Social Forecasting

Mortality in the USA, 2015-2023

Imelda Finn

4th May 2023

Student ID Number	22334657
Module	POP77014
	Social Forecasting
Lecturer	Thomas Chadefaux

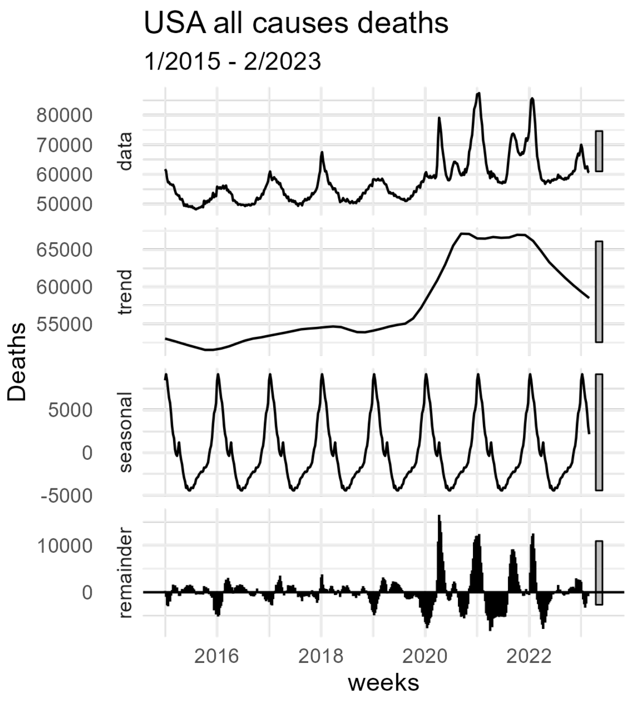

Figure 1: Mortality Data - USA all cause mortality

1 Introduction

1. The data is the USA data from the OECD website: COVID-19 Health Indicators, Mortality (by week)[OECD, 2023] The data shows a gradual upwards trend and clear seasonality from 2015 to 2019.(see Figure 1)

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.

2 Method - Ensemble Model

The data prior to 2020 was quite stable, so could be modelled using a number of different algorithms. (see Appendix-2) COVID-19 led to a sudden surge in deaths, which meant that including data from 2020 resulted in non-stationary, auto-correlated, data. None of the models produced residuals which could be considered normal and independently distributed.

The models either failed to incorporate the surge in mortality, so under-predicted, or included too much of the surge, so over-predicted. The HoltWinters model, with a high α parameter selected by the function, included most of the up-to-date data. Even so, it still underestimated the deaths in 2021 and beyond. (see Appendix-1)

An ensemble model was run, to try and compensate for the deficiencies of the individual models. 7 models were combined using the `modeltime` library; the models and results are shown below. MAE is 3669 and MAPE is 6.26. [Dancho, 2023]

- ARIMA(2,1,0)(0,0,1)[13][1]
- ARIMA(3,1,0)(1,0,0)[13] W/ XGBOOST ERRORS
- ARIMA(3,1,0)(0,0,2)[52] (from `auto.arima`)
- ETS(M,AD,N)
- PROPHET (linear growth, 25 change points, weekly seasonality)
- LM (formula = Value   as.numeric(date) + factor(month(date, label = TRUE), ordered = FALSE))
- EARTH (MARS regression)

