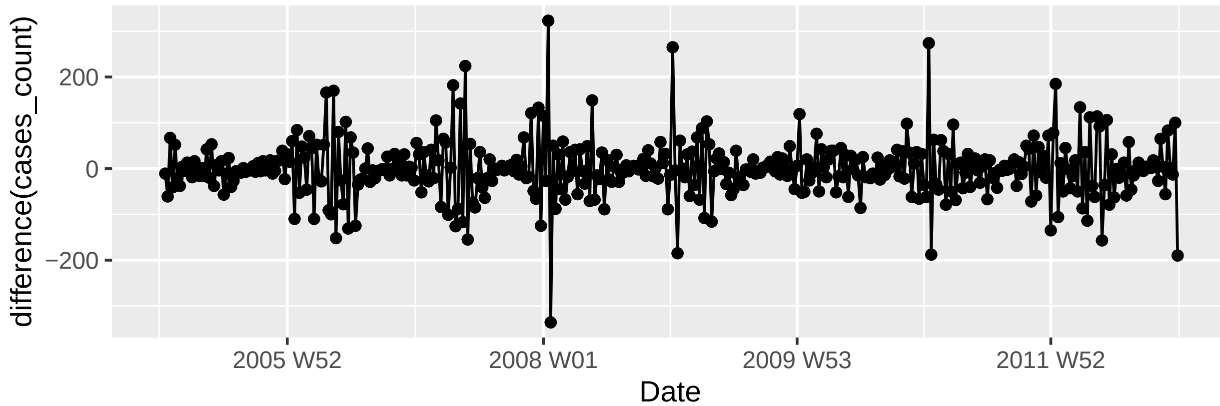

```
#Central Hungary region training data
hch.tsb_w_agg_train %>%
  filter(region == 'Central Hungary') %>%
  autoplot(cases_count) +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  ) #plot the output
```

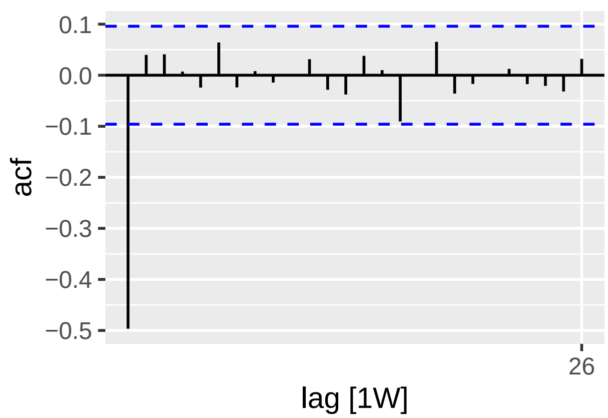


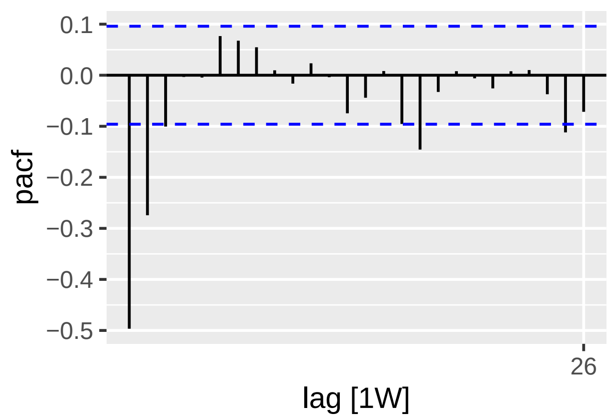

```
#Plot ACF and PACF at country and county levels.
hch.tsb_w_agg_train %>% filter(is_aggregated(region)) %>%  gg_tsdisplay(cases_count, plot_type='partial
```




```
hch.tsb_w_agg_train %>% filter(county == 'BUDAPEST') %>%  gg_tsdisplay(difference(cases_count), plot_ty
```



