POP77014: Assignment 2

Imelda Finn (22334657)

08/04/2023

Overview

This report analyses the evolution of all cause mortality in the USA 2015-2023.

1. The data is from the OECD website: COVID-19 Health Indicators, Mortality (by week)(OECD 2023)

Mortality rates have been generally decreasing over time, i.e. life expectancy has been rising. (This may not continue in the future, particularly in developed nations, as lifestyle factors such as obesity may tend to reduce inter-generational life expectancy.)

In a stable population (where births/migration replace deaths), mortality rates will be gradually reducing, as life expectancy causes fewer deaths at each age. In an expanding population, average age is decreasing so mortality rates reduce faster. In a contracting population, the average age is increasing as deaths are not balanced by births or inward (younger) migrants, so mortality rates will be steady or even rising (temporarily).

COVID-19 started in 2019, the first deaths in America were in 2020; this will distort the mortality predictions for 2021 and beyond. Some models may be better at capturing what happened, and so might be more useful in future pandemics.

Final Model: ensemble - mean of 7 models

- HoltWinters with high α (=0.94), quickly takes surge in mortality in 2020 into account

"if you want to make prediction intervals for forecasts made using exponential smoothing methods, the prediction intervals require that the forecast errors are uncorrelated and are normally distributed with mean zero and constant variance.' ' (Coghlan 2023) Assumption doesn't hold with this data

3. If relevant, all estimated equations associated with constructing forecasts from this method

4. Report the MAPE and MAE for the training period and the validation period. You may also report other metrics if relevant.

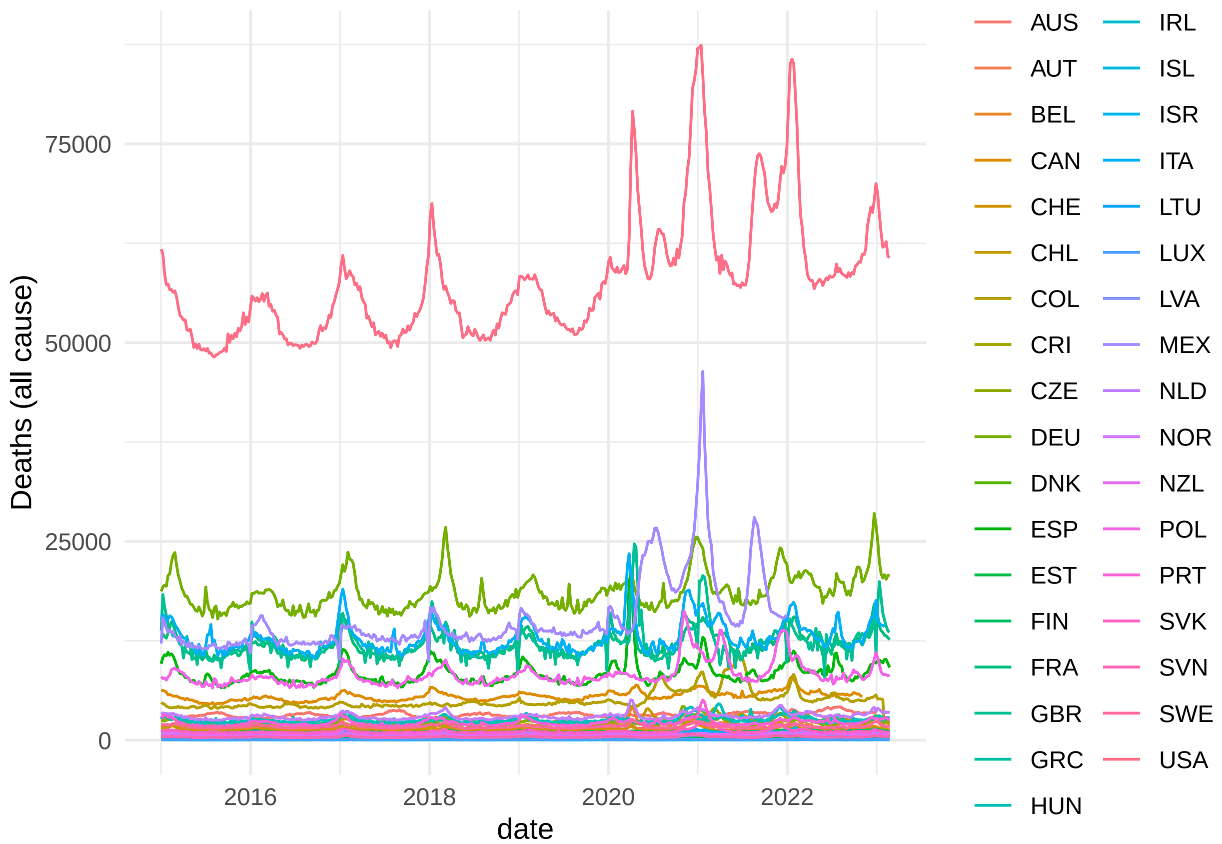
6. A single figure showing the fit of the final version of the model to the entire period available in the data (i.e., in-sample fit. For options 1 and 2, you do not have access to the "future" data).

```
# read in the data
# show_col_types = FALSE

#https://stats.oecd.org/Index.aspx?DataSetCode=HEALTH_MORTALITY
mort <- read_csv("data/HEALTH_MORTALITY_all.csv")
```

```
## Rows: 268379 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): COUNTRY, Country, GENDER, Gender, AGE, Age, VARIABLE, Variable, Fl...
## dbl  (5): WEEK, Week number, YEAR, Year, Value
##
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
mort$date <- ISOweek2date(paste0(mort$YEAR, "-W", sprintf("%02d", mort$WEEK),"-4"))
mort <- mort %>%
  filter(Age == "Total" & Gender == "Total"  & VARIABLE == "ALLCAUNB") %>%
  arrange(desc(COUNTRY), date)

#summary(mort)
#unique(mort$COUNTRY)

mort %>% ggplot(aes(x=date, y=Value, colour = COUNTRY)) + geom_line() +
  ylab("Deaths (all cause)")
```




```r
country_code <- "USA"
country <- mort %>% filter(COUNTRY == country_code) %>% select(date, Value, YEAR)
country %>% arrange(date)
```

```
## # A tibble: 425 x 3
##    date        Value  YEAR
##    <date>      <dbl> <dbl>
##  1 2015-01-01 61763  2015
##  2 2015-01-08 61163  2015
##  3 2015-01-15 58652  2015
##  4 2015-01-22 57297  2015
##  5 2015-01-29 57367  2015
##  6 2015-02-05 56668  2015
##  7 2015-02-12 56678  2015
```

```
##  8 2015-02-19 56334  2015
##  9 2015-02-26 56509  2015
## 10 2015-03-05 55741  2015
## # i 415 more rows
```

```
length(country$date)
```

```
## [1] 425
```

```
min(country$date)
```

```
## [1] "2015-01-01"
```

```
# first week in data  is 1, 2015
# 2020 has 53 weeks

# set the parameters for the time series
startDate <- min(country$date)
startYear <- year(startDate)
startMonth <- month(startDate)
startWeek <- week(startDate)
startDay <- day(startDate)

endDate <- max(country$date)
endYear <- year(endDate)
endMonth <- month(endDate)
endWeek <- week(endDate)

# specify the forecasting parameters
# solve for recommended
# look for smaller alphas to smooth out effect of pandemic
ALPHA <-  0.95
FREQ<- 52
WEEKS <- length(country$Value)

future <- as.integer(WEEKS*0.2)   # 85 for full data

fivenum(country$Value)
```

```
## [1] 48194 51838 56680 60260 87415
```

```
# default graph labels
mtitle <- paste0(country_code," all causes deaths (weekly)")
stitle <- paste0(startMonth, "/", startYear, " - ", endMonth, "/",endYear)
y_lab <- "Deaths"
x_lab <- "Year"
us_y_lim <- c(48000, 90000)     # check when incorporate leap year
y_lim <- us_y_lim
x_lim <- c(startDate, endDate)
```

Data is weekly, 425 weeks from 2015-01-01 to 2023-02-23 inclusive.

```
#Convert the death numbers to a time series
country.ts <- ts(country$Value, start=c(startYear,startWeek), frequency=FREQ)

print(tsfeatures(country.ts))
```

```
## # A tibble: 1 x 20
```

```
##   frequency nperiods seasonal_period trend      spike linearity curvature e_acf1
##       <dbl>    <dbl>           <dbl> <dbl>      <dbl>     <dbl>     <dbl>  <dbl>
## 1        52        1              52 0.740 0.00000121      12.1     0.226  0.940
## # i 12 more variables: e_acf10 <dbl>, seasonal_strength <dbl>, peak <dbl>,
## #   trough <dbl>, entropy <dbl>, x_acf1 <dbl>, x_acf10 <dbl>, diff1_acf1 <dbl>,
## #   diff1_acf10 <dbl>, diff2_acf1 <dbl>, diff2_acf10 <dbl>, seas_acf1 <dbl>
```

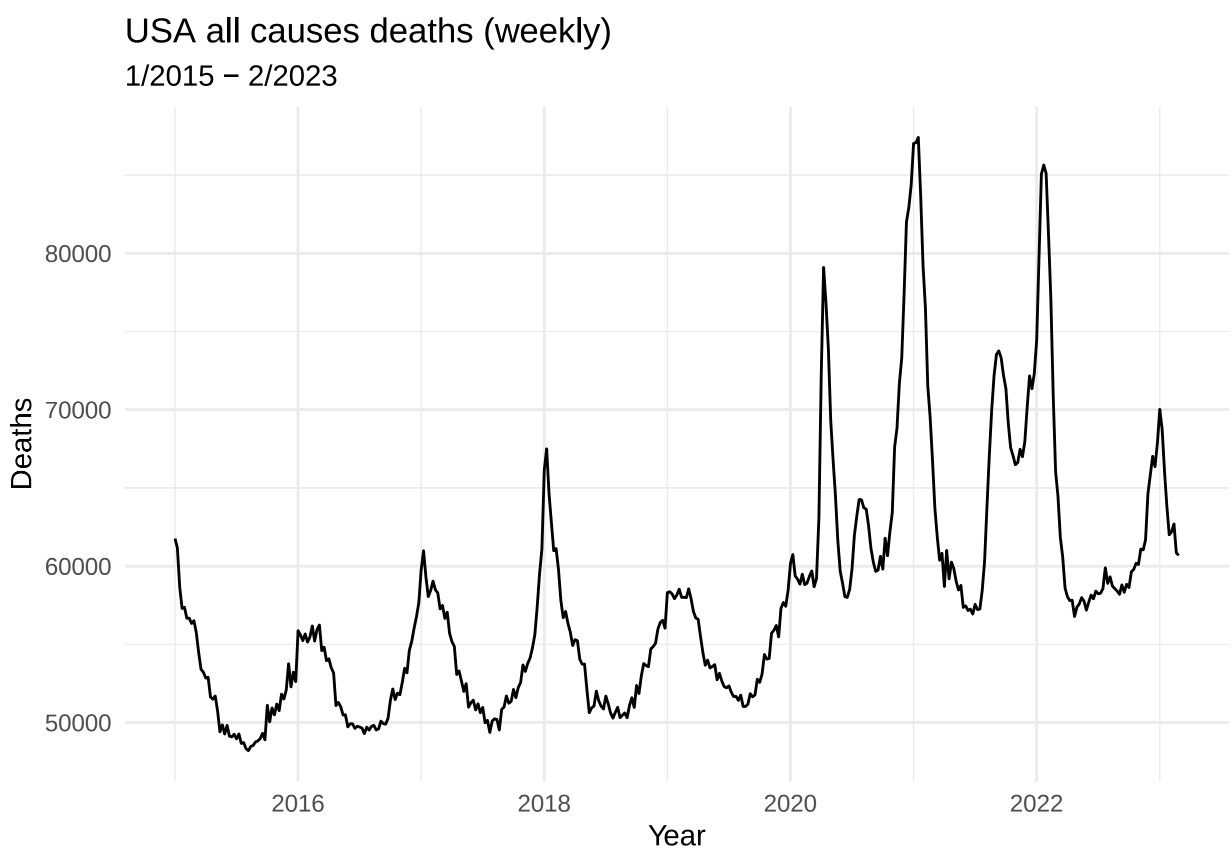
```
summary(country.ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   48194   51838   56680   57863   60260   87415
```

```
head(tail(country,85),1)  #2021-07-15  # first week of test set
```

```
## # A tibble: 1 x 3
##   date       Value  YEAR
##   <date>     <dbl> <dbl>
## 1 2021-07-15 57253  2021
```

```
# plot the whole time series
autoplot(country.ts) +
  ggtitle(mtitle, subtitle = stitle) +
  xlab(x_lab) +
  ylab(y_lab)
```

