Predicting Covid Vaccination Rates

Edbert Jao

12/20/2021

Load required libraries

```
library(fredr);
```

```
## Warning: package 'fredr' was built under R version 4.0.5
```

```
library(vars);
```

```
## Warning: package 'vars' was built under R version 4.0.5
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 4.0.5
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.0.5
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 4.0.5
```

```
## Loading required package: urca
```

```
## Warning: package 'urca' was built under R version 4.0.5
```

```
## Loading required package: lmtest
```

```
## Warning: package 'lmtest' was built under R version 4.0.5
```

```
library(urca);
library(tidyverse);
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x stringr::boundary() masks strucchange::boundary()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x dplyr::select()     masks MASS::select()
```

```
library(aTSA);
```

```
## Warning: package 'aTSA' was built under R version 4.0.3
```

```
##
## Attaching package: 'aTSA'
```

```
## The following object is masked from 'package:vars':
##
##     arch.test
```

```
## The following object is masked from 'package:graphics':
##
##     identify
```

```
library(tsDyn);
```

```
## Warning: package 'tsDyn' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(lmtest);
library(xts);
```

```
## Warning: package 'xts' was built under R version 4.0.4
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
library(lubridate);
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

Load data

```
setwd("C:\\Users\\edber\\OneDrive\\Desktop\\Road to PHD\\Fall 2021 @ Tufts\\Applications of Econometrics\\Final Project\\Predicting");
covid_df = read.csv("owid-covid-data.csv");
```

Initial Data Cleaning

```
covid_df %>% str();
```

```
## 'data.frame':    148800 obs. of  67 variables:
##  $ iso_code                                : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ continent                               : chr  "Asia" "Asia" "Asia" "Asia" ...
##  $ location                                : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##  $ date                                    : chr  "2020-02-24" "2020-02-25" "2020-02-26" "2020-02-27" ...
##  $ total_cases                             : num  5 5 5 5 5 5 5 5 5 ...
##  $ new_cases                               : num  5 0 0 0 0 0 0 0 0 ...
##  $ new_cases_smoothed                      : num  NA NA NA NA NA 0.714 0.714 0 0 0 ...
##  $ total_deaths                            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_deaths                              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_deaths_smoothed                     : num  NA NA NA NA NA 0 0 0 0 0 ...
##  $ total_cases_per_million                 : num  0.126 0.126 0.126 0.126 0.126 0.126 0.126 0.126 0.126 0.126 ...
##  $ new_cases_per_million                   : num  0.126 0 0 0 0 0 0 0 0 0 ...
##  $ new_cases_smoothed_per_million          : num  NA NA NA NA NA 0.018 0.018 0 0 0 ...
##  $ total_deaths_per_million                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_deaths_per_million                  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_deaths_smoothed_per_million         : num  NA NA NA NA NA 0 0 0 0 0 ...
##  $ reproduction_rate                       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ icu_patients                            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ icu_patients_per_million                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hosp_patients                           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hosp_patients_per_million               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ weekly_icu_admissions                   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ weekly_icu_admissions_per_million       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ weekly_hosp_admissions                  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ weekly_hosp_admissions_per_million      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests                               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests                             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_tests_per_thousand                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_per_thousand                  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed                      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_tests_smoothed_per_thousand         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ positive_rate                           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tests_per_case                          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ tests_units                             : chr  "" "" "" "" ...
##  $ total_vaccinations                      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ people_vaccinated                       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ people_fully_vaccinated                 : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_boosters                          : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ new_vaccinations                         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_vaccinations_smoothed                : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_vaccinations_per_hundred           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ people_vaccinated_per_hundred            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ people_fully_vaccinated_per_hundred      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_boosters_per_hundred               : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_vaccinations_smoothed_per_million    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_people_vaccinated_smoothed           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_people_vaccinated_smoothed_per_hundred: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stringency_index                         : num  8.33 8.33 8.33 8.33 8.33 ...
##  $ population                               : num  39835428 39835428 39835428 39835428 39835428 ...
##  $ population_density                       : num  54.4 54.4 54.4 54.4 54.4 ...
##  $ median_age                               : num  18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6 18.6 ...
##  $ aged_65_older                            : num  2.58 2.58 2.58 2.58 2.58 ...
##  $ aged_70_older                            : num  1.34 1.34 1.34 1.34 1.34 ...
##  $ gdp_per_capita                           : num  1804 1804 1804 1804 1804 ...
##  $ extreme_poverty                          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ cardiovasc_death_rate                    : num  597 597 597 597 597 ...
##  $ diabetes_prevalence                      : num  9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59 9.59 ...
##  $ female_smokers                           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ male_smokers                             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ handwashing_facilities                   : num  37.7 37.7 37.7 37.7 37.7 ...
##  $ hospital_beds_per_thousand               : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ life_expectancy                          : num  64.8 64.8 64.8 64.8 64.8 ...
##  $ human_development_index                  : num  0.511 0.511 0.511 0.511 0.511 0.511 0.511 0.511 0.511 0.511 ...
##  $ excess_mortality_cumulative_absolute     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ excess_mortality_cumulative              : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ excess_mortality                         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ excess_mortality_cumulative_per_million  : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
covid_df = covid_df %>% dplyr::filter(iso_code == "USA");

df = select(covid_df, date, people_fully_vaccinated_per_hundred, new_vaccinations_smoothed, new_deaths_per_million);
df %>% str();
```

```
## 'data.frame':    699 obs. of  4 variables:
##  $ date                               : chr  "2020-01-22" "2020-01-23" "2020-01-24" "2020-01-25" ...
##  $ people_fully_vaccinated_per_hundred: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_vaccinations_smoothed          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ new_deaths_per_million             : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
date = df$date[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
vacc_rate = df$people_fully_vaccinated_per_hundred[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
ln_newvacc = log(df$new_vaccinations_smoothed)[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
new_deaths_per_million = df$new_deaths_per_million[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
dft_reference = cbind(vacc_rate, new_deaths_per_million, ln_newvacc) %>% as.data.frame(); # creating a dataframe to referenc
e for in-sample predictions
```

Stage 1: Predicting in-sample for the past 30 days, using all days for which data is available from 31 days ago and prior.

Start by specifying the model. The 30 most recent days' data are removed. They will be predicted for. Comparing the predicted values and the in-sample values will show the model's effectiveness.