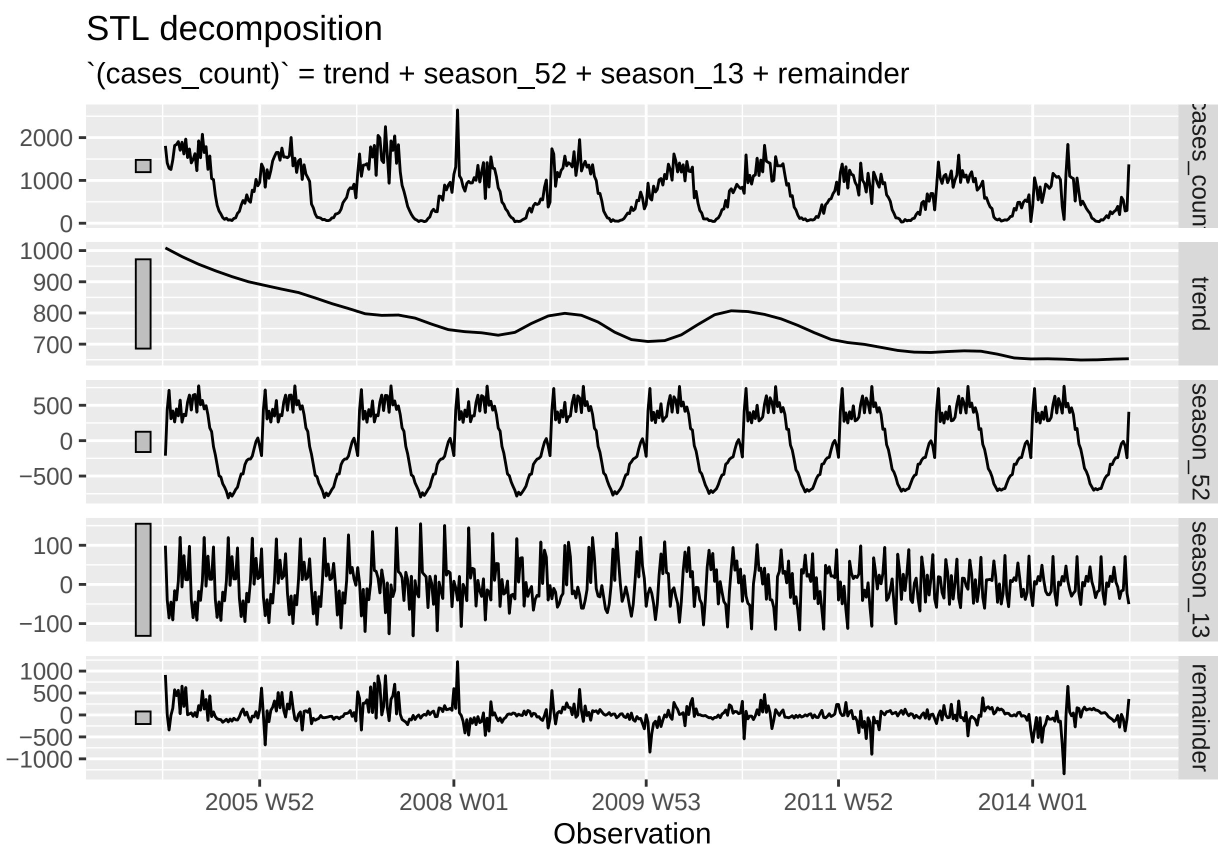




10.1 Plot the decomposition using STL.

- We note there's a strong yearly seasonal pattern, and a weak weekly pattern .

```
hch.tsb_w_agg  %>% filter(is_aggregated(region)) %>%
  model(
    STL((cases_count) ~ season(period = 52) +
          season(period = 13),
        robust = TRUE)
  )  %>%
  components()  %>%
  autoplot() + labs(x = "Observation")
```



11. Model ETS,STL Decomposition (with a non-seasonal method applied to the seasonally adjusted data) and reconcile using bottom_up, top_down, and minimum trace methodlogies.

```
#based on the FPP3 textbook, https://otexts.com/fpp3/complexseasonality.html#stl-with-multiple-seasonal

#
my_dcmp_spec <- decomposition_model(
  STL((cases_count) ~ season(period = 52) +
        season(period = 13),
      robust = TRUE),
  ETS(season_adjust ~ season("N"))
)


fit_w <- hch.tsb_w_agg_train %>%
  model(ets = ETS(cases_count),
        #arima = ARIMA((cases_count)),
        #lm = TSLM((cases_count) ~ season()),
        #`Seasonal naïve` = SNAIVE(cases_count),
         ms = my_dcmp_spec) %>%
  reconcile(bu_ets = bottom_up(ets),
            td_ets = top_down(ets),
            min_trace_ets = min_trace(ets, "mint_shrink"),
            #bu_arima = bottom_up(arima),
```

```
            #td_arima = top_down(arima),
            #min_trace_arima = min_trace(arima, "mint_shrink"),
            #bu_tslm = bottom_up(lm),
            #td_tslm = top_down(lm),
            #min_trace_tslm = min_trace(lm, "mint_shrink"),
            bu_ms = bottom_up(ms),
            td_ms = top_down(ms),
            min_trace_ms = min_trace(ms, "mint_shrink")
            )
```

```
fit_w  %>%
  filter(is_aggregated(region)|county=='BUDAPEST') %>%
  select(region,county,ets,ms) %>%
  pivot_longer(-c(region,county), names_to = "Model name",
                        values_to = "cases_count") %>%
  kable()
```