[1]frequency is 13 per quarter, ie 52

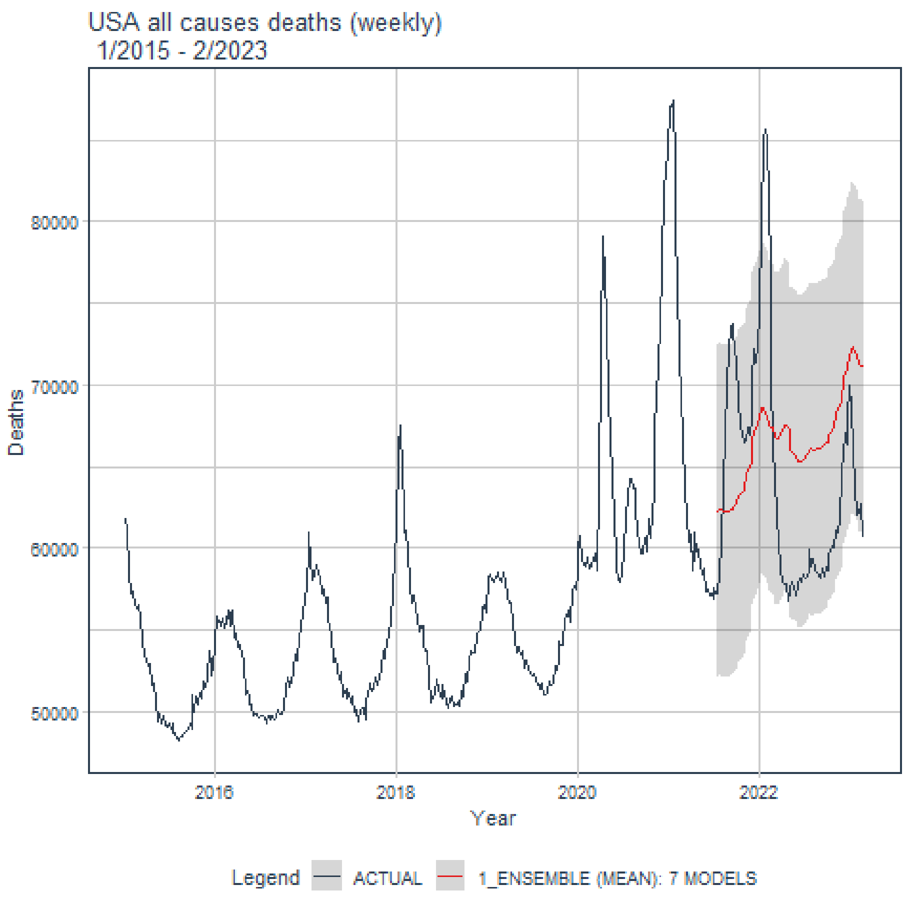

Figure 2: Ensemble model (modeltime)

model	mae	mape	mase	smape	rmse	rsq
ensemble (mean)	3669.	6.26	4.19	6.28	5179.	0.623

Table 1: Ensemble model results

2.1 Alternates

Individual methods tried were:

HoltWinters best model on test set ($\alpha = 0.94, \beta = 0, \gamma = 1$)

Linear Regression - $\ln y \sim trend + season$ ($\hat{Y} = \alpha + \beta_t \times t + \Sigma_{k=1}^{52} \beta_k \times x_{t-k}$)

This gave the best results for the training set, but created a linear trend, including the surge due to COVID, so ended up with the worst results for the test set. (Over predicts on test, so penalised more by MAPE)

ARIMA auto arima: ARIMA(3,1,0)(0,0,2)[52]

($X_t - mu = (\beta_1 * (X_t - 1 - mu)) + (\beta_2 * (Xt - 2 - mu)) + (\beta_3 * (Xt - 3 - mu)) + Z_t$), where X_t is the time series, mu is the mean of X_t, β_1,β_2 and β_3 are parameters to be estimated, and Z_t is white noise with mean zero and constant variance.

ETS the ets function couldn't handle 52-week seasonality, so was run with 13 week frequency, but failed to capture seasonal component.