```
vacc_rate = df$people_fully_vaccinated_per_hundred[328:(length(df$people_fully_vaccinated_per_hundred)-30)] %>% ts();
ln_newvacc = log(df$new_vaccinations_smoothed)[328:(length(df$people_fully_vaccinated_per_hundred)-30)] %>% ts();
new_deaths_per_million = df$new_deaths_per_million[328:(length(df$people_fully_vaccinated_per_hundred)-30)] %>% ts();
dft = cbind(vacc_rate, new_deaths_per_million, ln_newvacc) %>% as.data.frame();

grangertest(ln_newvacc~vacc_rate);             # ln_newvacc granger-causes vacc_rate
```

```
## Granger causality test
##
## Model 1: ln_newvacc ~ Lags(ln_newvacc, 1:1) + Lags(vacc_rate, 1:1)
## Model 2: ln_newvacc ~ Lags(ln_newvacc, 1:1)
##   Res.Df Df      F   Pr(>F)
## 1    338
## 2    339 -1 48.619 1.64e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(new_deaths_per_million~vacc_rate); # new_deaths_per_million granger-causes vacc_rate
```

```
## Granger causality test
##
## Model 1: new_deaths_per_million ~ Lags(new_deaths_per_million, 1:1) + Lags(vacc_rate, 1:1)
## Model 2: new_deaths_per_million ~ Lags(new_deaths_per_million, 1:1)
##   Res.Df Df      F    Pr(>F)
## 1    338
## 2    339 -1 12.627 0.0004342 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tseries::adf.test(vacc_rate);                 #stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  vacc_rate
## Dickey-Fuller = -2.7675, Lag order = 6, p-value = 0.2529
## alternative hypothesis: stationary
```

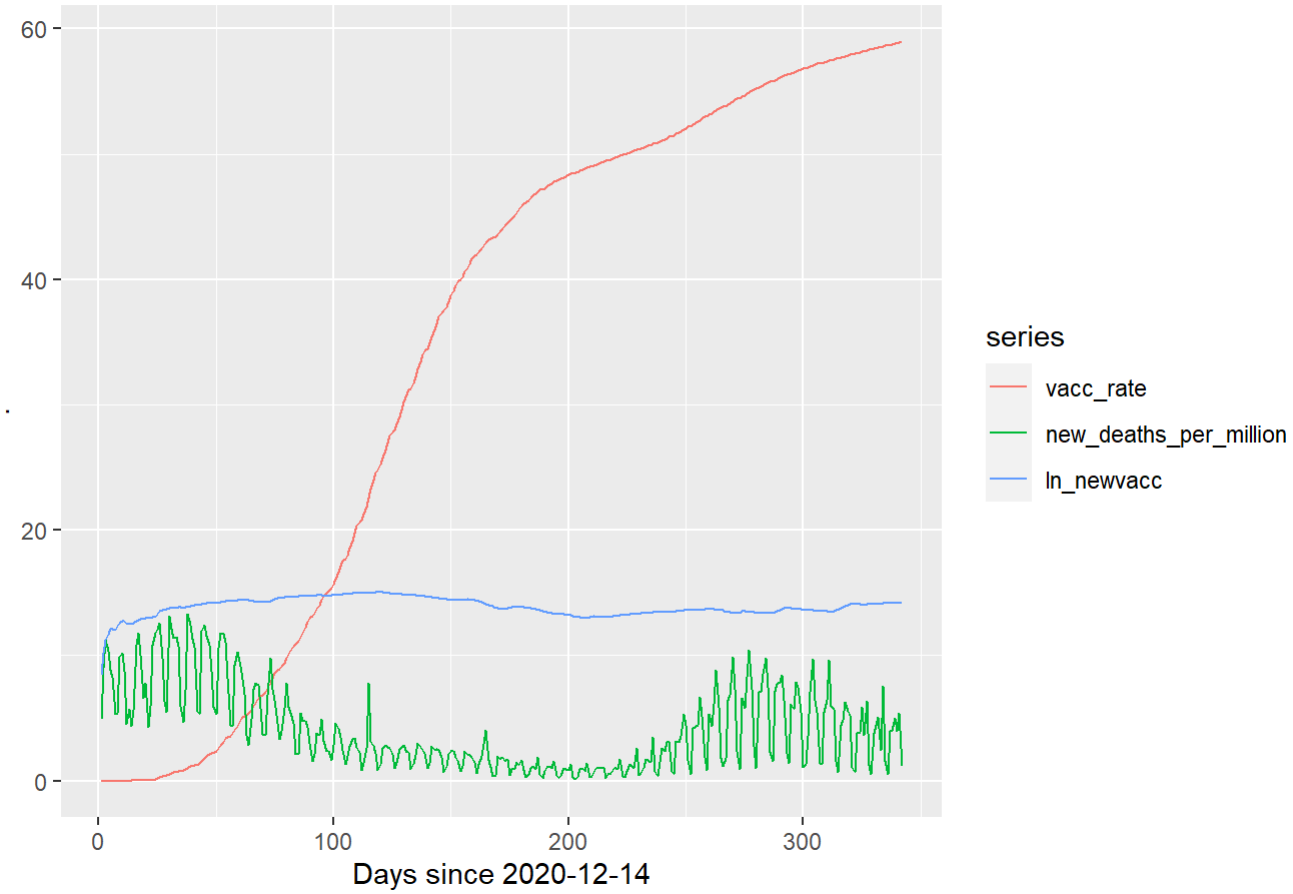
```
tseries::adf.test(ln_newvacc);                #non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ln_newvacc
## Dickey-Fuller = -1.8571, Lag order = 6, p-value = 0.637
## alternative hypothesis: stationary
```