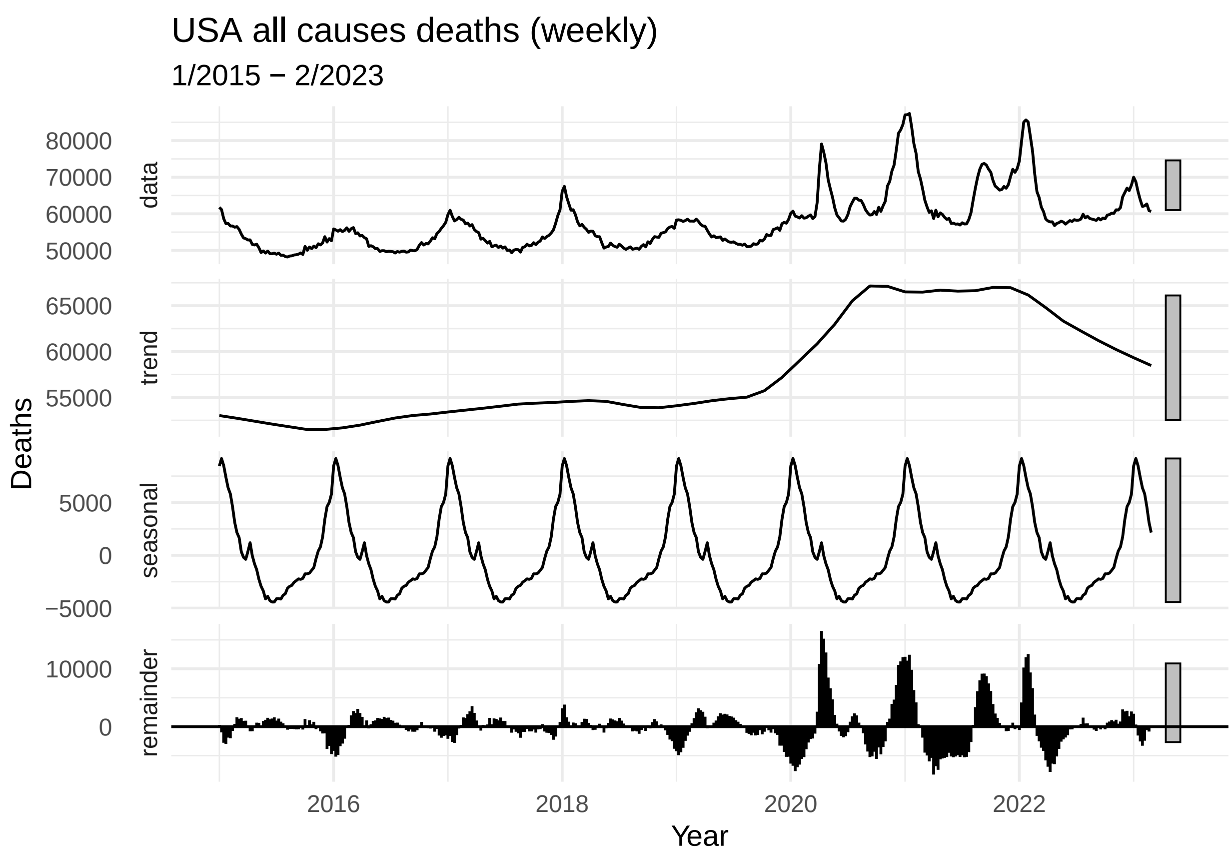



```
nValid <- future
nTrain <- length(country.ts) - nValid
train.ts <- window(country.ts, start = c(startYear, startWeek),
```

```
                   end = c(startYear, nTrain))
valid.ts <- window(country.ts, start = c(startYear, nTrain+1),
                   end = c(startYear, nTrain+nValid))
```

Consider the timeseries decomposition: seasonal component; the trend; and the remainder.

```
#plot the decomposition
(country.stl <- country.ts %>%
  stl(s.window="periodic"))%>%
  autoplot() +
  ggtitle(mtitle, subtitle = stitle) +
  xlab(x_lab) +
  ylab(y_lab)
```



value of time series at time t $= y_t = T_t + S_t + R_t$

```
# This toggles plots from plotly (interactive) to ggplot (static)
interactive <- FALSE

#country %>% plot_time_series(date, Value, .interactive = interactive)

FREQ<- 52
WEEKS <- length(country$Value)

future <- as.integer(WEEKS*0.2)
dlimit <- head(tail(country,future),1)$date

nTrain <- length(country$date) - future
```

```r
train <- country %>% select(Value, date) %>% filter(date < dlimit)
valid <- country %>% select(Value, date) %>% filter(date >= dlimit)

# Split Data 80/20
splits <- initial_time_split(train, prop = 0.8)

# Model 1: auto_arima ----
model_fit_arima_no_boost <- arima_reg() %>%
  set_engine(engine = "auto_arima") %>%
  fit(Value ~ date, data = training(splits))
```

```
## frequency = 13 observations per 1 quarter
```

```r
# Model 1b: auto_arima ----   ARIMA(3,1,0)(0,0,2)[52]
model_fit_arima_52 <- arima_reg(seasonal_period = 52,
        non_seasonal_ar = 3, non_seasonal_differences = 1,
        non_seasonal_ma = 0, seasonal_ar = 0, seasonal_differences = 0,
        seasonal_ma = 2) %>%
  set_engine(engine = "arima") %>%
  fit(Value ~ date, data = training(splits))


# Model 2: arima_boost ----
model_fit_arima_boosted <- arima_boost(
  min_n = 1,
  learn_rate = 0.015
) %>%
  set_engine(engine = "auto_arima_xgboost") %>%
  fit(Value ~ date + as.numeric(date) + factor(month(date, label = TRUE), ordered = F),
      data = training(splits))
```

```
## frequency = 13 observations per 1 quarter
```

```r
# Model 3: ets ----
model_fit_ets <- exp_smoothing() %>%
  set_engine(engine = "ets") %>%
  fit(Value ~ date, data = training(splits))
```

```
## frequency = 13 observations per 1 quarter
```

```r
# Model 4: prophet ----
model_fit_prophet <- prophet_reg(seasonality_weekly = TRUE) %>%
  set_engine(engine = "prophet") %>%
  fit(Value ~ date, data = training(splits))
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```r
# Model 5: lm ----
model_fit_lm <- linear_reg() %>%
  set_engine("lm") %>%
  fit(Value ~ as.numeric(date) + factor(month(date, label = TRUE), ordered = FALSE),
      data = training(splits))

# Model 6: earth ----
model_spec_mars <- mars(mode = "regression") %>%  set_engine("earth")
```

```r
recipe_spec <- recipe(Value ~ date, data = training(splits)) %>%
  step_date(date, features = "month", ordinal = FALSE) %>%
  step_mutate(date_num = as.numeric(date)) %>%
  step_normalize(date_num) %>%
  step_rm(date)

wflw_fit_mars <- workflow() %>%
  add_recipe(recipe_spec) %>%
  add_model(model_spec_mars) %>%
  fit(training(splits))
```

```
## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

##
## Attaching package: 'plotrix'

## The following object is masked from 'package:scales':
##
##     rescale

## Loading required package: TeachingDemos
```

```r
models_tbl <- modeltime_table(
  model_fit_arima_no_boost,
  model_fit_arima_boosted,
  model_fit_arima_52,
  model_fit_ets,
  model_fit_prophet,
  model_fit_lm,
  wflw_fit_mars
)

models_tbl
```

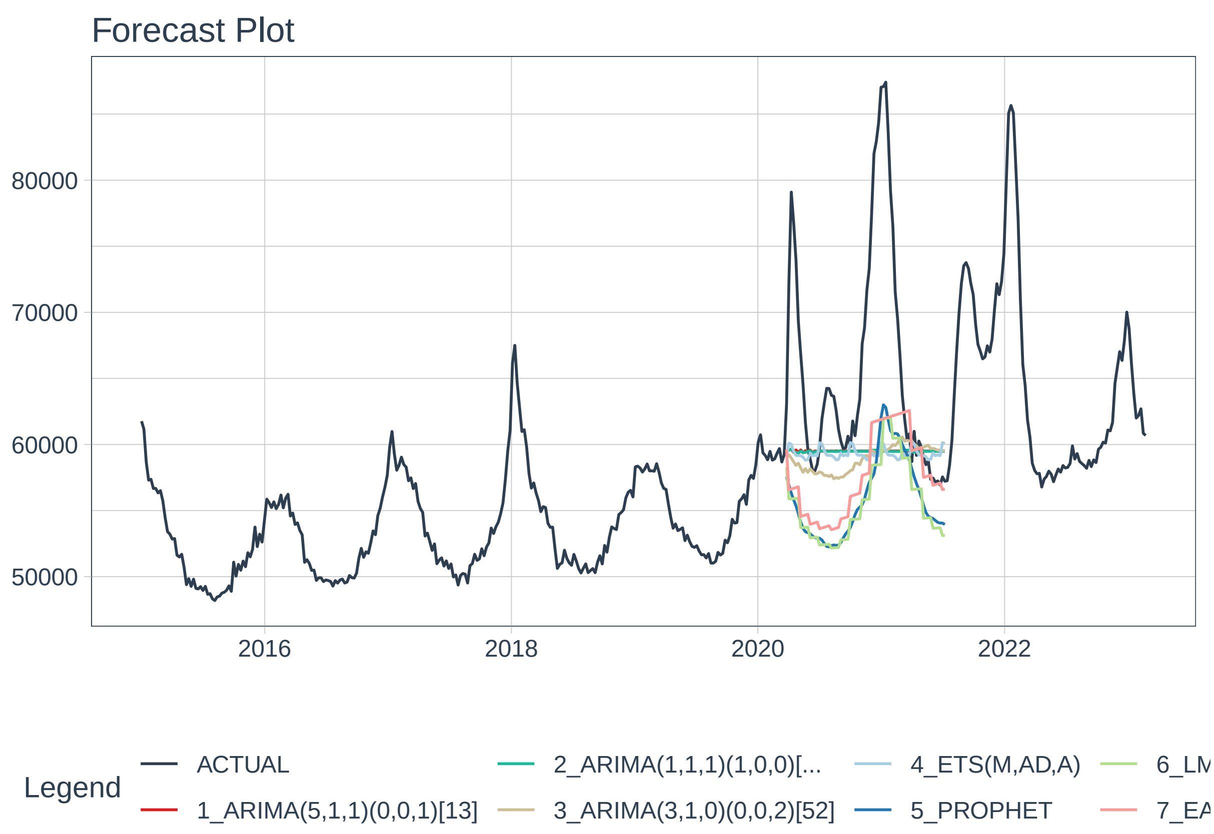
```
## # Modeltime Table
## # A tibble: 7 x 3
##   .model_id .model     .model_desc
##       <int> <list>     <chr>
## 1         1 <fit[+]>   ARIMA(5,1,1)(0,0,1)[13]
## 2         2 <fit[+]>   ARIMA(1,1,1)(1,0,0)[13] W/ XGBOOST ERRORS
## 3         3 <fit[+]>   ARIMA(3,1,0)(0,0,2)[52]
## 4         4 <fit[+]>   ETS(M,AD,A)
## 5         5 <fit[+]>   PROPHET
## 6         6 <fit[+]>   LM
## 7         7 <workflow> EARTH
```

```r
# calibrate
calibration_tbl <- models_tbl %>%
  modeltime_calibrate(new_data = train)

calibration_tbl
```

```
## # Modeltime Table
## # A tibble: 7 x 5
```

```
##   .model_id .model     .model_desc                        .type .calibration_data
##       <int> <list>     <chr>                              <chr> <list>
## 1         1 <fit[+]>   ARIMA(5,1,1)(0,0,1)[13]            Test  <tibble>
## 2         2 <fit[+]>   ARIMA(1,1,1)(1,0,0)[13] W/ XGBOO~ Test  <tibble>
## 3         3 <fit[+]>   ARIMA(3,1,0)(0,0,2)[52]            Test  <tibble>
## 4         4 <fit[+]>   ETS(M,AD,A)                        Test  <tibble>
## 5         5 <fit[+]>   PROPHET                            Test  <tibble>
## 6         6 <fit[+]>   LM                                 Test  <tibble>
## 7         7 <workflow> EARTH                              Test  <tibble>
```

```r
# test set forecast and accuracy
calibration_tbl %>%
  modeltime_forecast(
    new_data    = testing(splits),
    actual_data = country,
  ) %>%
  plot_modeltime_forecast(
    .legend_max_width = 25, # For mobile screens
    .interactive      = interactive,
    .conf_interval_show = FALSE
  )
```




```r
calibration_tbl %>%
  modeltime_accuracy() %>%
  table_modeltime_accuracy(
    .interactive = interactive
  )
```

Accuracy Table

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rs
1	ARIMA(5,1,1)(0,0,1)[13]	Test	6194.18	11.11	7.08	10.68	7718.55	0.0
2	ARIMA(1,1,1)(1,0,0)[13] W/ XGBOOST ERRORS	Test	6185.72	11.10	7.07	10.67	7714.75	0.0
3	ARIMA(3,1,0)(0,0,2)[52]	Test	6695.46	12.10	7.65	11.50	7976.54	0.1
4	ETS(M,AD,A)	Test	6055.22	10.84	6.92	10.45	7617.15	0.0
5	PROPHET	Test	2509.52	3.77	2.87	4.07	5475.89	0.5
6	LM	Test	2695.90	4.08	3.08	4.40	5641.87	0.4
7	EARTH	Test	2358.15	3.57	2.69	3.81	5107.08	0.5