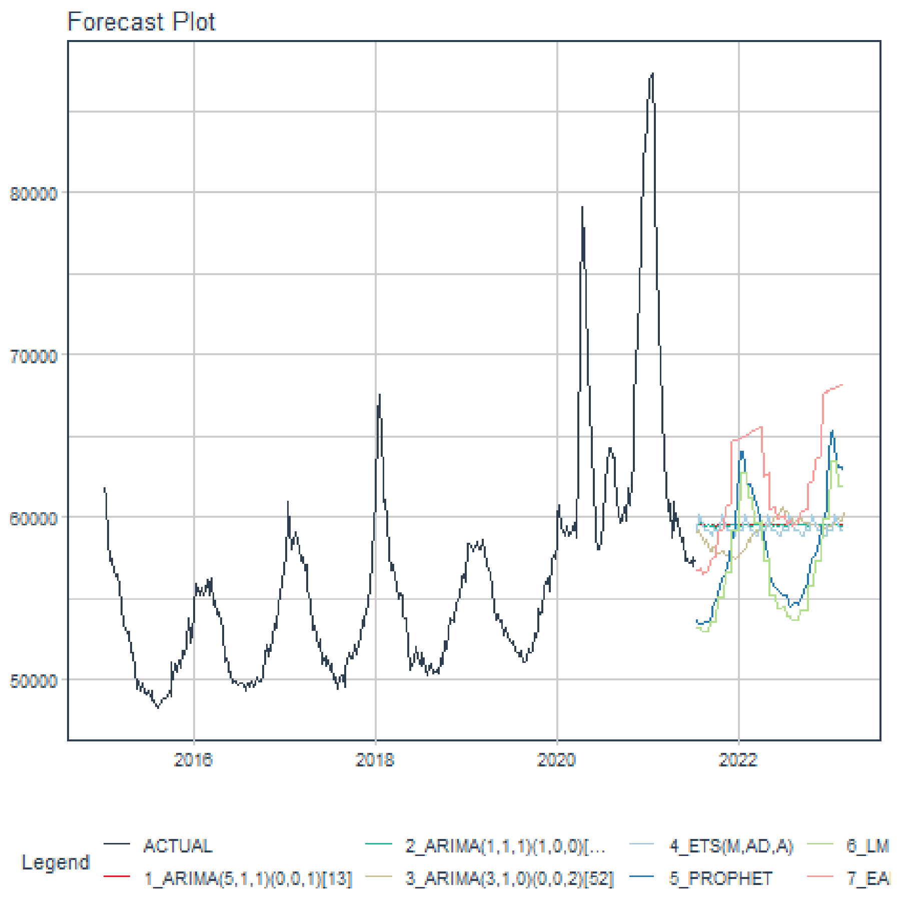

Figure 3: Ensemble models

Appendix-1 Individual Models

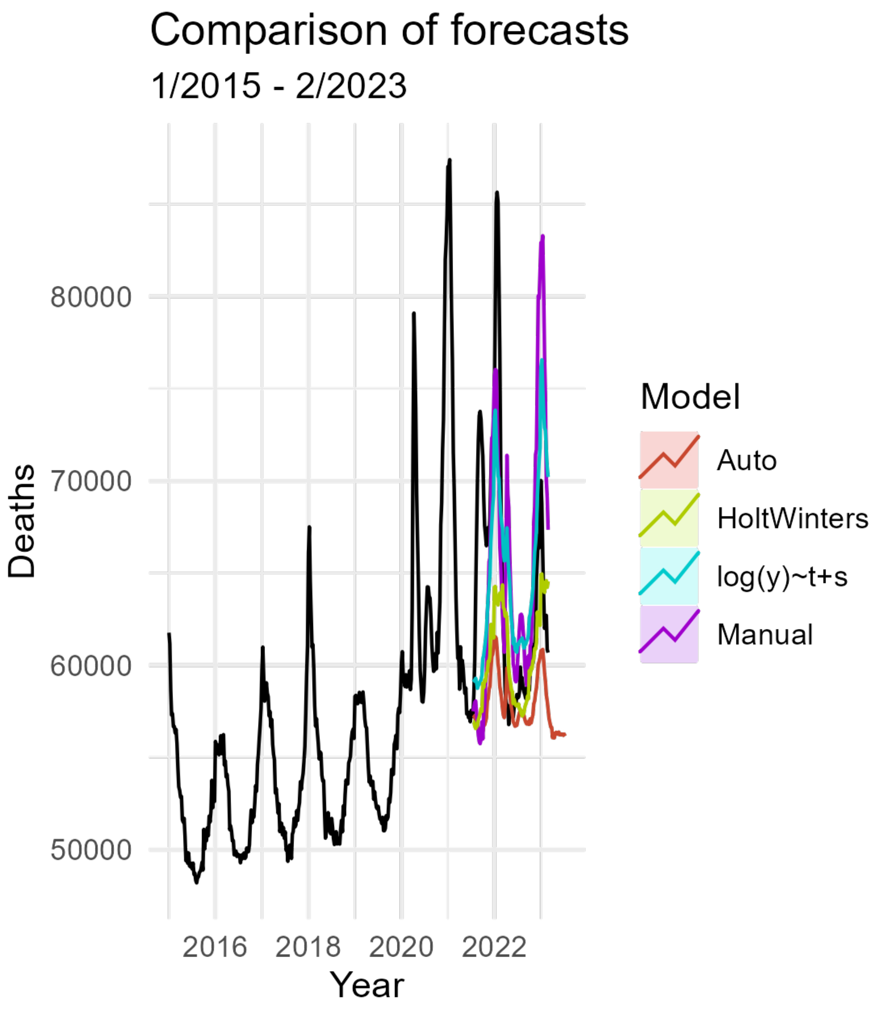

Figure 4: Individual models

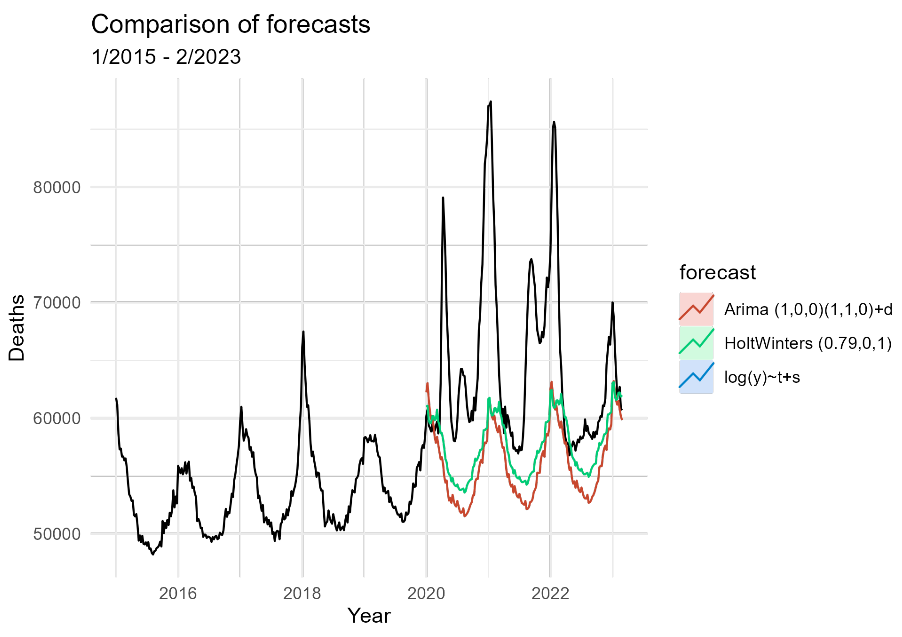

Figure 5: Models trained only on pre-COVID data

Appendix-2 Pre/Post COVID-19

The 80:20 split means that the training data used above includes 2020 and part of 2021. I wanted to see how the models would perform if they were trained on pre-2020 only.

Restricting the decomposition to pre-covid data changes the shape of the seasonal component slightly and makes the upward trend more pronounced. 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.