```
tseries::adf.test(new_deaths_per_million);    #non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  new_deaths_per_million
## Dickey-Fuller = -0.87075, Lag order = 6, p-value = 0.9547
## alternative hypothesis: stationary
```

```
cbind(vacc_rate, new_deaths_per_million, ln_newvacc) %>% autoplot() + xlab("Days since 2020-12-14");
```



```
lagselect = VARselect(dft);
lagselect;                                  # AIC recommends 10 lags
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10     10     10     10
##
## $criteria
##                   1             2             3             4             5
## AIC(n) -9.8266984864 -1.106098e+01 -1.120662e+01 -1.123762e+01 -1.128648e+01
## HQ(n)  -9.7718496046 -1.096499e+01 -1.106949e+01 -1.105936e+01 -1.106708e+01
## SC(n)  -9.6891634875 -1.082029e+01 -1.086278e+01 -1.079063e+01 -1.073634e+01
## FPE(n)  0.0000539909  1.571395e-05  1.358478e-05  1.317097e-05  1.254419e-05
##                   6             7             8             9            10
## AIC(n) -1.171807e+01 -1.277167e+01 -1.342790e+01 -1.413180e+01 -1.423730e+01
## HQ(n)  -1.145754e+01 -1.247000e+01 -1.308509e+01 -1.374786e+01 -1.381222e+01
## SC(n)  -1.106478e+01 -1.201523e+01 -1.256830e+01 -1.316906e+01 -1.317140e+01
## FPE(n)  8.148320e-06  2.841757e-06  1.474722e-06  7.297254e-07  6.569493e-07
```

```
ctest1e = ca.jo(dft, type = "eigen", K = 10);  # Johansen procedure: testing for and estimating cointegrating systems
ctest1e %>% summary();                         # strong evidence for 2 cointegrating relationship: VECM is viable
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.07472631 0.04841692 0.01205709
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  4.03  6.50  8.18 11.65
## r <= 1 | 16.48 12.91 14.90 19.19
## r = 0  | 25.79 18.90 21.07 25.75
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                            vacc_rate.l10 new_deaths_per_million.l10
## vacc_rate.l10                    1.00000                   1.000000
## new_deaths_per_million.l10       4.90245                  -4.172518
## ln_newvacc.l10                 -81.46156                  41.638819
##                            ln_newvacc.l10
## vacc_rate.l10                 1.00000000
## new_deaths_per_million.l10   10.51791165
## ln_newvacc.l10                0.08814658
##
## Weights W:
## (This is the loading matrix)
##
##                          vacc_rate.l10 new_deaths_per_million.l10
## vacc_rate.d              -2.795101e-04              -0.0001349817
## new_deaths_per_million.d  4.005519e-03              -0.0011575221
## ln_newvacc.d             -9.210099e-06              -0.0002319830
##                          ln_newvacc.l10
## vacc_rate.d               -8.098283e-05
```

```
## new_deaths_per_million.d  -5.700867e-03
## ln_newvacc.d              -3.485494e-05
```

```
# all eigenvalues lie within the unit circle
```

Diagnostic tests and plots for the model.

```
vecm1 = VECM(dft, 10, r = 2, estim = "ML");
vecm1 %>% summary();
```

```
## Warning in if (class(x) == "numeric") return(noquote(r)): the condition has
## length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "matrix") return(matrix(noquote(r), ncol = ncol(x), :
## the condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "numeric") return(noquote(r)): the condition has
## length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "matrix") return(matrix(noquote(r), ncol = ncol(x), :
## the condition has length > 1 and only the first element will be used
```