12 Base Models selected.

region	county	Model name	cases_count
Central Hungary	BUDAPEST	ets	<ETS(A,Ad,N)>
Central Hungary	BUDAPEST	ms	
		ets	<ETS(M,N,N)>
		ms	

13. Analyze the IC metrics of ETS, ARIMA, and related reconcile() functions

- Country-level IC metrics:

STL Decomposition +ETS

```
fit_w %>%
  filter(is_aggregated(region))%>%
  select(ms) %>%report()
```

```
## Series: cases_count
## Model: STL decomposition model
## Transformation: (cases_count)
## Combination: season_adjust + season_52 + season_13
##
## ==================================================
##
## Series: season_adjust + season_52
## Model: COMBINATION
## Combination: season_adjust + season_52
##
## ======================================
##
## Series: season_adjust
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.242132
##
```

```
##   Initial states:
##      l[0]
##  1155.638
##
##   sigma^2:  40802.31
##
##      AIC     AICc      BIC
## 6946.870 6946.928 6958.969
##
## Series: season_52
## Model: SNAIVE
##
## sigma^2: 19.3084
##
##
## Series: season_13
## Model: SNAIVE
##
## sigma^2: 62.0519
```

ETS

```
fit_w %>%
  filter(is_aggregated(region))%>%
  select(ets) %>%report()
```

```
## Series: cases_count
## Model: ETS(M,N,N)
##   Smoothing parameters:
##     alpha = 0.8197294
##
##   Initial states:
##      l[0]
##  1523.214
##
##   sigma^2:  0.1634
##
##      AIC     AICc      BIC
## 7025.464 7025.522 7037.564
```

- Budapest IC metrics:

STL Decomposition +ETS

```
fit_w %>%
  filter(county=='BUDAPEST') %>%
  select(ms) %>%report()
```

```
## Series: cases_count
## Model: STL decomposition model
## Transformation: (cases_count)
## Combination: season_adjust + season_52 + season_13
##
## ==================================================
##
## Series: season_adjust + season_52
## Model: COMBINATION
```

```
## Combination: season_adjust + season_52
##
## =====================================
##
## Series: season_adjust
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.2347407
##
##   Initial states:
##      l[0]
##  87.32583
##
##   sigma^2:  1853.35
##
##      AIC     AICc      BIC
## 5657.613 5657.671 5669.712
##
## Series: season_52
## Model: SNAIVE
##
## sigma^2: 0.586
##
##
## Series: season_13
## Model: SNAIVE
##
## sigma^2: 7.7895
```

ETS

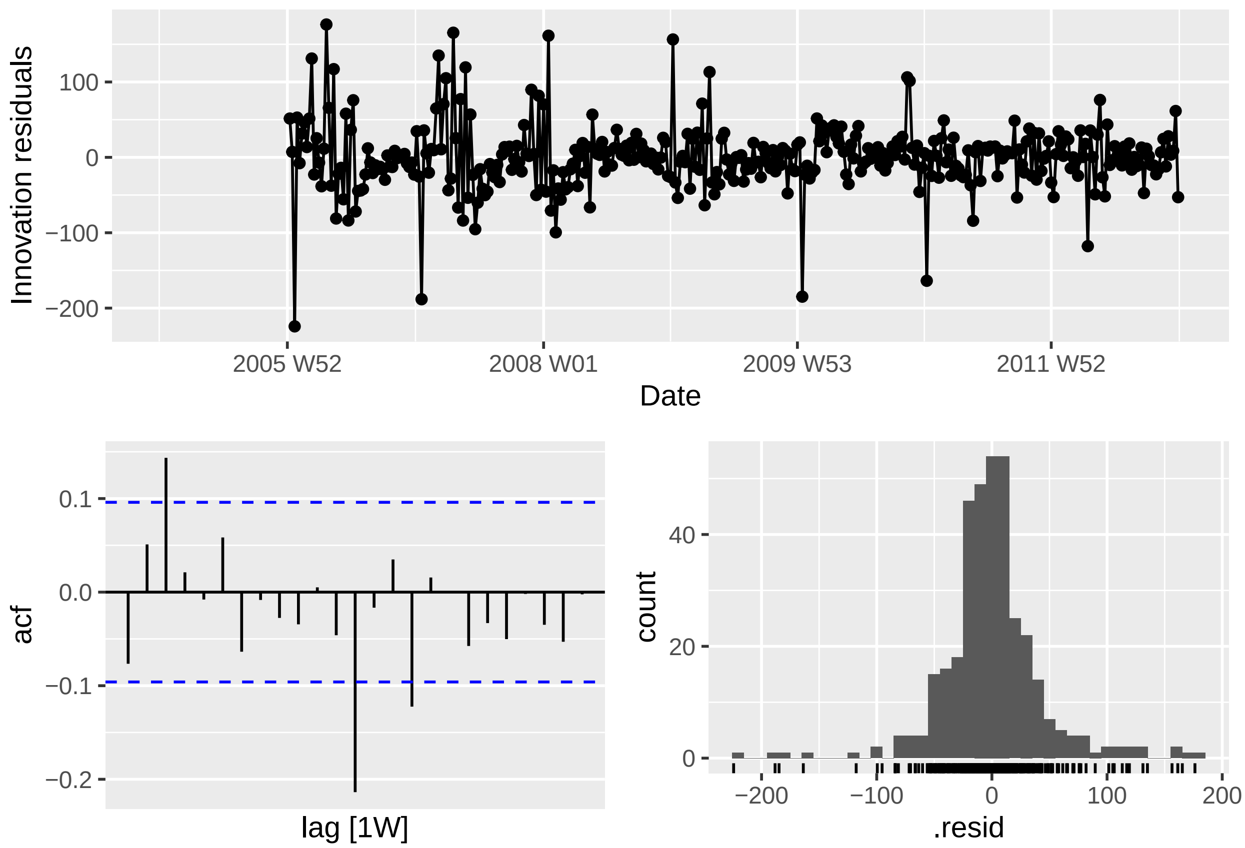
```
fit_w %>%
 filter(county=='BUDAPEST') %>%
  select(ets) %>%report()
```

```
## Series: cases_count
## Model: ETS(A,Ad,N)
##   Smoothing parameters:
##     alpha = 0.2241247
##     beta  = 0.1320557
##     phi   = 0.8000005
##
##   Initial states:
##      l[0]      b[0]
##  159.4953 -2.971327
##
##   sigma^2:  2667.694
##
##      AIC     AICc      BIC
## 5812.467 5812.672 5836.666
```