Appendix-3 Code

All the data and code are in

https://github.com/I-finn/asds_samples/forecasting/

code files: mortality.R, mortality.Rmd, ensemble.R

```
#####################
# Imelda Finn
# 22334657
# Social Forecasting
# POP77014
#####################

rm(list=ls())


# detach all libraries
detachAllPackages <- function() {
  basic.packages <- c("package:stats", "package:graphics", "package:grDevices",
  package.list <- search()[ifelse(unlist(gregexpr("package:", search()))==1, TRU
  package.list <- setdiff(package.list, basic.packages)
  if (length(package.list)>0)  for (package in package.list) detach(package,
character.only=TRUE)
}
detachAllPackages()
print(ensemble_fit$model_tbl$.model_desc)
# load libraries
pkgTest <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[,  "Package"])]
  if (length(new.pkg))
```

```
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}
##```{r libraries, message=FALSE, eval=TRUE, include=FALSE}

#genl
lapply(c("ggplot2", "tidyverse", "lubridate", "here", "patchwork"),
pkgTest)
#specific
lapply(c("forecast", "zoo", "fpp2", "ISOweek", "tsfeatures", "fUnitRoots"),
pkgTest)

setwd(here())
getwd()

stackPlot <- function(x) Reduce("/", x)

# Get Data ------------------------------------------------

# settings
theme_set(theme_minimal())

#https://stats.oecd.org/Index.aspx?DataSetCode=HEALTH_MORTALITY
mort <- read_csv("data/HEALTH_MORTALITY_all.csv")
ds <- mort
ds$date <- ISOweek2date(paste0(ds$YEAR, "-W", sprintf("%02d", ds$WEEK),"-4"))
ds <- ds %>% filter(Age == "Total" & Gender == "Total" & VARIABLE == "ALLCAUNB"
```

```
  arrange(COUNTRY, date)

summary(ds)
unique(ds$COUNTRY)

ds %>% ggplot(aes(x=date, y=Value, colour = COUNTRY)) + geom_line()

head(mort)
unique(mort[c("WEEK", "YEAR")]) %>% filter(YEAR==2015)

mort %>% filter(YEAR==2015, WEEK ==1) %>%select(Country, COUNTRY) %>%
  unique()

length(ds$date)

## create country dataframe
country_code <- "USA"
country <- ds %>% filter(COUNTRY == country_code) #%>% select(date, Value)
country %>% arrange(date)
length(country$date)

save(country, file = here("data", "country.Rdata"))

cycle(country.ts)

#Time Series:
#  Start = c(2015, 1)
```

```
#End = c(2023, 9)
#Frequency = 52



# first in data is week 1, 2015
# 2020 is leap year - so 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)
```

```
fivenum(country$Value)
#usa: 48194 51838 56680 60260 87415

# default graph labels
mtitle <- paste0(country_code," all causes deaths ")
stitle <- paste0(startMonth, "/", startYear, " - ", endMonth, "/",endYear)
y_lab <- "Deaths"
x_lab <- "weeks"
ire_y_lim <- c(450, 1000)
us_y_lim <- c(48000, 90000)      # check when incorporate leap year
y_lim <- us_y_lim
x_lim <- c(startDate, endDate)

# #```
#
# Data is weekly, `r WEEKS` weeks from `r startDate` to `r endDate` inclusive.
#
#

# #```{r makeTS, eval=TRUE, include=TRUE}
#Convert the death numbers to a time series
country.ts <- ts(country$Value, start=c(startYear,startWeek), frequency=FREQ) #%
print(tsfeatures(country.ts))
summary(country.ts)

#unit root tests are available in the fUnitRoots package, available on CRAN, but
```

```
require("fUnitRoots")
unitrootTest(country.ts)
# Fri Apr 28 21:35:44 2023 ------------------------------



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

country %>% ggplot(aes(x=date, y=Value, colour = COUNTRY)) + geom_line()

#```

#```{r makeTrain, eval=TRUE, include=TRUE}
# forecast parameter & train/test sets
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))

#```
```

```
#```{r decomposition, eval=TRUE, include=TRUE}
#plot the decomposition
# country or train?
(country.stl <- train.ts %>%
    stl(s.window="periodic"))%>%  #, t.window = 5
  autoplot() +
  ggtitle(mtitle, subtitle = stitle) +
  xlab(x_lab) +
  ylab(y_lab)

(country.stl <- country.ts %>%
    stl(s.window="periodic"))%>%  #, t.window = 5
  autoplot() +
  ggtitle(mtitle, subtitle = stitle) +
  xlab(x_lab) +
  ylab(y_lab)
ggsave(here("docs", "stl.png"))
#```

#```{r ensemble, eval=TRUE, include=TRUE}

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

country %>%  plot_time_series(date, Value, .interactive = interactive,
                              .title = mtitle, .x_lab = x_lab, .y_lab = y_lab)
```

```
dlimit <- head(tail(country,future),1)$date

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 13 weeks
model_fit_arima_no_boost <- arima_reg() %>%
  set_engine(engine = "auto_arima") %>%
  fit(Value ~ date, data = training(splits))
#model_fit_arima_no_boost  ARIMA(5,1,1)(0,0,1)[13]

# 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 = 2,
  learn_rate = 0.015
```

```
) %>%
  set_engine(engine = "auto_arima_xgboost") %>%
  fit(Value ~ date + as.numeric(date) + factor(month(date, label = TRUE), ordere
      data = training(splits))

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

# Model 4: prophet ----
model_fit_prophet <- prophet_reg(seasonality_weekly = TRUE) %>%
  set_engine(engine = "prophet") %>%
  fit(Value ~ date, data = training(splits))
model_fit_prophet
# Model 5: lm ----
model_fit_lm <- linear_reg() %>%
  set_engine("lm") %>%
  fit(Value ~ as.numeric(date) + factor(month(date, label = TRUE), ordered = FAL
      data = training(splits))

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

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

model_spec_mars

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)



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

models_tbl

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

calibration_tbl


# test set forecast and accuracy
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
  )

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

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

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
  )

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

ensemble_fit

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

# get split for in-sample forecast
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()

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)

dev.off()

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

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

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

#```



# HoltWinters
# gives best test mape of individual models
country.hw<- HoltWinters(train.ts)
country.hw

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 seasonal peaks
#- 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,
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="HoltWinter"))
```