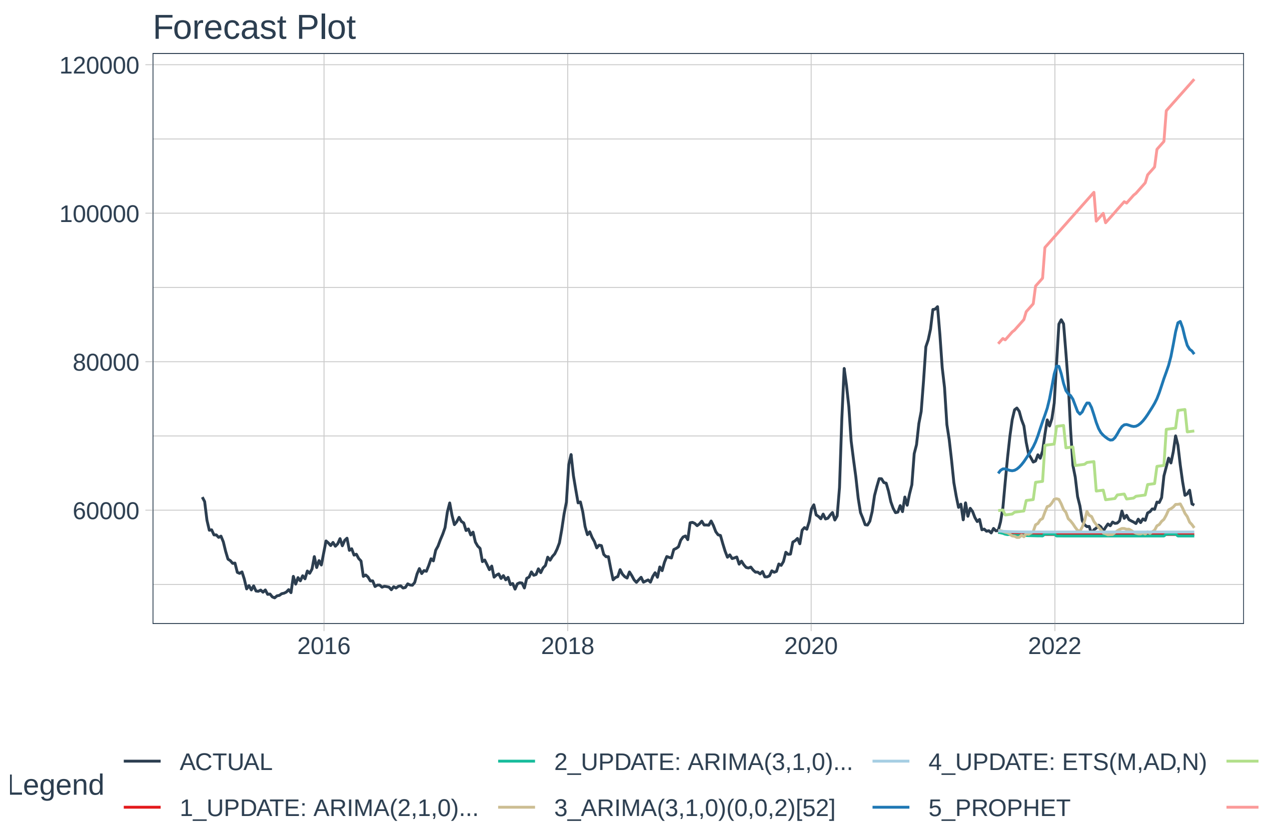
```r
refit_tbl <- calibration_tbl %>%
  modeltime_refit(data = train)
```

```
## frequency = 13 observations per 1 quarter
```

```
## frequency = 13 observations per 1 quarter
## frequency = 13 observations per 1 quarter
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```r
refit_tbl %>%
  modeltime_forecast(h = "85 weeks", actual_data = country) %>%
  plot_modeltime_forecast(
    .legend_max_width = 25, # For mobile screens
    .interactive      = interactive,
    .conf_interval_show = FALSE
  )
```




```r
ensemble_fit <- refit_tbl %>%
  ensemble_average(type = "mean")

ensemble_fit
```

```
## -- Modeltime Ensemble -------------------------------------------
## Ensemble of 7 Models (MEAN)
##
## # Modeltime Table
## # A tibble: 7 x 5
##   .model_id .model     .model_desc                         .type .calibration_data
##       <int> <list>     <chr>                               <chr> <list>
## 1         1 <fit[+]>   UPDATE: ARIMA(2,1,0)(0,0,1)[13]     Test  <tibble>
## 2         2 <fit[+]>   UPDATE: ARIMA(3,1,0)(1,0,0)[13] ~ Test  <tibble>
## 3         3 <fit[+]>   ARIMA(3,1,0)(0,0,2)[52]             Test  <tibble>
## 4         4 <fit[+]>   UPDATE: ETS(M,AD,N)                 Test  <tibble>
## 5         5 <fit[+]>   PROPHET                             Test  <tibble>
## 6         6 <fit[+]>   LM                                  Test  <tibble>
## 7         7 <workflow> EARTH                               Test  <tibble>
```

```r
# Calibration
e.calibration_tbl <- modeltime_table(
  ensemble_fit
) %>%
  modeltime_calibrate(train, quiet = FALSE)

# get split for in-sample forecast
```

```r
splits <- initial_time_split(country, prop = 0.8)
# Forecast vs Test Set
#par(mfrow = c(2,1))

png("docs/ensemble_models.png")
calibration_tbl %>%
  modeltime_forecast(
    new_data    = testing(splits),
    actual_data = train,
  ) %>%
  plot_modeltime_forecast(
    .legend_max_width = 25, # For mobile screens
    .interactive      = interactive,
    .conf_interval_show = FALSE
  )
dev.off()
```

```
## pdf
##   2
```

```r
png("docs/ensemble.png")
e.calibration_tbl %>%
  modeltime_forecast(
    new_data    = testing(splits),
    actual_data = country
  ) %>%
  plot_modeltime_forecast(.interactive = FALSE,
                          .title = paste0(mtitle,"\n ",stitle),
                          .x_lab = x_lab,
                          .y_lab = y_lab)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
#par(mfrow = c(1,1))

e.calibration_tbl %>%
  modeltime_accuracy() %>%
  table_modeltime_accuracy(
    .interactive = interactive
  )
```

Accuracy Table

.model_id	.model_desc	.type	mae	mape	mase	smape	rmse	rsq
1	ENSEMBLE (MEAN): 7 MODELS	Test	3669.21	6.26	4.19	6.28	5178.64	0.62

```r
e.calibration_tbl %>%
  modeltime_accuracy() #%>% table()
```

```
## # A tibble: 1 x 9
##   .model_id .model_desc               .type   mae  mape  mase smape  rmse   rsq
##       <int> <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         1 ENSEMBLE (MEAN): 7 MODELS Test  3669.  6.26  4.19  6.28 5179. 0.623
```

Individual Models

Using HoltWinters with trend and seasonality (parameters derived from data)

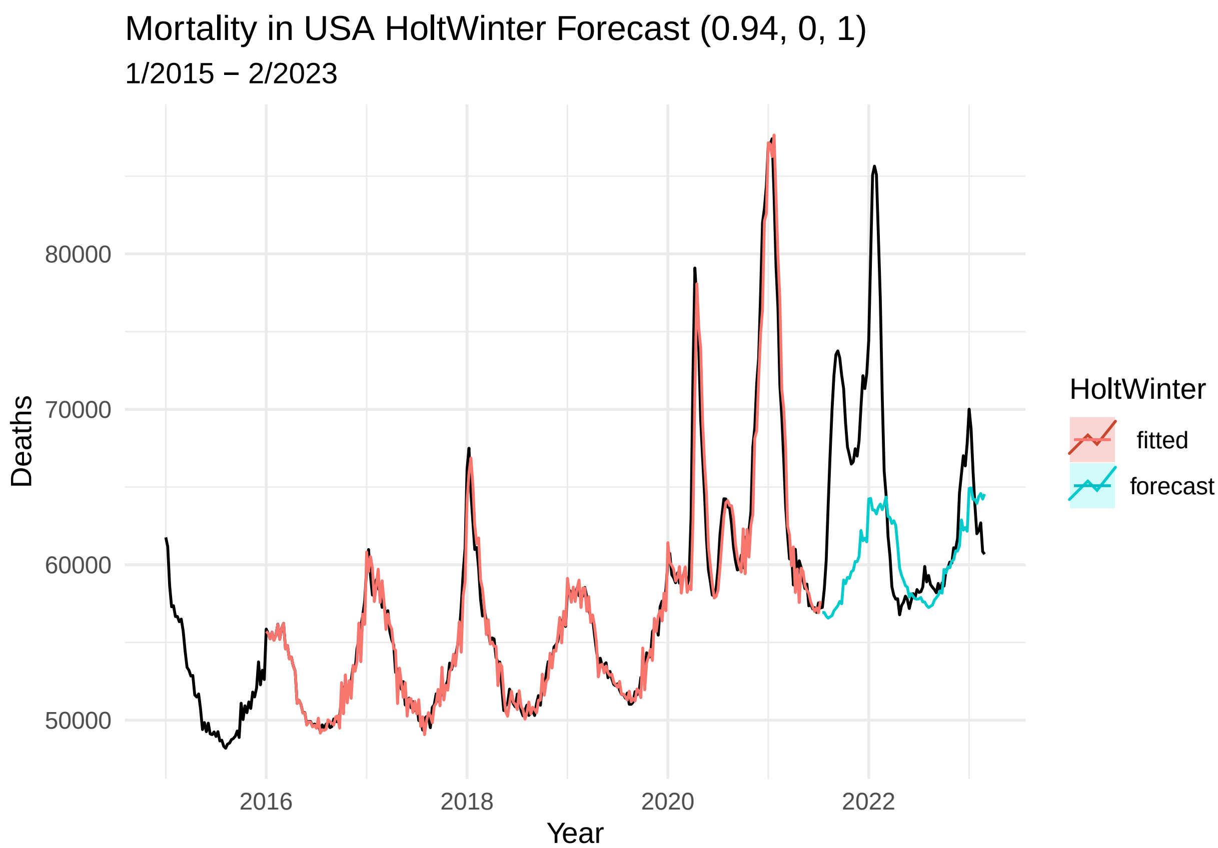
HoltWinters is the best for the test data (ie last 20%)

```
country.hw<- HoltWinters(train.ts)
country.hw
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train.ts)
##
## Smoothing parameters:
##  alpha: 0.9376002
##  beta : 0
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   59738.22486
## b      13.09048
## s1  -2821.36446
## s2  -2834.79382
## s3  -3080.29107
## s4  -3224.46196
## s5  -3154.92509
## s6  -3091.05137
## s7  -2785.08032
## s8  -2644.83795
## s9  -2490.27139
## s10 -2228.03700
## s11 -2388.65894
## s12  -872.73337
## s13 -1121.50495
## s14  -734.70776
## s15  -808.60782
## s16  -395.09193
## s17  -320.47990
## s18   238.23181
## s19   213.15821
## s20   532.28066
## s21  2194.28354
## s22  1522.97739
## s23  1673.04283
## s24  1422.59552
## s25  4169.85871
## s26  4190.41023
## s27  3437.99924
```

```
## s28  3417.28944
## s29  3149.68591
## s30  3571.42453
## s31  3761.24021
## s32  3389.39582
## s33  3698.40033
## s34  4171.86394
## s35  2912.11974
## s36  2832.39934
## s37  2432.61027
## s38  2599.34089
## s39  2275.34285
## s40   960.92896
## s41  -486.16957
## s42  -970.60752
## s43 -1278.61882
## s44 -1656.34527
## s45 -1763.14950
## s46 -2423.46344
## s47 -2204.59565
## s48 -2397.96531
## s49 -2564.29923
## s50 -2607.12016
## s51 -2598.92930
## s52 -2519.22486
```

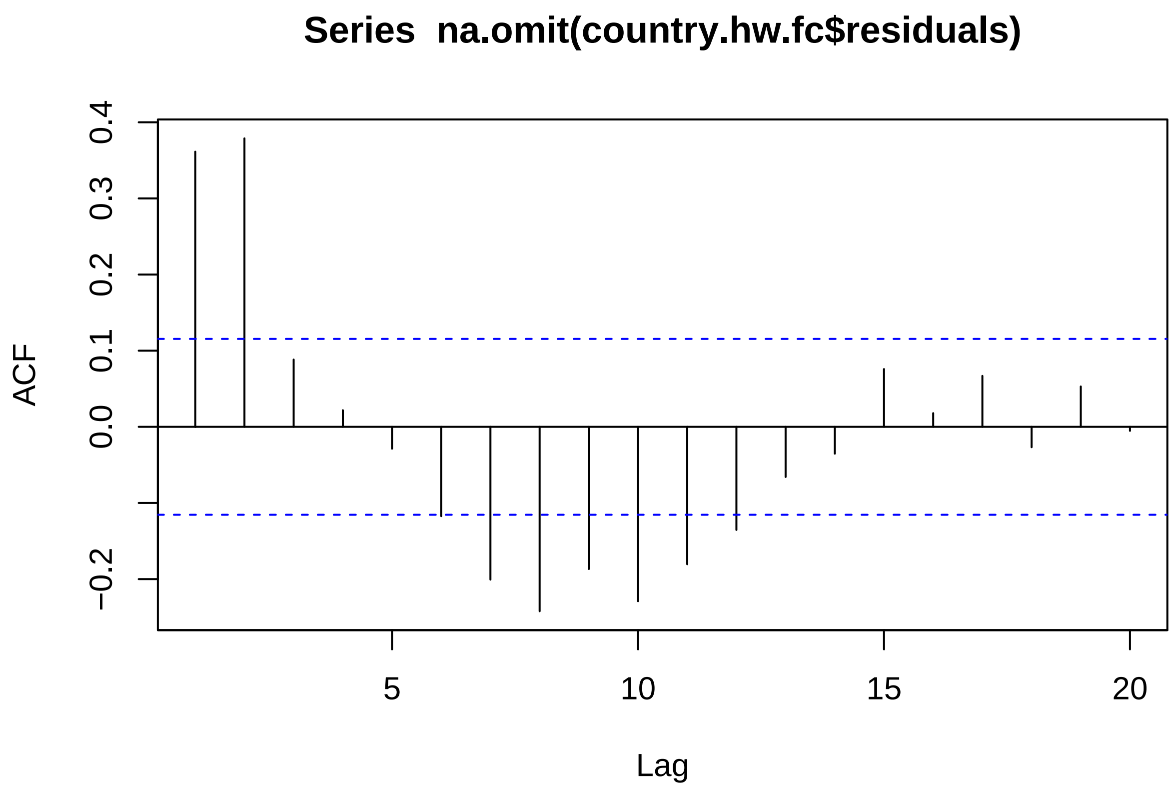
```r
country.hw.fc <- forecast(country.hw, h = future)

#We see from the plot that the Holt-Winters exponential method is very successful in modelling the seas
#- level is off for predictions because not taking enough account of the surge
autoplot(country.ts) +
  autolayer(country.hw.fc, PI=F, series="forecast") +
  autolayer(country.hw$fitted[,1], series = " fitted")+
  ggtitle(paste0("Mortality in ", country_code, " HoltWinter Forecast (0.94, 0, 1)"),
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="HoltWinter"))
```




```
ggsave(here("docs", "holtwinter.png"))
```

```
## Saving 6.5 x 4.5 in image
```