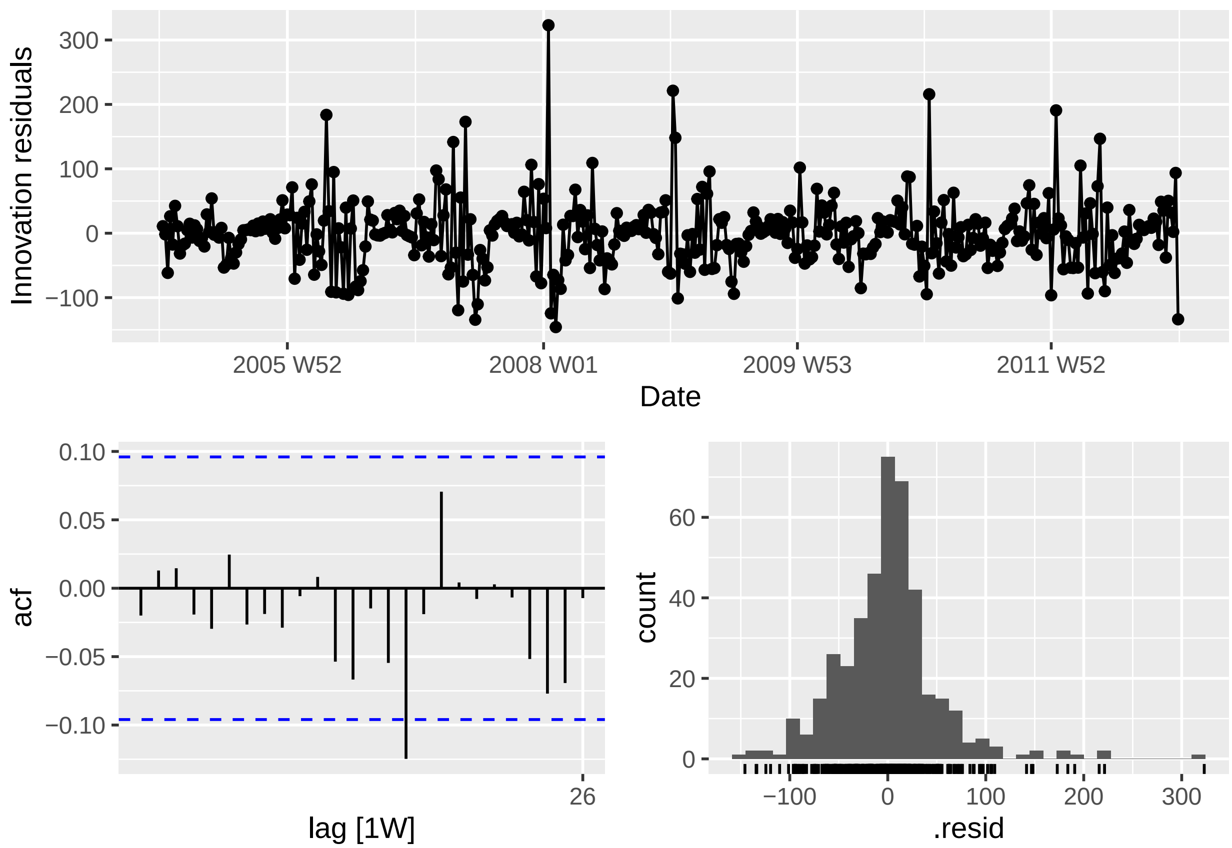
14. Get the residuals plot of STL+ETS and base ETS