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


Acf(na.omit(country.hw.fc$residuals), lag.max = 20)
Box.test(country.hw.fc$residuals, lag=20, type = "Ljung-Box")
checkresiduals(country.hw)

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

# tslm gives best training mape, but bad test mape
#https://www.rdocumentation.org/packages/forecast/versions/8.21/topics/tslm

# trend only
tslm(train.ts~ trend, lambda="auto") %>%
  forecast(h=future) %>% accuracy()
#mape=6.419287

# trend^2
tslm(train.ts~ trend+I(trend^2), lambda="auto") %>% forecast(h=future) %>% accur
#mape 6.180604

# season only
tslm(train.ts~ season, lambda="auto") %>% forecast(h=future) %>% accuracy()
# mape 6.299878

# season + trend
tslm(train.ts~ season+trend, lambda="auto") %>% forecast(h=future) %>% accuracy(
# mape 4.373946

tslm(train.ts~ season+trend, lambda=NULL) %>% forecast(h=future) %>% accuracy()
# mape 5.03524

tslm(log(train.ts)~ season+trend, lambda=NULL) %>% forecast(h=future) %>% accura
# mape 0.4258694

# final lm model
country.lm <- tslm(log(train.ts)~ trend + season, lambda=NULL)
country.lm.fc <- forecast(country.lm,h=future)
accuracy(country.lm.fc, valid.ts)
# test mape = 99.9826714

country.lm
summary(country.lm)

acf(na.omit(country.lm.fc$residuals), lag.max = 20)
Box.test(country.lm.fc$residuals, lag=20, type = "Ljung-Box")
ggtsdisplay(country.lm.fc$residuals)
checkresiduals(country.lm)

#
autoplot(country.stl$time.series[,2])  +
  autolayer(exp(country.lm$fitted.values))+  autolayer(exp(country.lm.fc$mean))+
  ggtitle(mtitle, subtitle = paste0(stitle, "-trend")) +
  xlab(x_lab) +   ylab(y_lab)
```

```
#```

#Plot the data again, but this time add a simple exponential smoothing curve to
#predictions 12 periods ahead. Tip: use alpha = 0.05 for a smoother curve

#$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2y_{

#  so for $\alpha=0.94$: $\hat{y}_{T+1|T} = 0.94*y_T + 0.94(0.06)y_{T-1} + 0.94(
#```{r plotSmooth, eval=TRUE}
### TODO check modelvals - gamma, etc

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))

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)
# graph prediction
plot(a.ets.fc, ylab = y_lab,
```

```
     xlab = x_lab, bty = "l", xaxt = "n",
     main = mtitle,
     sub = paste0("Exp Smoothing ", ALPHA), flty = 2)
grid(nx = NULL, ny = NULL, col = "lightgray", lty = "dotted",
     lwd = par("lwd"), equilogs = TRUE)
axis(1, at = seq(startYear,endYear+1, 1), labels = format(seq(startYear,endYear+
lines(a.ets.fc$fitted, lwd = 2, col = "blue")
lines(ets.valid.ts)

a.ets
#output accuracy score
print(accuracy(a.pred, valid.ts))