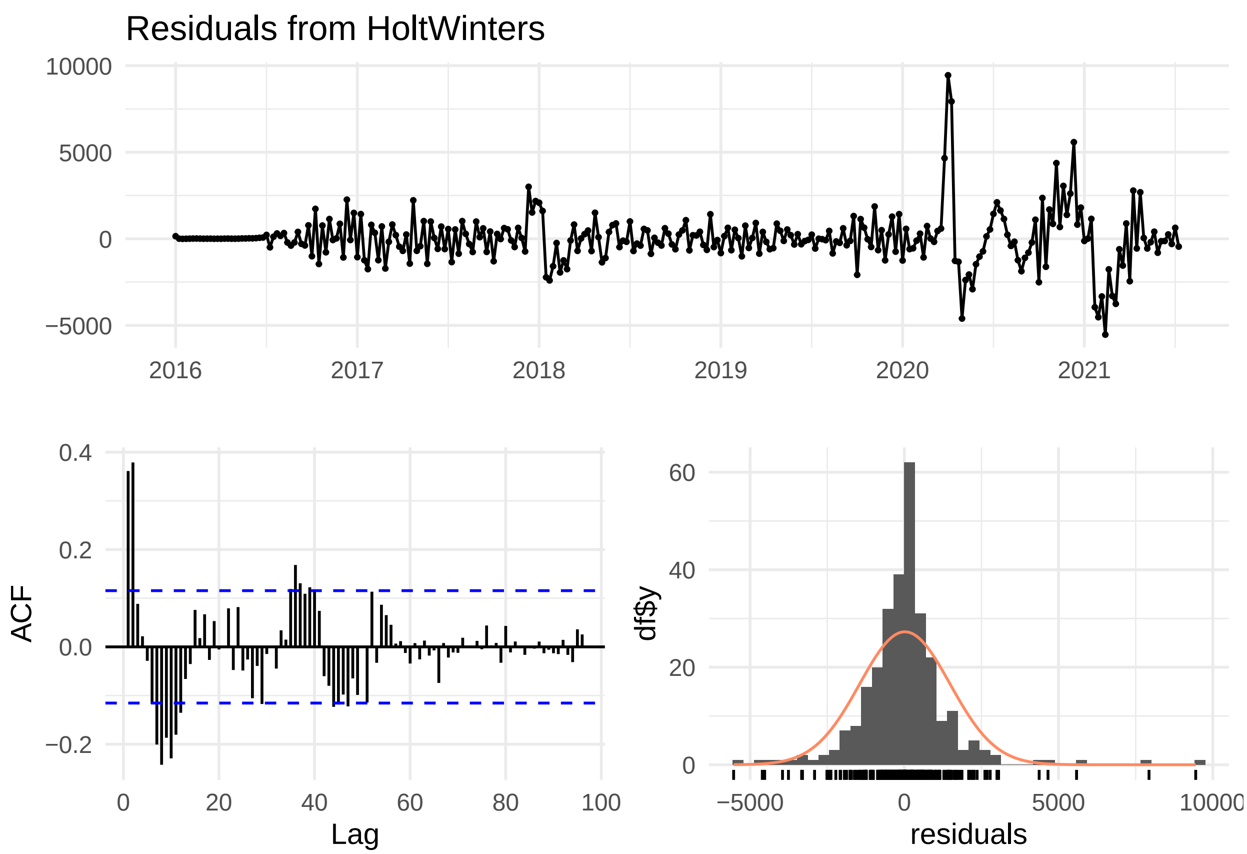
```
Acf(na.omit(country.hw.fc$residuals), lag.max = 20)
```




```
Box.test(country.hw.fc$residuals, lag=20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  country.hw.fc$residuals
## X-squared = 163.76, df = 20, p-value < 2.2e-16
```

```
checkresiduals(country.hw)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 255.43, df = 58, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 58
```

```
print(forecast::accuracy(country.hw.fc, valid.ts))
```

```
##                      ME     RMSE       MAE        MPE     MAPE      MASE
## Training set   16.66519 1465.785  905.4864 0.01273601 1.492413 0.2386424
## Test set     4260.12115 7832.559 5312.6288 5.71928962 7.498161 1.4001520
##                   ACF1 Theil's U
## Training set 0.3613692        NA
## Test set     0.9494772  4.066744
```

Linear regression model ln(y) ~ trend+season

best result on training, worst on test, so overfitted and over predicts on test, so penalised more by MAPE

```
#https://www.rdocumentation.org/packages/forecast/versions/8.21/topics/tslm

tslm(train.ts~ season+trend, lambda=NULL) %>% forecast(h=future) %>% forecast::accuracy()
```

```
##                         ME    RMSE      MAE        MPE    MAPE      MASE
## Training set -3.635105e-13 4294.39 3017.928 -0.4148513 5.03524 0.7953799
##                    ACF1
```

```
## Training set 0.964033
```

```r
# mape 5.03524

tslm(log(train.ts)~ season+trend, lambda=NULL) %>% forecast(h=future) %>% forecast::accuracy()
```

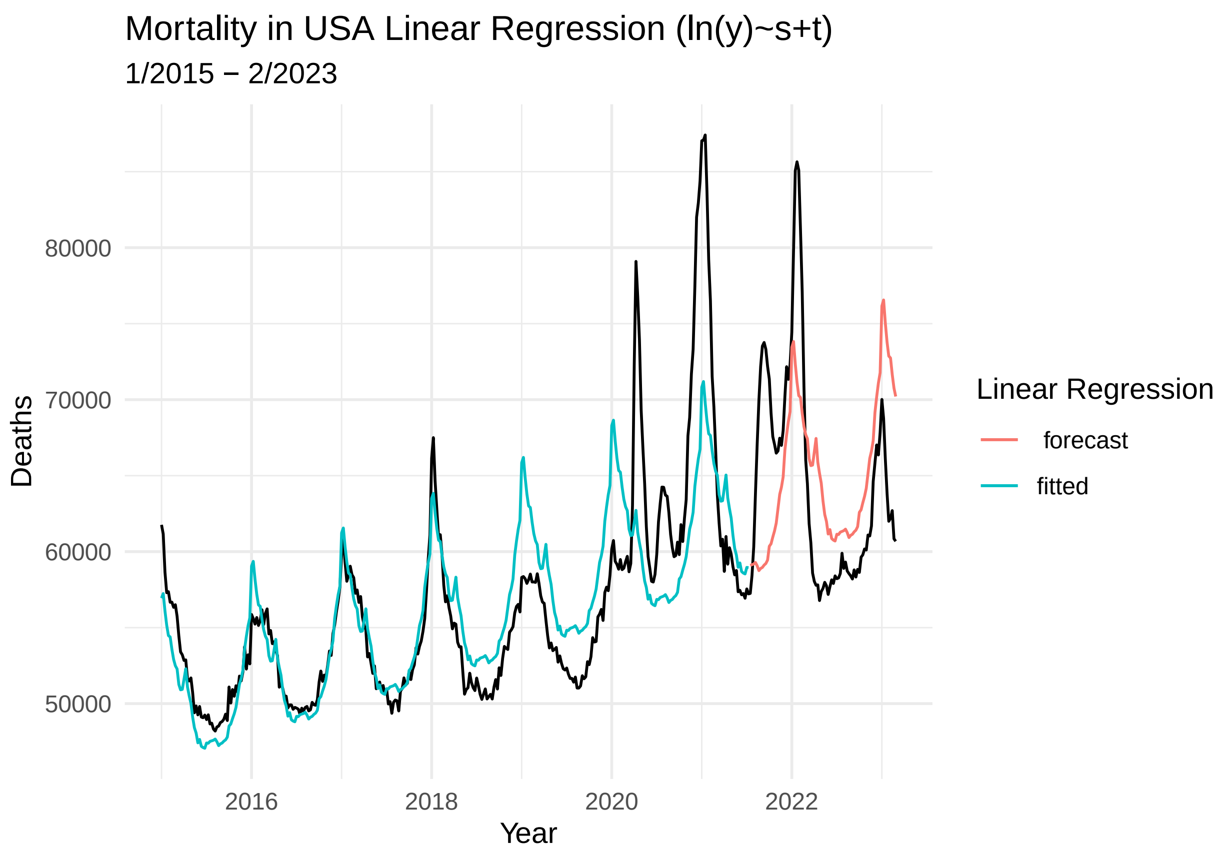
```
##                        ME       RMSE        MAE          MPE      MAPE
## Training set 6.270777e-17 0.06448101 0.04687401 -0.003380151 0.4258694
##                   MASE      ACF1
## Training set 0.7531913 0.9616223
```

```r
# mape 0.4258694

# final lm model
country.lm <- tslm(log(train.ts)~ trend + season, lambda=NULL)
country.lm.fc <- forecast(country.lm,h=future)
forecast::accuracy(country.lm.fc, valid.ts)
```

```
##                         ME         RMSE          MAE          MPE       MAPE
## Training set -4.176698e-17 6.448101e-02 4.687401e-02 -0.003380151  0.4258694
## Test set      6.461939e+04 6.500931e+04 6.461939e+04 99.982671444 99.9826714
##                      MASE      ACF1 Theil's U
## Training set 7.531913e-01 0.9616223        NA
## Test set     1.038332e+06 0.9585577  38.23552
```

```r
#  mape: train= 0.4258694, test = 99.9826714

autoplot(country.ts) +
  autolayer(exp(country.lm$fitted.values), PI=F, series="fitted") +
  autolayer(exp(country.lm.fc$mean), series = " forecast")+
  ggtitle(paste0("Mortality in ", country_code, " Linear Regression (ln(y)~s+t)"),
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="Linear Regression"))
```

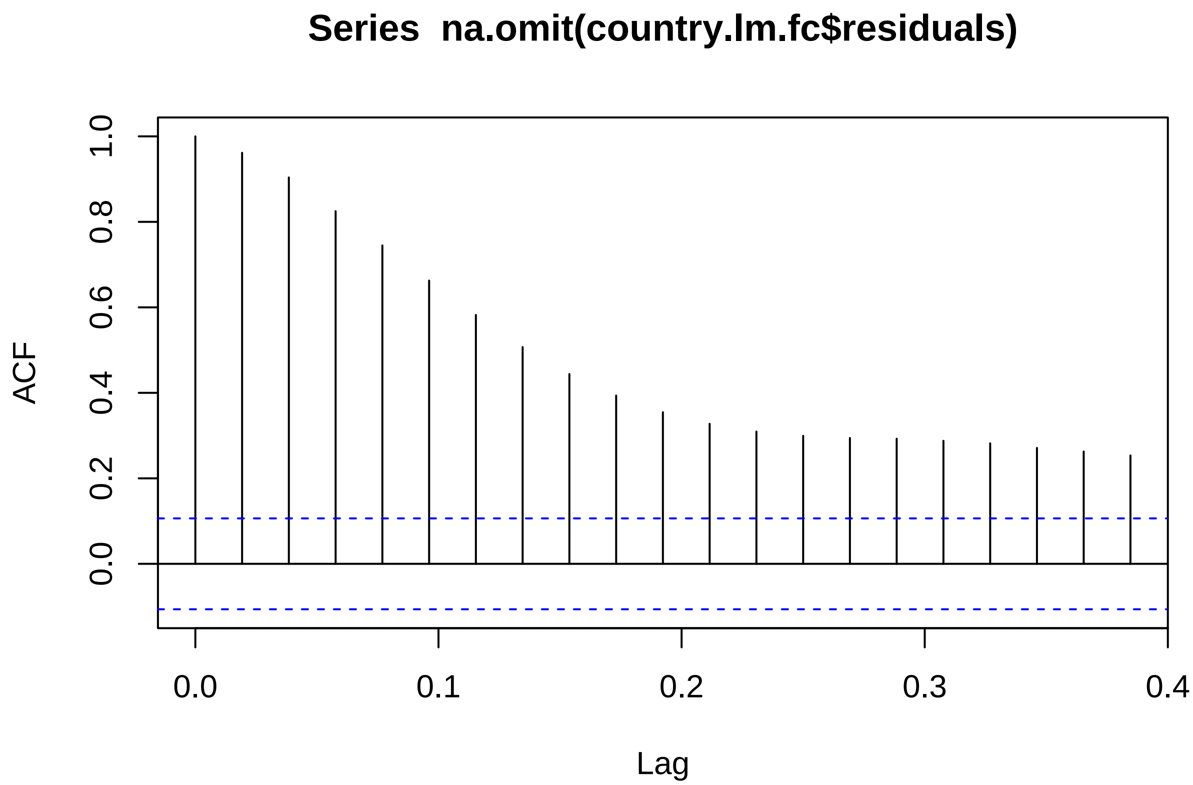
```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y =
## .data[["seriesVal"]], : Ignoring unknown parameters: `PI`
```




```
country.lm
```

```
##
## Call:
## tslm(formula = log(train.ts) ~ trend + season, lambda = NULL)
##
## Coefficients:
## (Intercept)        trend      season2      season3      season4      season5
##   10.949092     0.000699     0.004547    -0.016308    -0.033887    -0.047049
##     season6      season7      season8      season9     season10     season11
##   -0.049354    -0.065235    -0.078424    -0.086962    -0.091705    -0.112080
##    season12     season13     season14     season15     season16     season17
##   -0.119692    -0.119851    -0.107349    -0.094909    -0.118964    -0.130976
##    season18     season19     season20     season21     season22     season23
##   -0.141607    -0.160590    -0.175472    -0.183548    -0.197699    -0.193596
##    season24     season25     season26     season27     season28     season29
##   -0.203943    -0.206333    -0.208037    -0.201393    -0.202446    -0.200515
##    season30     season31     season32     season33     season34     season35
##   -0.200475    -0.200368    -0.199401    -0.203820    -0.209925    -0.208137
##    season36     season37     season38     season39     season40     season41
##   -0.207721    -0.205808    -0.204761    -0.201643    -0.187132    -0.185217
##    season42     season43     season44     season45     season46     season47
##   -0.178645    -0.172756    -0.164975    -0.150572    -0.136308    -0.129738
##    season48     season49     season50     season51     season52
##   -0.119964    -0.093855    -0.079548    -0.067418    -0.058610
```