- Budapest (county level) decomposition:

```
fit_w %>%
  filter(county=='BUDAPEST') %>%
  select(ms) %>%gg_tsresiduals()
```

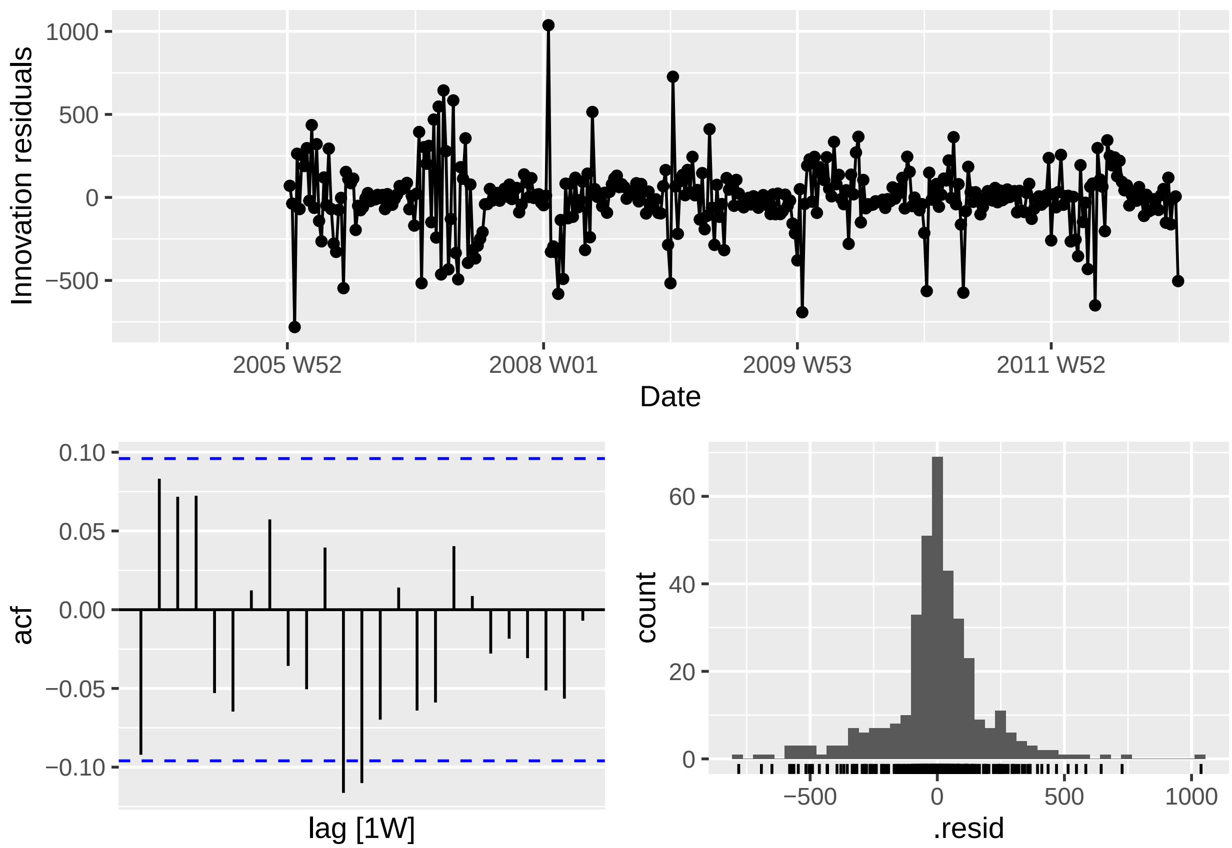



```
fit_w %>%
  filter(county=='BUDAPEST') %>%
  select(ets) %>%gg_tsresiduals()
```