#```

#ets with no parameters will estimate alpha, beta, gamma, phi (for a_d) & damped
autoplot(country.hw.fc, PI=FALSE) + autolayer(valid.ts) +
 autolayer(country.hw$fitted[,1])

autoplot(country.ts) +
  autolayer(country.hw.fc, PI=F, series="forecast") +
  autolayer(country.hw$fitted[,1], PI=F, series = " fitted")+
  ggtitle(paste0("Mortality in ", country_code, " HoltWinter Forecast"),
          subtitle = stitle) +
  theme_minimal() +
  xlab(x_lab) + ylab(y_lab) +
  guides(colour=guide_legend(title="HoltWinter"))
```

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


acf(na.omit(country.hw.fc$residuals), lag.max = 20)
Box.test(country.hw.fc$residuals, lag=20, type = "Ljung-Box")
checkresiduals(country.hw.fc)

#The correlogram shows that the autocorrelations for the in-sample
#forecast errors do not exceed the significance bounds for lags 1-20.
#Furthermore, the p-value for Ljung-Box test is 0.6, indicating that there
#is little evidence of non-zero autocorrelations at lags 1-20.
ggtsdisplay(country.hw.fc$residuals)
#plotForecastErrors(na.omit(country.ts.fc2$residuals))

head(country.hw)
print(accuracy(country.hw.fc, valid.ts))
#mape 7.50
?HoltWinters

#there is an issue of autocorrelation with the data? How would you know and how

  #```{r acf, echo=TRUE}
# look at autocorrelation
ggAcf(train.ts, lag=20)
ggAcf(log(train.ts), lag=20)

ggtsdisplay(train.ts)
```

```
ggtsdisplay(log(train.ts))

#```


# ans
#The ACF values are all above the threshold.  The ACF of the time series is slow
# decreasing
#The partial ACF values are all inside the threshold except for lag 1 & 2
#You can't predict values for non-stationary data.

#There is a clear seasonal effect.  ``strong autocorrelation at multiples of a l
The positive lag-1 correlation suggests a strong linear trend.
#[@shmueli2016practical]



#For non-stationary data, the value of r1 is often large and positive
#```{r acfManual}
# using log transform, half-yearly and yearly (seasonal) diff to make stationary
p1 <- autoplot(train.ts) + ylab("orig") + xlab("")
#p1
p2 <- country.ts %>% log() %>% autoplot() + ylab("+log") + xlab("")
#p2
p3 <- country.ts %>% log() %>% diff(lag=1) %>%
  autoplot() + ylab("+lag-1") + xlab("")
#p3
p4 <- country.ts %>% log() %>% diff(lag=1) %>%
  diff(lag=52) %>% autoplot() +    xlab(x_lab) +    ylab("+lag-13")
```

```
#p4
p1/p2/p3/p4


#```

#Fit an arima model to the data and report the results of the model (i.e., print
#either choose your own ARIMA parameters, or use the auto.arima function in R. E
#(either yours or the one chosen by auto.arima) in equation form.

#```{r arima, echo=TRUE}
# fitting the raw data into the arima function

##MANUAL
# some combinations throw an error which stops compilation

fitArima <- function (timeseries, o, s) {
  tried <- try((fit<-Arima(country.ts, order = o, seasonal = s)),
               silent = TRUE)
  if(inherits(tried, "try-error")) {
    return(NULL)
  } else {
    return(fit)
  }
}
```

```
tuneArima <- function(timeseries, rv,seas = FALSE) {
  tbl_colnames <- c("a", "b", "c", "d", "e", "f", "aic", "aicc", "me",
                    "rmse", "mae", "mpe", "mape", "mase", "acf1")
  x <- tbl_colnames %>% purrr::map_dfc(~tibble::tibble(!!.x := numeric()))
  if(seas) {
    srv <- rv
  }
  else {
    srv <- 0:0
  }
  for (a in rv) {
    for (b in rv) {
      for (c in rv) {
        #print(paste(a, b,c))
        for (d in srv) {
          for (e in srv) {
            for (f in srv) {
              fit <- fitArima(timeseries, o = c(a, b, c), s= c(d, e, f))
              if (is.null(fit)) next
              acc<- as.data.frame(accuracy(fit))
              x <- x %>% add_row(a = a, b = b, c= c,