```
## #############
## ###Model VECM
## #############
## Full sample size: 342    End sample size: 331
## Number of variables: 3   Number of estimated slope parameters 99
## AIC -4710.367    BIC -4326.353   SSR 323.2995
## Cointegrating vector (estimated by ML):
##        vacc_rate new_deaths_per_million ln_newvacc
## r1  1.000000e+00                     0  -20.28973
## r2 -2.775558e-17                     1  -21.15123
##
##
##                                ECT1                ECT2
## Equation vacc_rate             -0.0004(0.0001)***  -0.0002(0.0003)
## Equation new_deaths_per_million 0.0023(0.0033)      0.0240(0.0105)*
## Equation ln_newvacc            -0.0002(8.7e-05)**  0.0008(0.0003)**
##                                Intercept           vacc_rate -1
## Equation vacc_rate             -0.1346(0.0899)     0.7949(0.0702)***
## Equation new_deaths_per_million 7.2652(2.9061)*     3.2127(2.2700)
## Equation ln_newvacc            0.1737(0.0755)*     -0.0215(0.0590)
##                                new_deaths_per_million -1 ln_newvacc -1
## Equation vacc_rate             -0.0021(0.0019)           -0.1500(0.0858).
## Equation new_deaths_per_million -0.7881(0.0618)***       2.4452(2.7736)
## Equation ln_newvacc            -0.0053(0.0016)***        0.6707(0.0721)***
##                                vacc_rate -2        new_deaths_per_million -2
## Equation vacc_rate             0.0671(0.0885)      -0.0002(0.0023)
## Equation new_deaths_per_million -0.3536(2.8606)     -0.7332(0.0756)***
## Equation ln_newvacc            0.1038(0.0743)      -0.0046(0.0020)*
##                                ln_newvacc -2       vacc_rate -3
## Equation vacc_rate             0.0695(0.1024)      -0.1152(0.0725)
## Equation new_deaths_per_million 2.9357(3.3125)      -2.4546(2.3439)
## Equation ln_newvacc            -0.0002(0.0861)     -0.0327(0.0609)
##                                new_deaths_per_million -3 ln_newvacc -3
## Equation vacc_rate             0.0037(0.0026)            -0.1061(0.0921)
## Equation new_deaths_per_million -0.6098(0.0853)***       -4.3309(2.9802)
## Equation ln_newvacc            -0.0041(0.0022).          -0.0504(0.0775)
##                                vacc_rate -4       new_deaths_per_million -4
## Equation vacc_rate             0.0194(0.0396)     -0.0011(0.0028)
## Equation new_deaths_per_million 0.5687(1.2793)     -0.5942(0.0908)***
```

```
## Equation ln_newvacc             0.0306(0.0332)     -0.0058(0.0024)*
##                                 ln_newvacc -4       vacc_rate -5
## Equation vacc_rate              0.0798(0.0782)      -0.0602(0.0395)
## Equation new_deaths_per_million -1.6621(2.5305)     -0.5400(1.2763)
## Equation ln_newvacc             -0.0788(0.0658)     -0.0360(0.0332)
##                                 new_deaths_per_million -5 ln_newvacc -5
## Equation vacc_rate              -0.0001(0.0029)           -0.0493(0.0648)
## Equation new_deaths_per_million -0.6053(0.0925)***        1.5241(2.0946)
## Equation ln_newvacc             -0.0059(0.0024)*          0.1471(0.0544)**
##                                 vacc_rate -6       new_deaths_per_million -6
## Equation vacc_rate              0.0455(0.0396)     0.0008(0.0028)
## Equation new_deaths_per_million 0.6584(1.2794)     -0.4349(0.0919)***
## Equation ln_newvacc             0.0337(0.0333)     -0.0043(0.0024).
##                                 ln_newvacc -6      vacc_rate -7
## Equation vacc_rate              0.0310(0.0659)     0.8448(0.0396)***
## Equation new_deaths_per_million 0.5951(2.1314)     0.0052(1.2796)
## Equation ln_newvacc             0.0099(0.0554)     -0.0798(0.0333)*
##                                 new_deaths_per_million -7 ln_newvacc -7
## Equation vacc_rate              0.0006(0.0028)            -0.1776(0.0626)**
## Equation new_deaths_per_million 0.2656(0.0892)**          -3.9046(2.0234).
## Equation ln_newvacc             -0.0041(0.0023).          -0.1896(0.0526)***
##                                 vacc_rate -8        new_deaths_per_million -8
## Equation vacc_rate              -0.6756(0.0744)***  0.0021(0.0026)
## Equation new_deaths_per_million -2.3180(2.4050)     0.2070(0.0839)*
## Equation ln_newvacc             0.0850(0.0625)      -0.0007(0.0022)
##                                 ln_newvacc -8      vacc_rate -9
## Equation vacc_rate              0.1150(0.0618).    -0.1282(0.0890)
## Equation new_deaths_per_million 3.1362(2.0001)     0.0481(2.8790)
## Equation ln_newvacc             0.1612(0.0520)**   -0.1229(0.0748)
##                                 new_deaths_per_million -9 ln_newvacc -9
## Equation vacc_rate              0.0009(0.0023)            0.0070(0.0618)
## Equation new_deaths_per_million 0.1071(0.0741)            2.5411(1.9988)
## Equation ln_newvacc             -2.9e-05(0.0019)          -0.0454(0.0520)
##                                 vacc_rate -10      new_deaths_per_million -10
## Equation vacc_rate              0.1211(0.0684).    -0.0040(0.0019)*
## Equation new_deaths_per_million 2.5821(2.2121)     0.0354(0.0612)
## Equation ln_newvacc             0.0351(0.0575)     -0.0013(0.0016)
##                                 ln_newvacc -10
## Equation vacc_rate              -0.0272(0.0334)
```

```
## Equation new_deaths_per_million -2.8332(1.0798)**
## Equation ln_newvacc             -0.0437(0.0281)
```