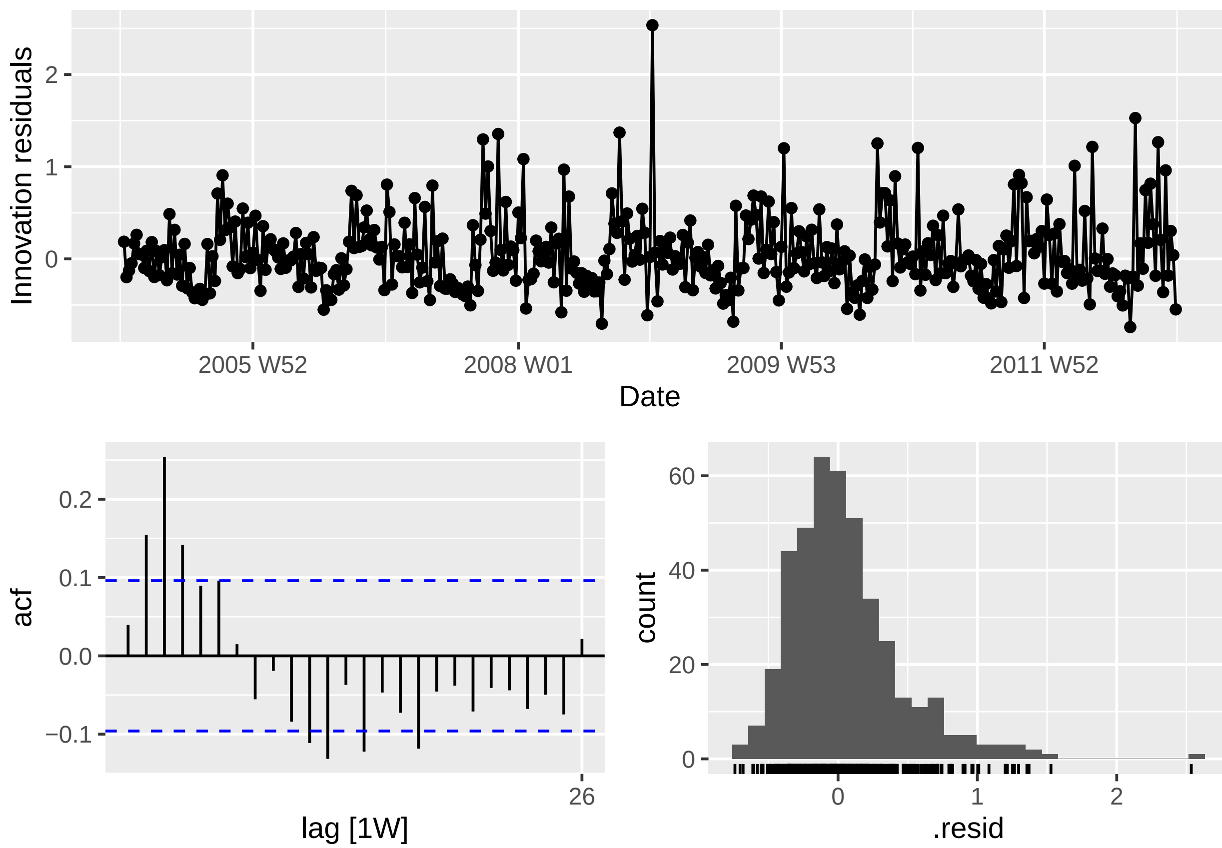


- Country-level decomposition:

```
fit_w %>%
  filter(is_aggregated(region)) %>%
   select(ms) %>%gg_tsresiduals()
```




```
fit_w %>%
  filter(is_aggregated(region)) %>%
   select(ets) %>%gg_tsresiduals()
```



15. Run the forecast of next 3 years on the model:

```
fc_w <- fit_w %>%
  fabletools::forecast(h=3*52)
```

15.1 Get the accuracy metrics for comparison:

- Country-level accuracy metrics:

```
fc_w %>%
  fabletools::accuracy(
    data = hch.tsb_w_agg,
    measures = list(rmse = RMSE, mase = MASE, mape = MAPE, mae=MAE)
  ) %>%
  filter(is_aggregated(region)) %>%
  arrange(rmse) %>%
  transmute(.model,
            region = if_else(is_aggregated(region),
                             'country-level',
                             as.character(region)),
            county, rmse, mase, mape, mae) %>%
  kable()
```

.model	region	county	rmse	mase	mape	mae
bu_ms	country-level		229.5733	0.7746279	63.08866	164.9512

.model	region	county	rmse	mase	mape	mae
min_trace_ms	country-level		243.3070	0.8400467	78.33881	178.8816
ms	country-level		283.4360	1.0393209	114.00108	221.3156
td_ms	country-level		283.4360	1.0393209	114.00108	221.3156
min_trace_ets	country-level		433.3375	1.7484112	171.29878	372.3110
bu_ets	country-level		434.2118	1.7477170	169.68212	372.1632
ets	country-level		492.5353	1.8632157	123.55050	396.7577
td_ets	country-level		492.5353	1.8632157	123.55050	396.7577