                                 d=d, e=e, f=f, aic = fit$aic,
                                 aicc = fit$aicc, me=acc$ME, rmse=acc$RMSE,
                                 mae=acc$MAE, mpe=acc$MPE, mape=acc$MAPE,
                                 mase=acc$MASE, acf1=acc$ACF1
                                 )
```

```
                # add in mape
              }#f
            }#e
          }#d
        }#c
      }#b
    } #a
  return(x)
}
rangeVals <- 0:2
tunes<- tuneArima(country.ts, rangeVals, seas = TRUE)
saveRDS(tunes, file="data/arimaManualFit.rds")
head(tunes)

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

# compare on mape
best <- tunes[tunes$mape==min(tunes$mape),1:6] %>% as.double()
#print(best)
best <- c(1, 0, 2, 2, 2, 2)
print(best[1:3])


#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]

print(forecast::accuracy(country.am.fc, valid.ts))
print(forecast::accuracy(country.aa.fc, valid.ts))
#https://www.educba.com/arima-model-in-r/

#```

  #```{r arimaResid, eval=TRUE, include=TRUE }
ggtsdisplay(resid(country.am))
checkresiduals(country.am)

ggtsdisplay(resid(country.aa))
checkresiduals(country.aa)

#-----------------------------------------------------------------------------
# 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"))
ggsave(here("docs","allModels.png"))



#----------------------------------------------------------------------------
# look at modelling up to but not including covid

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))


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


(fit <- auto.arima(train.ts))
covid.arima <- forecast(fit, covid)
#ARIMA(1,0,0)(1,1,0)[52] with drift
```

```
print(accuracy(covid.arima, covid.ts))
#                    ME        RMSE        MAE         MPE
MAPE       MASE        ACF1
#Training set    28.90457    737.236   538.2044   0.0454478   0.9856819 0.3765921 -0.
#Test set      8543.18139 11008.998 8668.1931 12.3285708 12.5378937 6.0653032
0.94748390
#Theil's U
#Training set          NA
#Test set        5.054504

fit.hw <- HoltWinters(train.ts)
covid.hw <- forecast(fit.hw, covid)

print(accuracy(covid.hw, covid.ts))
#mape
#


tslm(log(train.ts)~ season+trend, lambda=NULL) %>%
  forecast(h=covid) %>% accuracy()
# mae, mpe, mape  569.5378 0.02037614  1.034954
# mae, mpe, mape  7226.7225 9.85287711 10.246378



# final lm model
fit.lm <- tslm(log(train.ts)~ trend + season, lambda=NULL)
covid.lm <- forecast(fit.lm,h=covid)
```

```
accuracy(covid.lm, covid.ts)
# mae, mpe, mape  1.154522e-02 -0.0002987687  0.105619
# mae, mpe, mape  6.471031e+04 99.9828737377 99.982874
summary(fit.lm)
# adj r^2 = 0.8866
# significant trend coefficient  2.719e-04
fit.lm$coefficients[2]

# 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"))
ggsave(here("docs", "noCovid.png"))

#```
#```{r unused, eval=FALSE, include=FALSE}

non_included <- function() {

}
#```
```

$\#`r\ paste(date(),\ "\backslash n")`$

References

[Coghlan, 2023] Coghlan, A. (2023). Welcome to a little book of r for time series!

[Dancho, 2023] Dancho, M. (2023). *modeltime: The Tidymodels Extension for Time Series Modeling*. https://github.com/business-science/modeltime, https://business-science.github.io/modeltime/.

[Hyndman and Athanasopoulos, 2021] Hyndman, R. J. and Athanasopoulos, G. (2021). *Forecasting: principles and practice*. Monash University, 3 edition.

[OECD, 2023] OECD (2023). Covid-19 health indicators, mortality (by week).

[Shmueli and Lichtendahl, 2016] Shmueli, G. and Lichtendahl, K. C. (2016). *Practical Time Series Forecasting with R: A Hands-On Guide [2nd Edition]*. Practical Analytics. Axelrod Schnall Publishers.