```
model1var <- vec2var(ctest1e, r = 2);
serial.test(model1var, type = "BG");          # strong evidence of serial correlation
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object model1var
## Chi-squared = 109.19, df = 45, p-value = 2.957e-07
```

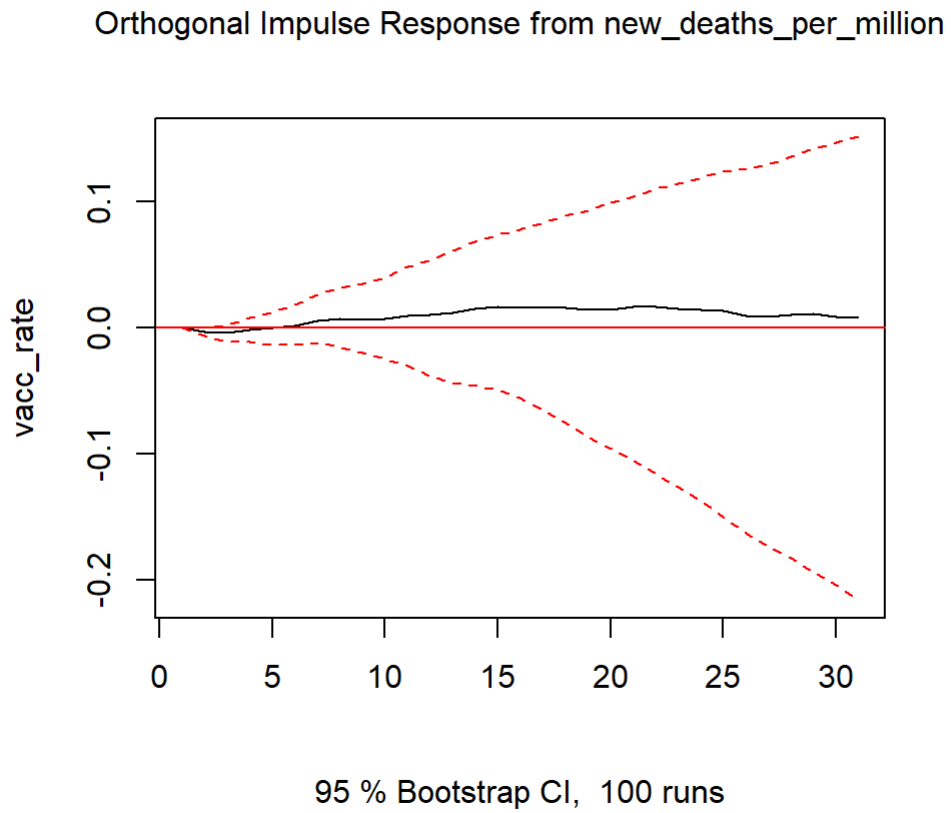
```
vars::arch.test(model1var);                   # strong evidence of heteroskedastic residuals
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object model1var
## Chi-squared = 407.88, df = 180, p-value < 2.2e-16
```

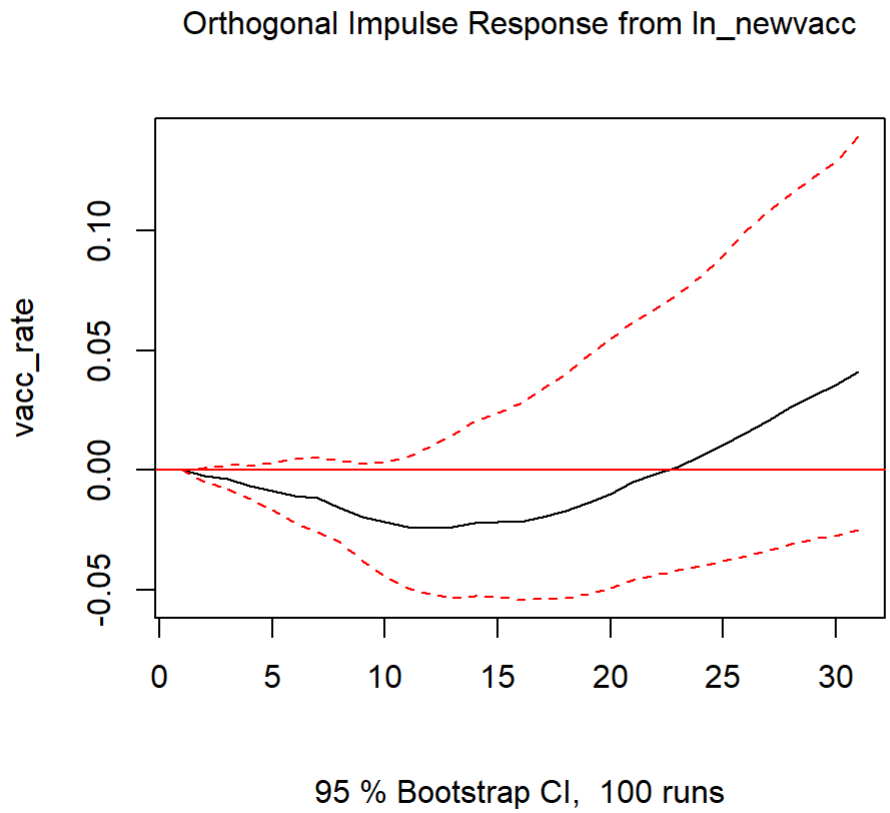
```
normality.test(model1var);                    # strong evidence residuals are not normally distributed
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object model1var
## Chi-squared = 1262.4, df = 6, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object model1var
## Chi-squared = 76.468, df = 3, p-value = 2.22e-16
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object model1var
## Chi-squared = 1185.9, df = 3, p-value < 2.2e-16
```

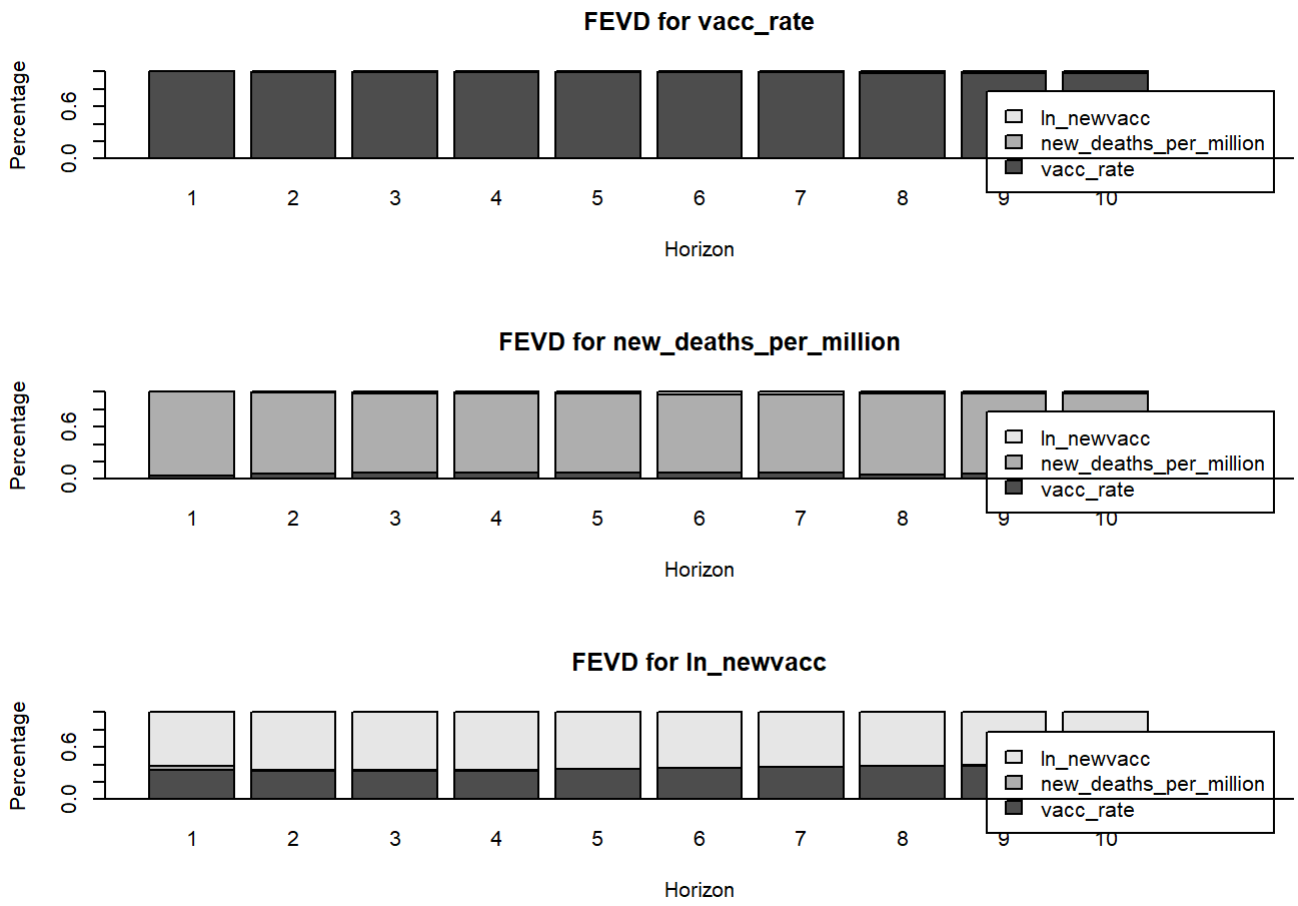
```
plot(irf(model1var, impulse = "new_deaths_per_million", response = "vacc_rate", n.ahead = 30, boot = TRUE));
```