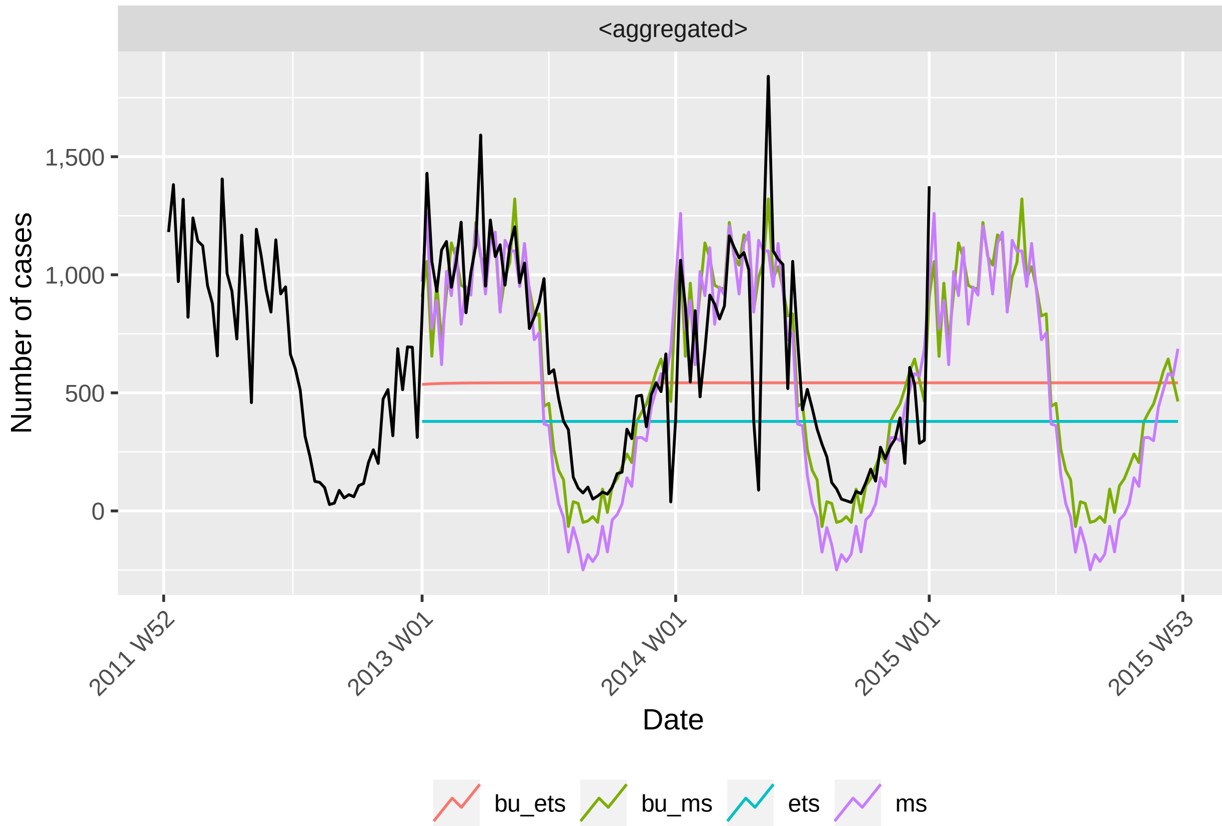
- Budapest (county-level) accuracy metrics:

```
fc_w %>%
  fabletools::accuracy(
    data = hch.tsb_w_agg,
    measures = list(rmse = RMSE, mase = MASE, mape = MAPE, mae=MAE)
  ) %>%
  filter(county=='BUDAPEST') %>%
  arrange(rmse) %>%
  transmute(.model,
            region = if_else(is_aggregated(region),
                             'country-level',
                             as.character(region)),
            county, rmse, mase, mape, mae) %>%
  kable()
```

.model	region	county	rmse	mase	mape	mae
bu_ms	Central Hungary	BUDAPEST	52.76401	0.8555008	86.75492	38.69676
ms	Central Hungary	BUDAPEST	52.76401	0.8555008	86.75492	38.69676
min_trace_ms	Central Hungary	BUDAPEST	52.88146	0.8703134	88.38020	39.36678
td_ms	Central Hungary	BUDAPEST	65.11551	1.1206434	146.75325	50.68993
min_trace_ets	Central Hungary	BUDAPEST	72.25726	1.2810529	211.35485	57.94571
bu_ets	Central Hungary	BUDAPEST	72.81865	1.3071944	231.95164	59.12816
ets	Central Hungary	BUDAPEST	72.81865	1.3071944	231.95164	59.12816
td_ets	Central Hungary	BUDAPEST	81.27664	1.3330600	128.43018	60.29814