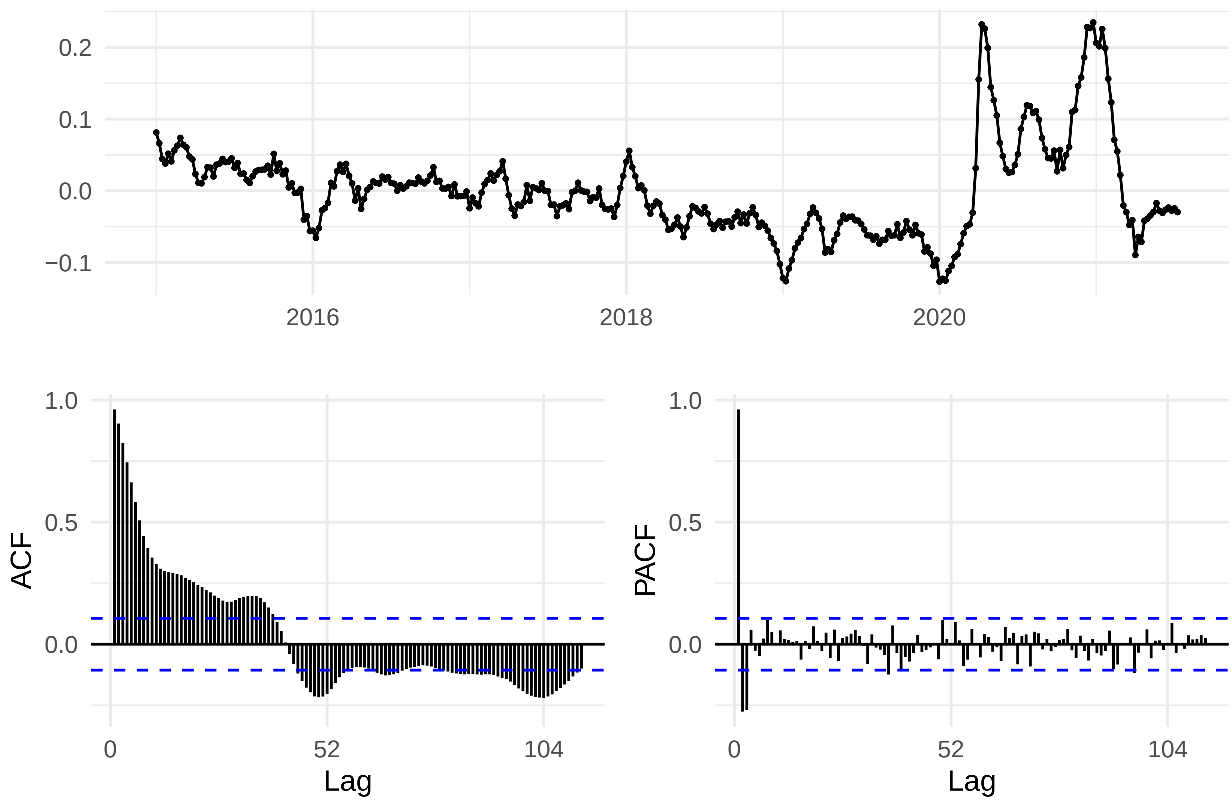
```r
acf(na.omit(country.lm.fc$residuals), lag.max = 20)
```



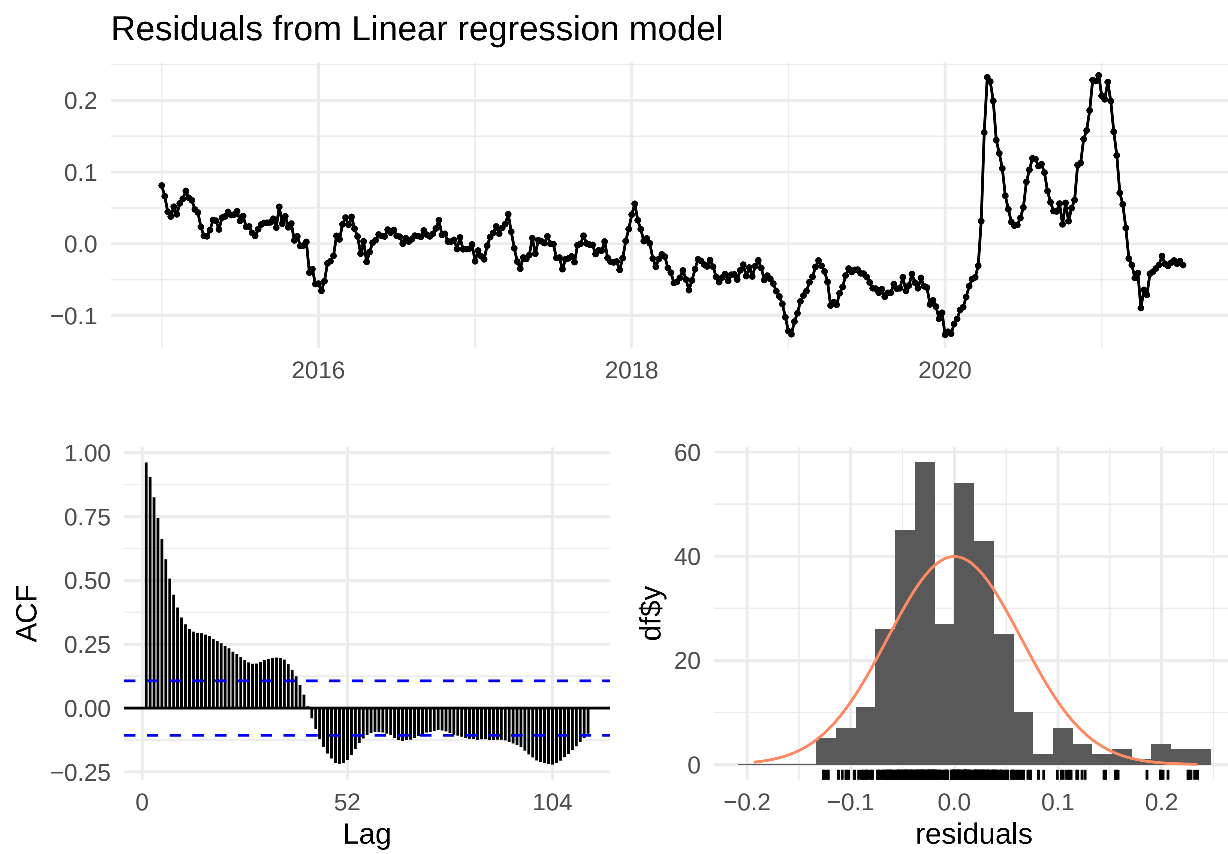

```r
Box.test(country.lm.fc$residuals, lag=20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  country.lm.fc$residuals
## X-squared = 1851.9, df = 20, p-value < 2.2e-16
```

```r
ggtsdisplay(country.lm.fc$residuals)
```




```
checkresiduals(country.lm)
```



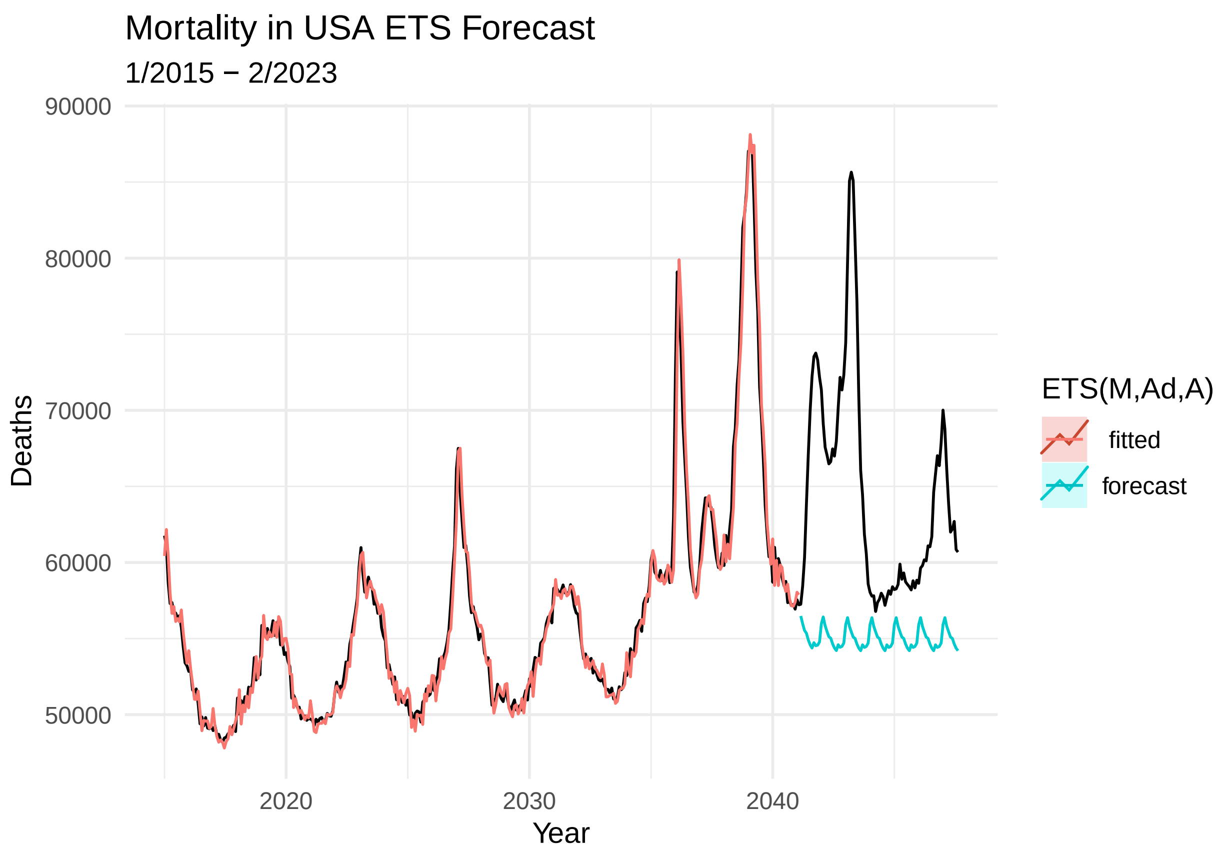

```
##
##  Breusch-Godfrey test for serial correlation of order up to 68
##
## data:  Residuals from Linear regression model
## LM test = 323.72, df = 68, p-value < 2.2e-16
```

```
# can't use default train.ts because returns error for frequency=52
ets.ts <- ts(country$Value, start=c(startYear,startWeek), frequency=13)
ets.train.ts <- window(ets.ts, start = c(startYear, startWeek),
                                   end = c(startYear, nTrain))
ets.valid.ts <- window(ets.ts, start = c(startYear, nTrain+1),
                       end = c(startYear, nTrain+nValid))

# if use ZZZ don't get any seasonality
a.ets <- ets(ets.train.ts, model="ZZZ", alpha = NULL)
a.ets <- ets(ets.train.ts, model="ZZA", alpha = NULL)

# get prediction
a.ets.fc <- forecast(a.ets, h = future, level = 0)
a.ets
```

```
## ETS(M,Ad,A)
##
## Call:
##  ets(y = ets.train.ts, model = "ZZA", alpha = NULL)
##
##   Smoothing parameters:
```

```
##     alpha = 0.9997
##     beta  = 0.124
##     gamma = 1e-04
##     phi   = 0.8001
##
##   Initial states:
##     l = 59748.0635
##     b = -286.6304
##     s = -301.2254 -524.9653 -585.354 -413.9644 -801.7451 -633.9307
##            -363.1328 5.6935 94.7059 437.1969 804.6165 1371.303 910.8021
##
##   sigma:  0.0197
##
##      AIC     AICc      BIC
## 6764.042 6766.417 6836.792
```

```r
autoplot(ets.ts) +
  autolayer(a.ets.fc, PI=T, series="forecast") +
  autolayer(a.ets$fitted, series = " fitted")+
  ggtitle(paste0("Mortality in ", country_code, " ETS Forecast "),
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="ETS(M,Ad,A)"))
```



```r
forecast::accuracy(a.ets.fc, ets.valid.ts)
```