```
plot(irf(model1var, impulse = "ln_newvacc", response = "vacc_rate", n.ahead = 30, boot = TRUE));
```

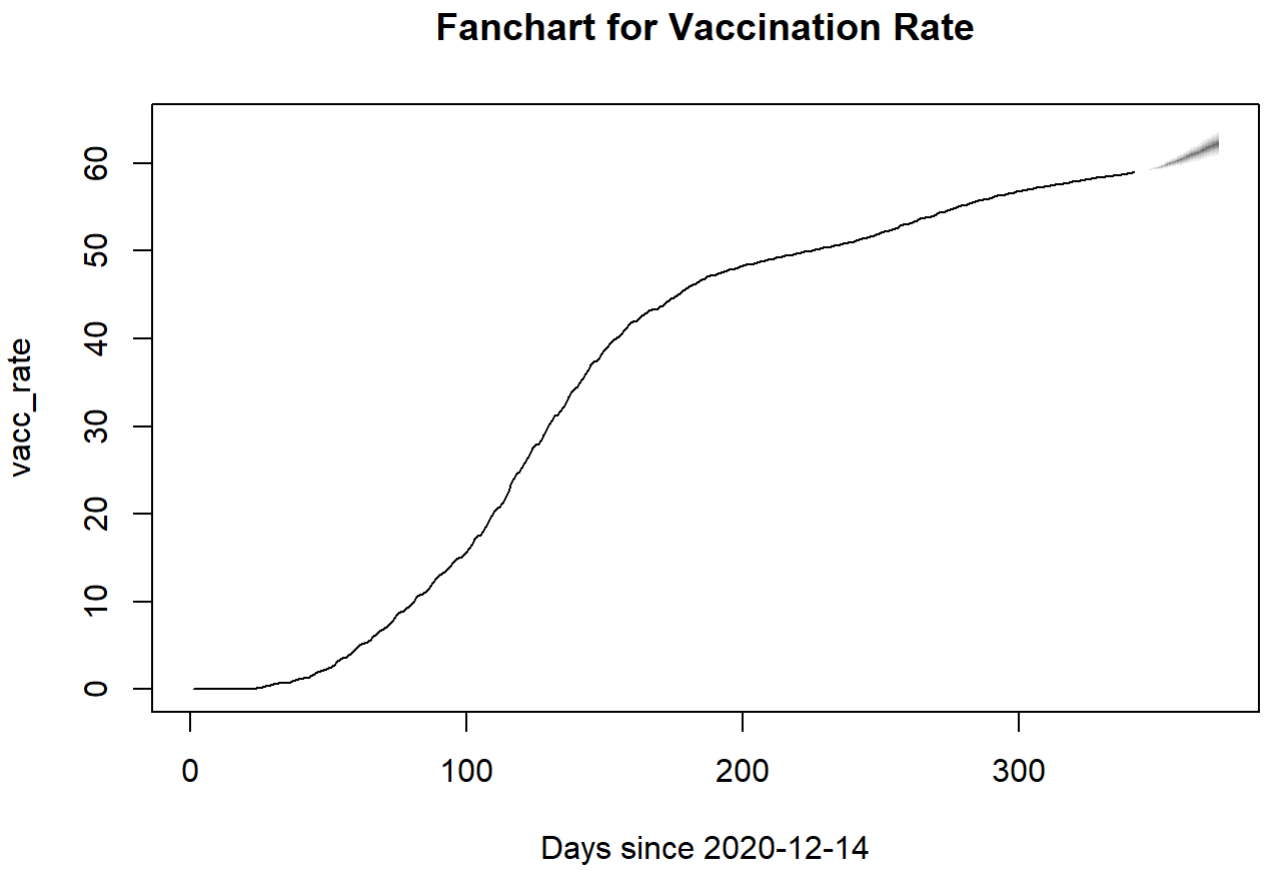