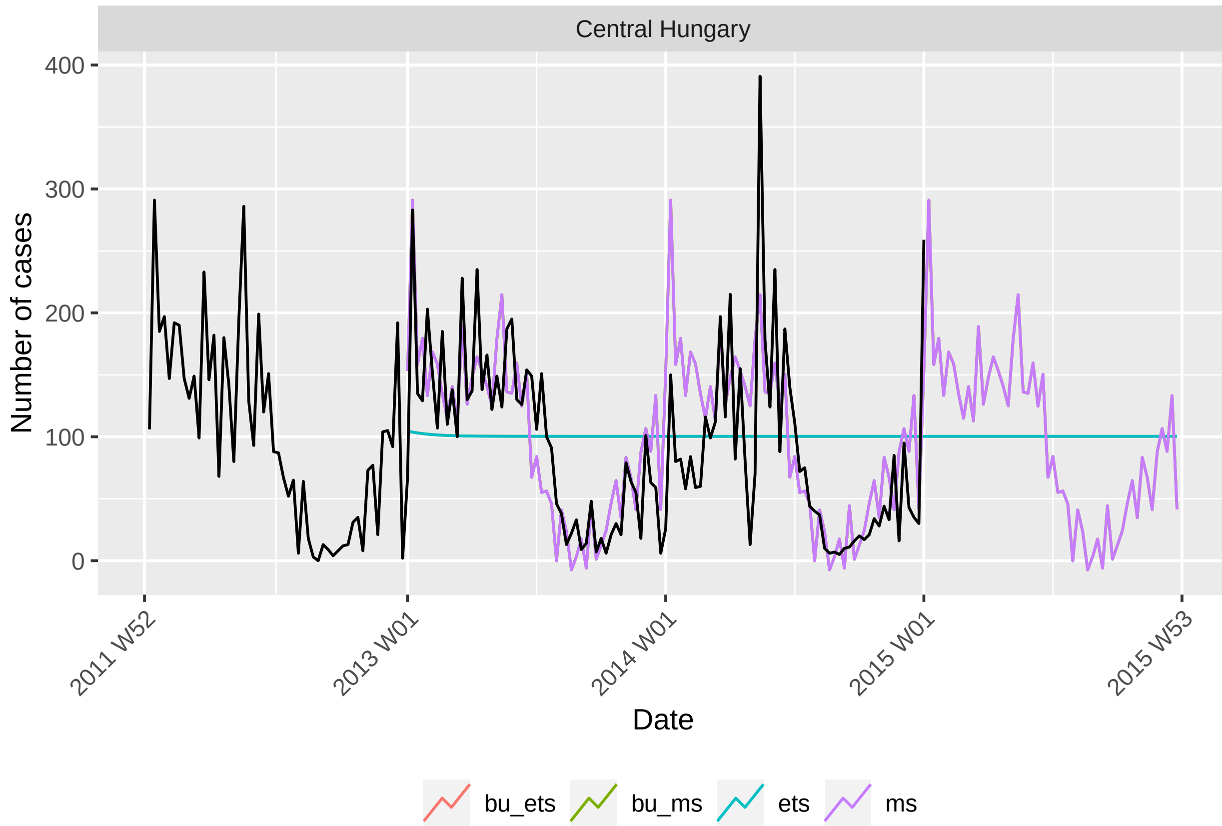
16. Plot the forecast at the country-level:

```
autoplot(
  fc_w %>% filter(.model%in%c('ets','ms','bu_ms','bu_ets')) %>%
    filter( is_aggregated(region)),
  hch.tsb_w_agg %>%
    ungroup() %>%
    filter(year(Date) >2011), level = NULL) +
  facet_wrap(~region, scales = "free_y") +
  scale_y_continuous(labels = scales::comma_format()) +
  labs(color = "", x = "Date" ,y = "Number of cases") +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  )
```



17.Plot the forecast at the country-level and all the regions

```
autoplot(
  fc_w %>% filter(.model%in%c('ets','ms','bu_ms','bu_ets')) %>%
    filter(county=='BUDAPEST' ),
  hch.tsb_w_agg %>%
    ungroup() %>%
    filter(year(Date) >2011), level = NULL) +
  facet_wrap(~region, scales = "free_y") +
  scale_y_continuous(labels = scales::comma_format()) +
  labs(color = "", x = "Date" ,y = "Number of cases") +
  theme(
    legend.position = 'bottom',
    axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1)
  )
```



18. Conduct Ljung-Box test on the fit:

- Ljung-Box test: we see a p-value much smaller than 0.05, thus we can reject the null hypothesis, indicating the time series does contain an autocorrelation.

```
augment(fit_w)  %>%
  filter(is_aggregated(region) | county=='BUDAPEST') %>%
  filter(.model=='ms'|.model=='ets')  %>%
  features(.resid, ljung_box,lag=13) %>%
  kable()
```

region	county	.model	lb_stat	lb_pvalue
Central Hungary	BUDAPEST	ets	5.131752	0.9721771
Central Hungary	BUDAPEST	ms	32.677784	0.0019043
		ets	58.584242	0.0000001
		ms	25.201117	0.0217181