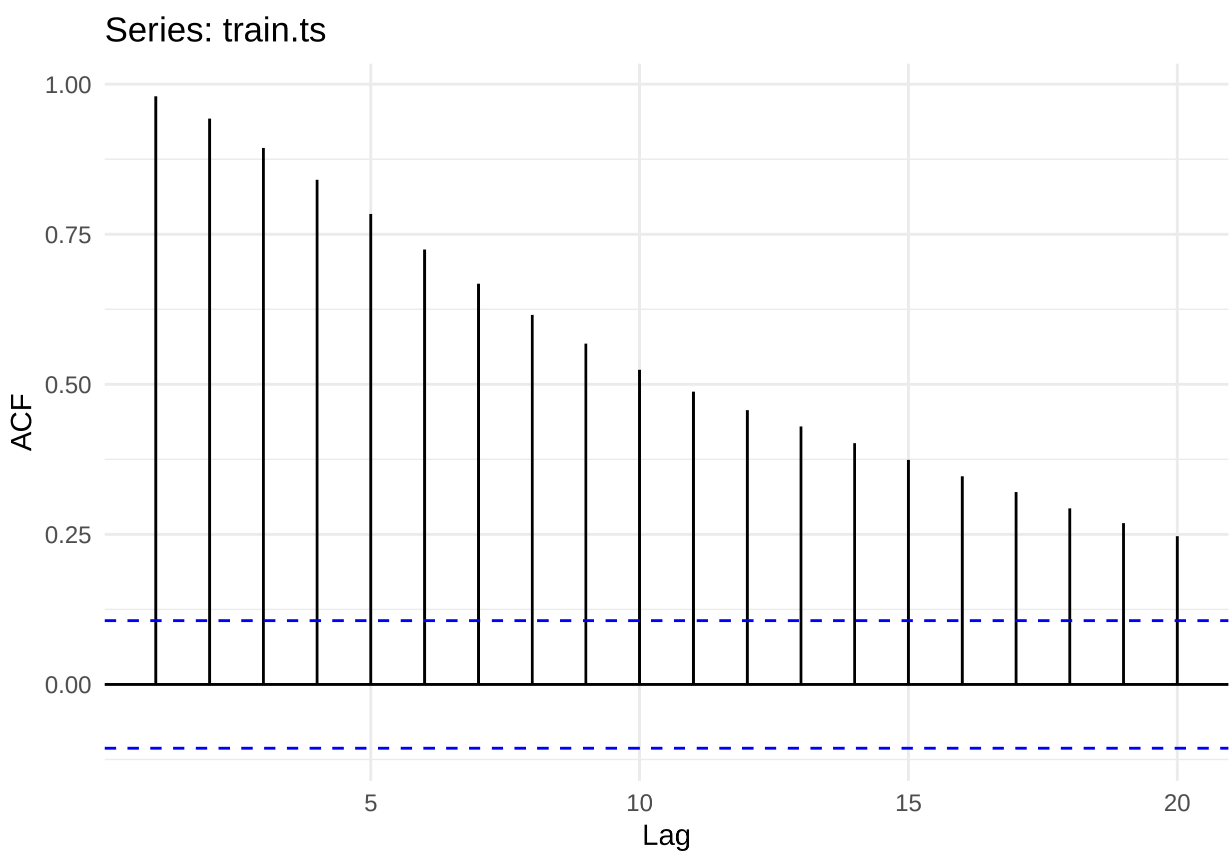
```
##                       ME      RMSE       MAE          MPE      MAPE      MASE
## Training set   -7.072029  1205.222  815.8181 -0.01508594  1.385793 0.1534662
## Test set     9592.288290 11946.367 9592.2883 13.90043923 13.900439 1.8044361
##                   ACF1 Theil's U
## Training set 0.3288586        NA
## Test set     0.9558169  6.330288
```

Autocorrelation

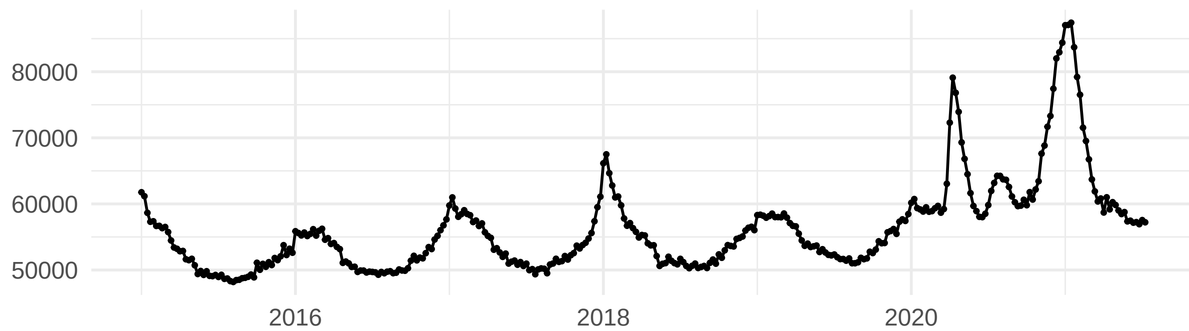
The ACF values are all above the threshold. The ACF of the time series is decreasing slowly, which suggests that it is non-stationary.
You can't predict values for non-stationary data.

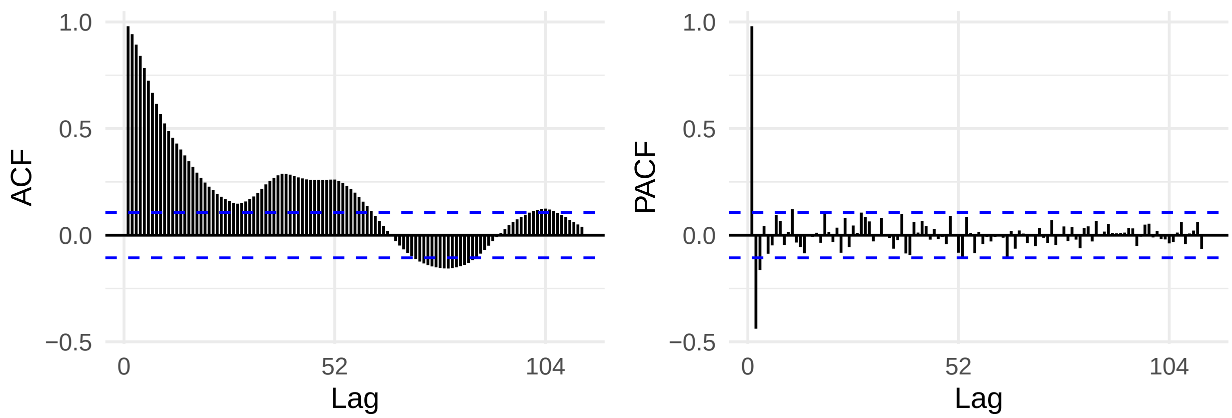
There is a clear seasonal (and cyclical) effect, (Shmueli and Lichtendahl 2016)

```r
# look at autocorrelation
ggAcf(train.ts, lag=20)
```




```r
ggtsdisplay(train.ts)
```





ARIMA

An AR (autoregressive) model is usually used to model a time series which shows longer term dependencies between successive observations. Intuitively, it makes sense that an AR model could be used to describe the time series of mortality, as we would expect some factors which affect mortality rates in one year to affect those in later years.

```
tunes <- readRDS(file="data/arimaManualFit.rds")

best <- tunes[tunes$mape==min(tunes$mape),1:6] %>% as.double()
print(best)
```

```
## [1] 1 0 2 2 2 2
```

```
tunes %>% arrange(mape)  %>% head()
```

```
## # A tibble: 6 x 15
##       a     b     c     d     e     f   aic  aicc    me  rmse   mae     mpe
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1     1     0     2     2     2     2 5714. 5715.  38.5  873.  555.  0.0482
## 2     2     0     1     2     2     2 5725. 5725.  57.0  897.  562.  0.0755
## 3     1     0     1     2     2     2 5767. 5767.  40.5  937.  587.  0.0500
## 4     1     0     0     2     2     2 5821. 5822.  39.4 1034.  618.  0.0473
## 5     0     1     2     2     2     2 5710. 5710. -19.9  968.  606. -0.0308
## 6     0     1     1     2     2     2 5758. 5758. -30.8 1007.  621. -0.0546
## # i 3 more variables: mape <dbl>, mase <dbl>, acf1 <dbl>
```

```r
# fit best manual arima model
#MANUAL
country.am.fc <- (country.am <- arima(train.ts, order = c(best[1:3]),
                    seasonal = c(best[4:6]))) %>% forecast()

##AUTO
country.aa.fc <-(country.aa <- auto.arima(train.ts)) %>% forecast()
#ARIMA(3,1,0)(0,0,2)[52]

# accuracy scores
print(forecast::accuracy(country.am.fc, valid.ts))
```

```
##                      ME     RMSE       MAE         MPE      MAPE      MASE
## Training set   49.28524 1013.021  626.5231  0.06686918  1.058339 0.1651212
## Test set     -791.75378 8580.879 6670.1722 -1.97661934 10.051741 1.7579348
##                    ACF1 Theil's U
## Training set 0.03455536        NA
## Test set     0.96428426  4.881296
```

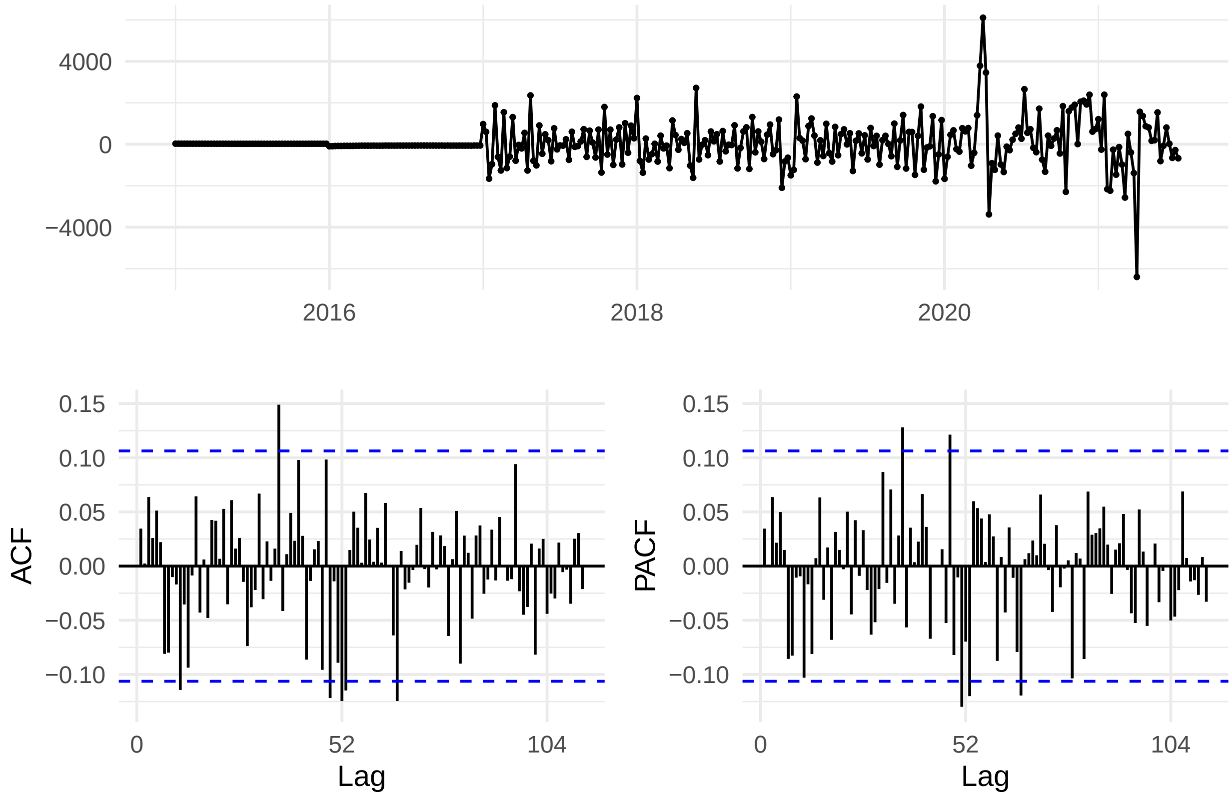
```r
print(forecast::accuracy(country.aa.fc, valid.ts))
```

```
##                       ME     RMSE       MAE         MPE     MAPE      MASE
## Training set   -5.377429 1160.430  791.2118 -0.01086611 1.357587 0.2085252
## Test set     6457.400499 9169.264 6647.6901  9.10709301 9.438398 1.7520096
##                     ACF1 Theil's U
## Training set 0.003307442        NA
## Test set     0.955775638  4.761724
```