```
plot(fevd(model1var));
```



Forecast the last 30 days vaccination rate using the model. Then calculate in sample Mean Squared Prediction Error

```
forecast1 <- predict(model1var, n.ahead = 30, ci = 0.95);
fanchart(forecast1, names = "vacc_rate", main = "Fanchart for Vaccination Rate", xlab = "Days since 2020-12-14", ylab = "vac
c_rate");
```




```
forecast1$fcst$vacc_rate[,1];
```

```
##  [1] 58.98547 59.04024 59.11171 59.19748 59.29156 59.37325 59.43533 59.50438
##  [9] 59.59458 59.70309 59.83055 59.96434 60.07859 60.17146 60.26946 60.38573
## [17] 60.52056 60.67544 60.83316 60.96600 61.07447 61.18629 61.31444 61.46125
## [25] 61.62911 61.79742 61.93696 62.05031 62.16545 62.29577
```

```
in_sampleMSPE = (1/30) * sum((forecast1$fcst$vacc_rate[,1]%>%as.matrix()) - (dft_reference$vacc_rate[343:372]%>%as.matrix
()))^2;
in_sampleMSPE;
```

```
## [1] 5.354198
```

Stage 2: Predicting out-of-sample, for the next 30 days. Diagnostic tests return the same conclusions even after returning 30 days to the sample.

```
vacc_rate2 = df$people_fully_vaccinated_per_hundred[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
ln_newvacc2 = log(df$new_vaccinations_smoothed)[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
new_deaths_per_million2 = df$new_deaths_per_million[328:(length(df$people_fully_vaccinated_per_hundred))] %>% ts();
dft2 = cbind(vacc_rate2, new_deaths_per_million2, ln_newvacc2) %>% as.data.frame();

grangertest(ln_newvacc2~vacc_rate2);             # ln_newvacc granger-causes vacc_rate
```

```
## Granger causality test
##
## Model 1: ln_newvacc2 ~ Lags(ln_newvacc2, 1:1) + Lags(vacc_rate2, 1:1)
## Model 2: ln_newvacc2 ~ Lags(ln_newvacc2, 1:1)
##   Res.Df Df      F    Pr(>F)
## 1    368
## 2    369 -1 46.073 4.576e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(new_deaths_per_million2~vacc_rate2); # new_deaths_per_million granger-causes vacc_rate
```

```
## Granger causality test
##
## Model 1: new_deaths_per_million2 ~ Lags(new_deaths_per_million2, 1:1) + Lags(vacc_rate2, 1:1)
## Model 2: new_deaths_per_million2 ~ Lags(new_deaths_per_million2, 1:1)
##   Res.Df Df      F    Pr(>F)
## 1    368
## 2    369 -1 13.024 0.0003502 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tseries::adf.test(vacc_rate2);                  #stationary
```

```
## Warning in tseries::adf.test(vacc_rate2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  vacc_rate2
## Dickey-Fuller = -5.4982, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
tseries::adf.test(ln_newvacc2);                 #non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ln_newvacc2
## Dickey-Fuller = -2.7055, Lag order = 7, p-value = 0.2792
## alternative hypothesis: stationary
```

```
tseries::adf.test(new_deaths_per_million2);     #non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  new_deaths_per_million2
## Dickey-Fuller = -1.187, Lag order = 7, p-value = 0.9077
## alternative hypothesis: stationary
```

```
lagselect2 = VARselect(dft2);
lagselect2;                                     # AIC recommends 10 lags
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10     10      9     10
##
## $criteria
##                   1             2             3             4             5
## AIC(n) -9.510610698 -1.087325e+01 -1.100638e+01 -1.103978e+01 -1.109166e+01
## HQ(n)  -9.459326602 -1.078350e+01 -1.087817e+01 -1.087311e+01 -1.088653e+01
## SC(n)  -9.381605918 -1.064749e+01 -1.068387e+01 -1.062052e+01 -1.057564e+01
## FPE(n)  0.000074062  1.895899e-05  1.659618e-05  1.605181e-05  1.524148e-05
##                   6             7             8             9            10
## AIC(n) -1.142728e+01 -1.246735e+01 -1.309212e+01 -1.379346e+01 -1.387790e+01
## HQ(n)  -1.118368e+01 -1.218529e+01 -1.277160e+01 -1.343447e+01 -1.348044e+01
## SC(n)  -1.081451e+01 -1.175783e+01 -1.228584e+01 -1.289043e+01 -1.287811e+01
## FPE(n)  1.089737e-05  3.852052e-06  2.062770e-06  1.023245e-06  9.407069e-07
```