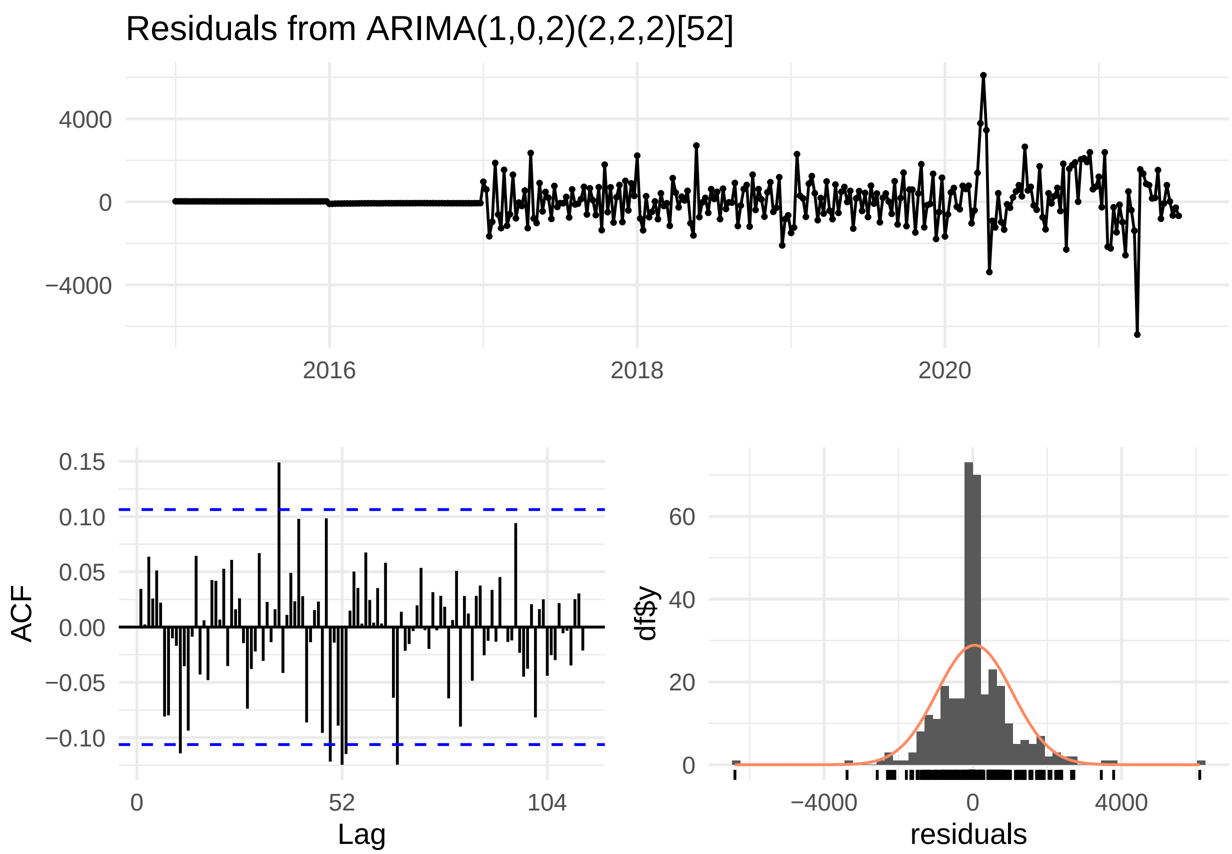
```r
#https://www.educba.com/arima-model-in-r/
```

Plotting the arima model:

```r
ggtsdisplay(resid(country.am))
```

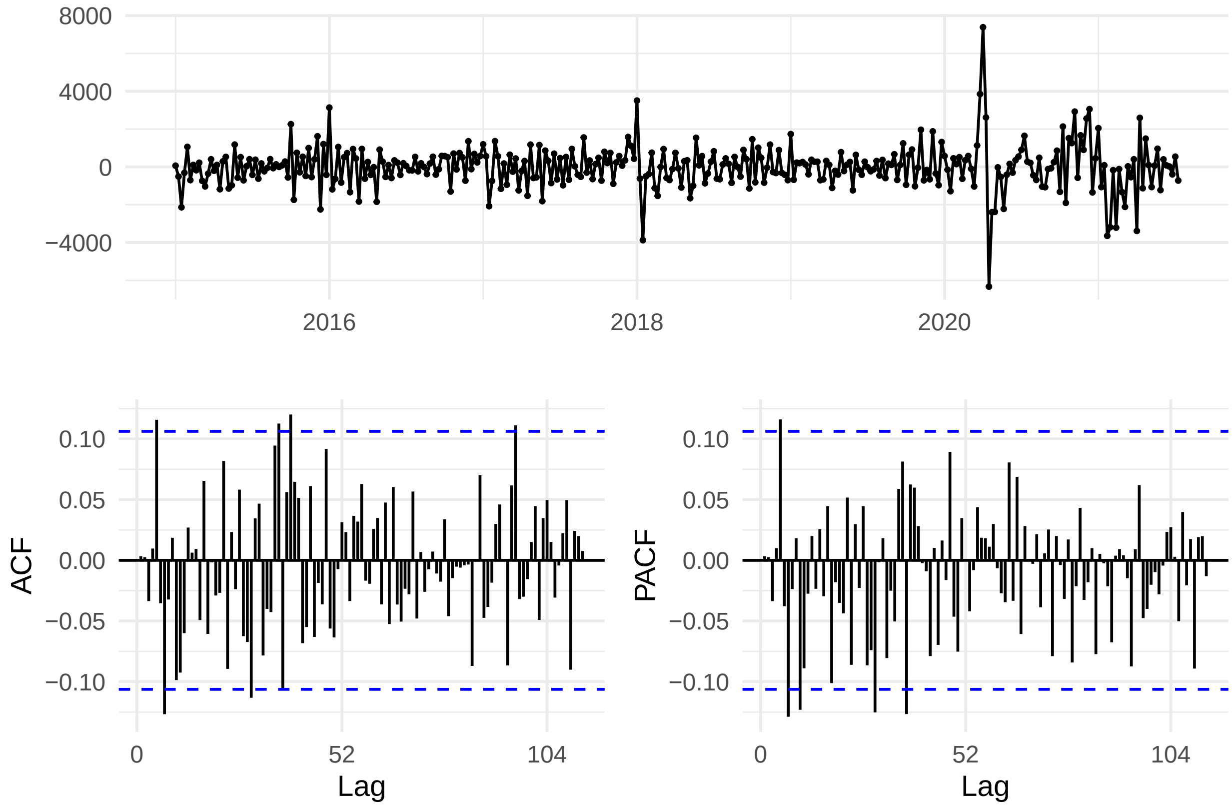



```
checkresiduals(country.am)
```

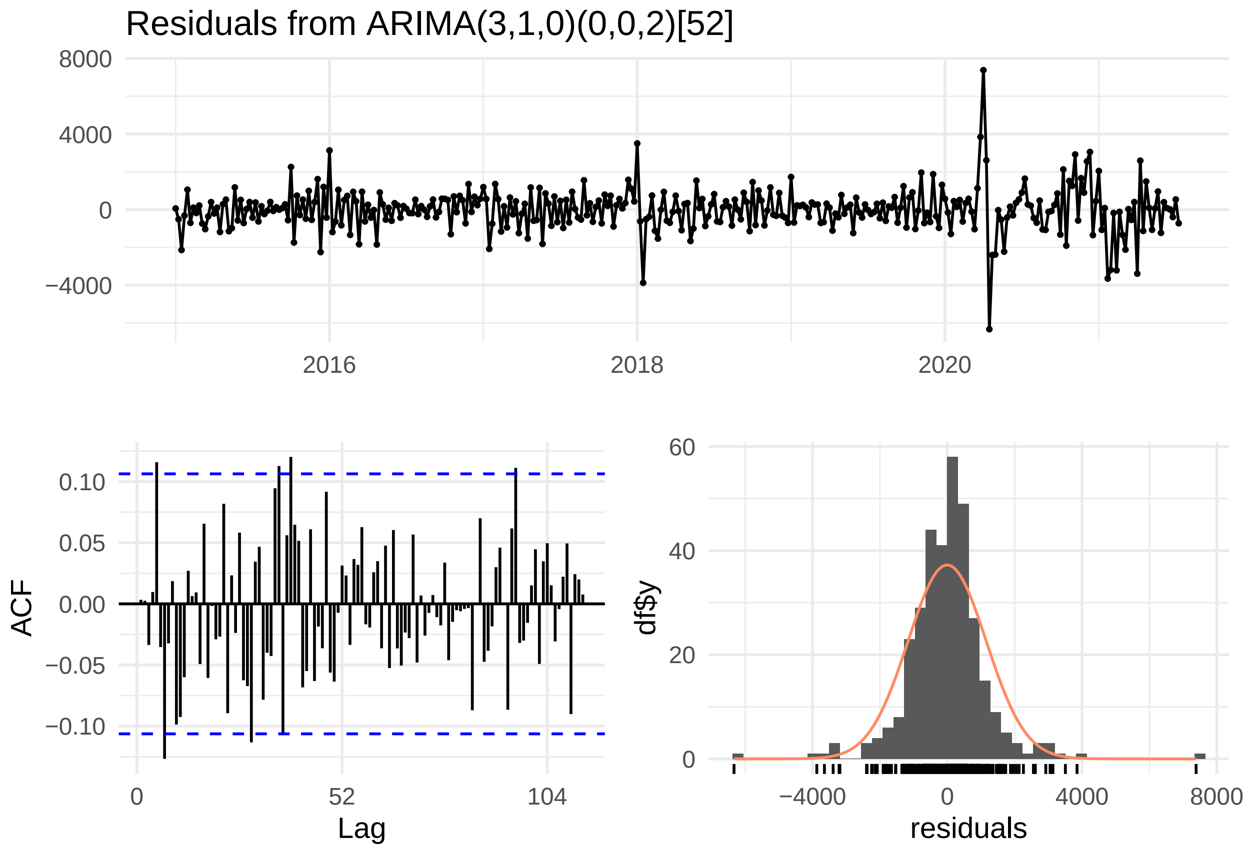



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2)(2,2,2)[52]
## Q* = 88.947, df = 61, p-value = 0.01128
##
## Model df: 7.   Total lags used: 68
```

```
ggtsdisplay(resid(country.aa))
```



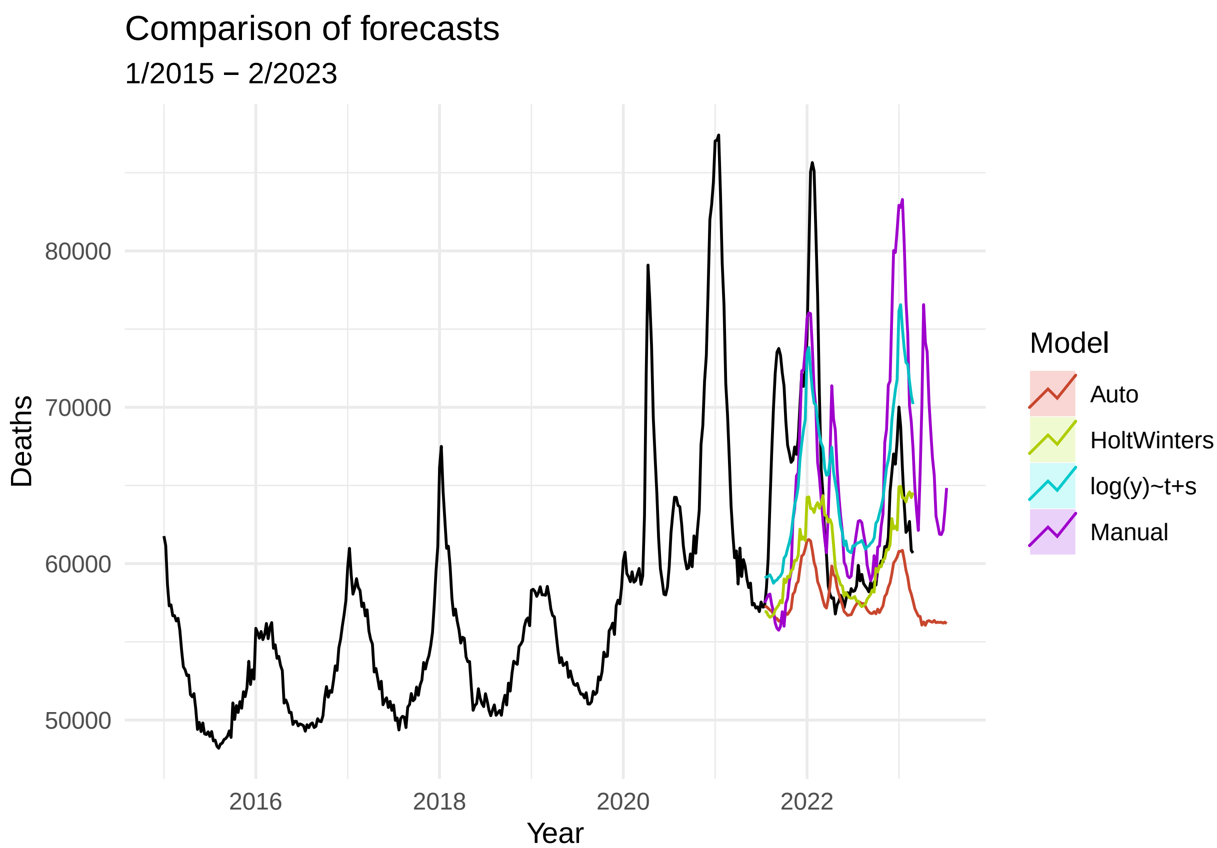

```
checkresiduals(country.aa)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,0)(0,0,2)[52]
## Q* = 89.649, df = 63, p-value = 0.01535
##
## Model df: 5.   Total lags used: 68
```

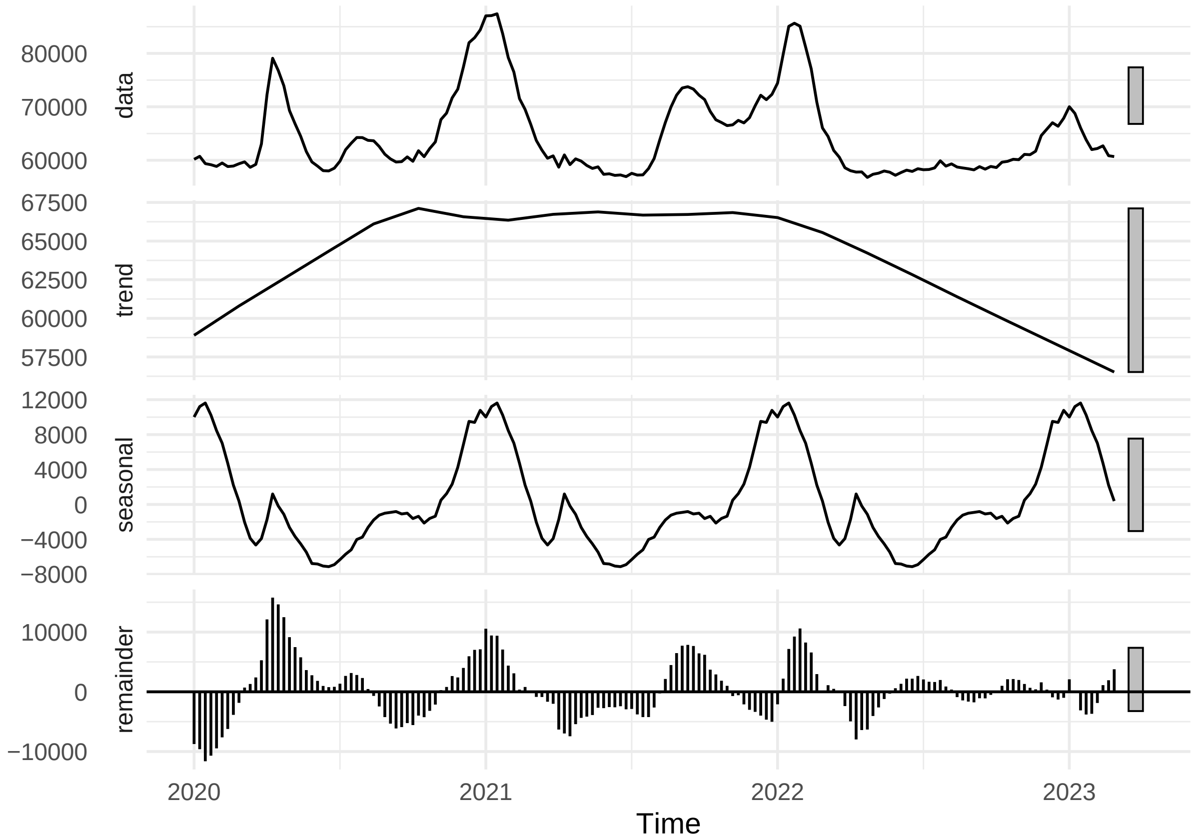
```
#---------------------------------------------------------------------
# compare forecasts for individual models
autoplot(country.ts) +
  autolayer(country.aa.fc, PI=F, series="Auto") +
  autolayer(country.am.fc, PI=F, series = "Manual")+
  autolayer(exp(country.lm.fc$mean), PI=F, series = "log(y)~t+s")+
  autolayer(country.hw.fc, PI=F, series = "HoltWinters")+
  ggtitle("Comparison of forecasts",
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="Model"))
```

```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y =
## .data[["seriesVal"]], : Ignoring unknown parameters: `PI`
```

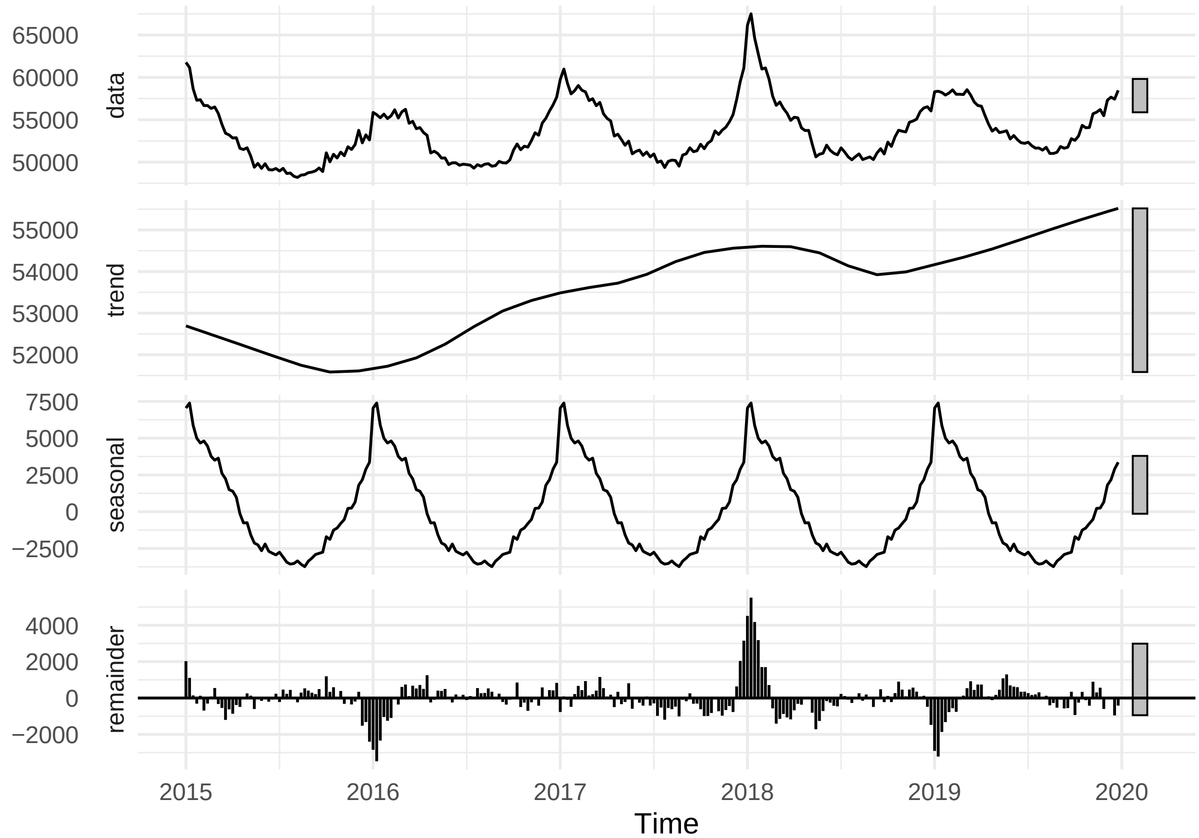


Appendix 1 - training on pre-COVID data only

```
covid <- length(country[country$YEAR>= 2020,]$date)

nTrain <- length(country.ts) - covid
train.ts <- window(country.ts, start = c(startYear, startWeek),
                   end = c(startYear, nTrain))
covid.ts <- window(country.ts, start = c(startYear, nTrain+1),
                   end = c(startYear, nTrain+covid))


# restricting the decomposition to pre-covid changes the shape of the seasonal component slightly and m
covid.ts %>% stl(s.window="periodic") %>% autoplot()
```




```
train.ts %>% stl(s.window="periodic") %>% autoplot()
```



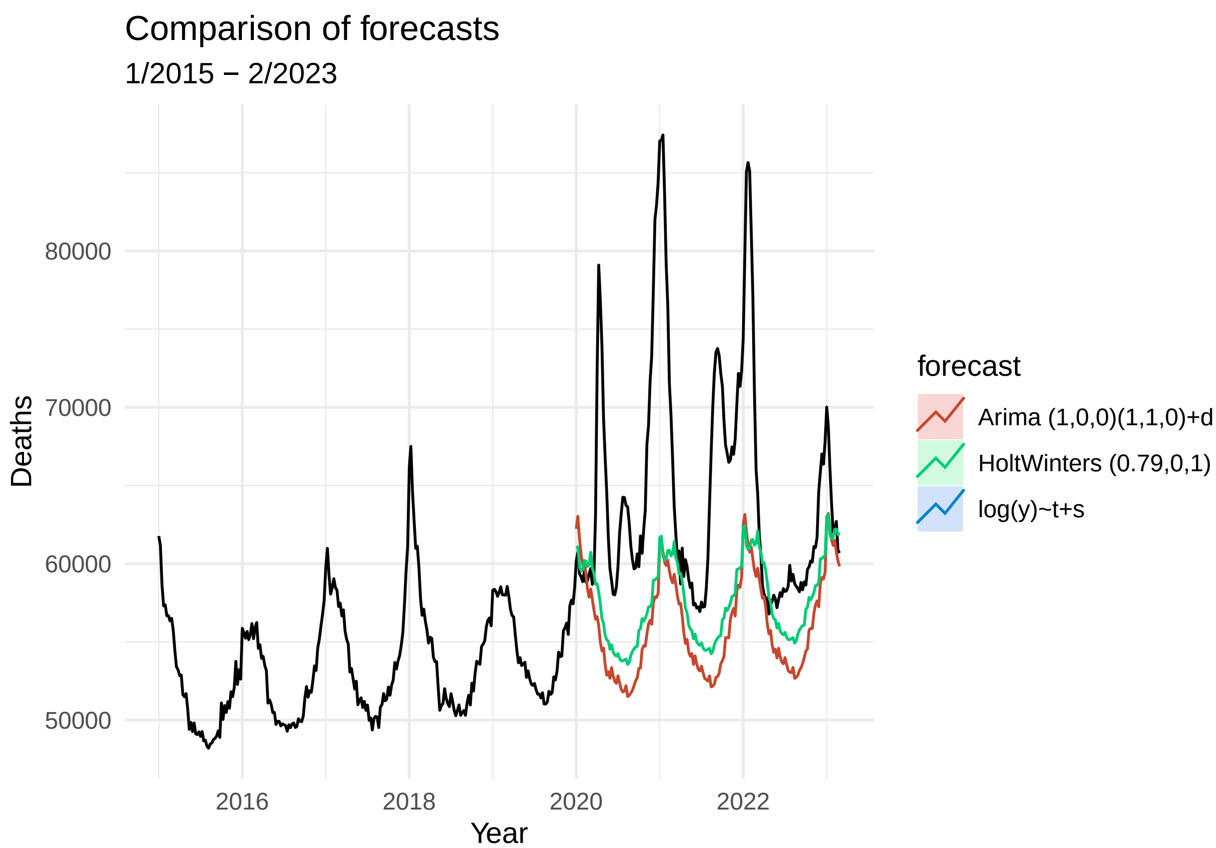

```r
#summary(fit.lm)
# adj r^2 = 0.8866
# significant trend coefficient  2.719e-04
#fit.lm$coefficients[2]

covid.arima <- auto.arima(train.ts) %>% forecast(covid)
covid.hw <- HoltWinters(train.ts) %>% forecast(covid)
covid.lm <-tslm(log(train.ts)~ trend + season, lambda=NULL) %>% forecast(covid)
```

```
## Warning in forecast.lm(., covid): newdata column names not specified,
## defaulting to first variable required.
```

```r
# plot model forecasts
autoplot(country.ts) +
  autolayer(covid.arima, PI=F, series="Arima (1,0,0)(1,1,0)+d") +
  autolayer(exp(covid.lm$mean), PI=F, series = "log(y)~t+s")+
  autolayer(covid.hw, PI=F, series = "HoltWinters (0.79,0,1)")+
  ggtitle("Comparison of forecasts",
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="forecast"))
```

```
## Warning in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y =
## .data[["seriesVal"]], : Ignoring unknown parameters: `PI`
```

```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```




```r
ggsave(here("docs", "noCovid.png"))
```

```
## Saving 6.5 x 4.5 in image
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

All the models fit well to the training data, and are consistent in predicting the mortality that would have been expected in 2020+ if COVID-19 hadn't happened, so they could be used to estimate the excess mortality in the USA due to COVID.

Thu May 4 00:12:59 2023

refs:

https://github.com/FinYang/tsdl/tree/master The Time Series Data Library (TSDL) was created by Rob Hyndman, Professor of Statistics at Monash University, Australia.

https://www.rdocumentation.org/packages/forecast/versions/8.21/topics/tslm

Coghlan, Avril. 2023. "Welcome to a Little Book of r for Time Series!" 2023. https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/.

Dancho, Matt. 2023. *Modeltime: The Tidymodels Extension for Time Series Modeling*.

Hyndman, Rob J, and George Athanasopoulos. 2023. *Forecasting: Principles and Practice*. 3rd ed. Monash University. https://otexts.com/fpp3/.

OECD. 2023. "COVID-19 Health Indicators, Mortality (by Week)." https://doi.org/https://doi.org/https://doi.org/10.1787/cd2bda32-en.

Shmueli, Galit, and Kenneth C Lichtendahl. 2016. *Practical Time Series Forecasting with r: A Hands-on Guide [2nd Edition]*. Practical Analytics. Axelrod Schnall Publishers. https://books.google.ie/books?id=mxWXDwAAQBAJ.