```
ctest2e = ca.jo(dft2, type = "eigen", K = 10);  # Johansen procedure: testing for and estimating cointegrating systems
ctest2e %>% summary();                          # strong evidence for 2 cointegrating relationship: VECM is viable
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.06317728 0.05327829 0.01028005
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  3.74  6.50  8.18 11.65
## r <= 1 | 19.82 12.91 14.90 19.19
## r = 0  | 23.62 18.90 21.07 25.75
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                             vacc_rate2.l10 new_deaths_per_million2.l10
## vacc_rate2.l10                   1.000000                   1.0000000
## new_deaths_per_million2.l10      8.022076                  -0.2848754
## ln_newvacc2.l10               -104.684514                   5.4464120
##                             ln_newvacc2.l10
## vacc_rate2.l10                      1.00000
## new_deaths_per_million2.l10        12.64119
## ln_newvacc2.l10                    -1.77976
##
## Weights W:
## (This is the loading matrix)
##
##                           vacc_rate2.l10 new_deaths_per_million2.l10
## vacc_rate2.d                -0.000137736               -0.0003304693
## new_deaths_per_million2.d    0.002399574               -0.0001967541
## ln_newvacc2.d                0.000094956               -0.0003545435
##                           ln_newvacc2.l10
## vacc_rate2.d                -4.680281e-05
```

```
## new_deaths_per_million2.d   -5.161020e-03
## ln_newvacc2.d               -3.184292e-05
```

```
                                        # all eigenvalues lie within the unit circle

vecm2 = VECM(dft2, 10, r = 2, estim = "ML");
vecm2 %>% summary();
```

```
## Warning in if (class(x) == "numeric") return(noquote(r)): the condition has
## length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "matrix") return(matrix(noquote(r), ncol = ncol(x), :
## the condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "numeric") return(noquote(r)): the condition has
## length > 1 and only the first element will be used
```

```
## Warning in if (class(x) == "matrix") return(matrix(noquote(r), ncol = ncol(x), :
## the condition has length > 1 and only the first element will be used
```

```
## #############
## ###Model VECM
## #############
## Full sample size: 372    End sample size: 361
## Number of variables: 3   Number of estimated slope parameters 99
## AIC -4997.878    BIC -4605.102   SSR 426.6115
## Cointegrating vector (estimated by ML):
##       vacc_rate2 new_deaths_per_million2 ln_newvacc2
## r1  1.000000e+00                       0   -2.667861
## r2 -1.387779e-17                       1  -15.957674
##
##
##                                  ECT1               ECT2
## Equation vacc_rate2              -0.0004(0.0001)*** -0.0006(0.0004)
## Equation new_deaths_per_million2 0.0027(0.0039)     0.0196(0.0129)
## Equation ln_newvacc2             -0.0002(0.0001)*   0.0009(0.0003)**
##                                  Intercept           vacc_rate2 -1
## Equation vacc_rate2              -0.1094(0.0755)     0.7646(0.0665)***
## Equation new_deaths_per_million2 4.1850(2.6916)      1.3897(2.3683)
## Equation ln_newvacc2             0.1971(0.0714)**    -0.0492(0.0628)
##                                  new_deaths_per_million2 -1 ln_newvacc2 -1
## Equation vacc_rate2              -0.0009(0.0017)            -0.0910(0.0761)
## Equation new_deaths_per_million2 -0.8169(0.0607)***         8.5797(2.7122)**
## Equation ln_newvacc2             -0.0052(0.0016)**          0.7709(0.0719)***
##                                  vacc_rate2 -2      new_deaths_per_million2 -2
## Equation vacc_rate2              0.0759(0.0838)     0.0014(0.0021)
## Equation new_deaths_per_million2 0.2263(2.9863)     -0.7520(0.0750)***
## Equation ln_newvacc2             0.1419(0.0792).    -0.0044(0.0020)*
##                                  ln_newvacc2 -2      vacc_rate2 -3
## Equation vacc_rate2              0.0395(0.0929)      -0.0842(0.0692)
## Equation new_deaths_per_million2 0.7056(3.3106)      -1.0502(2.4666)
## Equation ln_newvacc2             -0.0641(0.0878)     -0.0106(0.0654)
##                                  new_deaths_per_million2 -3 ln_newvacc2 -3
## Equation vacc_rate2              0.0025(0.0024)             -0.1075(0.0840)
## Equation new_deaths_per_million2 -0.5862(0.0854)***         -8.5305(2.9949)**
## Equation ln_newvacc2             -0.0042(0.0023).           -0.0774(0.0794)
##                                  vacc_rate2 -4      new_deaths_per_million2 -4
## Equation vacc_rate2              0.0147(0.0386)     -0.0004(0.0026)
## Equation new_deaths_per_million2 0.9612(1.3739)     -0.6079(0.0912)***
```

```
## Equation ln_newvacc2             0.0237(0.0364)     -0.0055(0.0024)*
##                                  ln_newvacc2 -4      vacc_rate2 -5
## Equation vacc_rate2              0.0526(0.0730)      -0.0588(0.0384)
## Equation new_deaths_per_million2 -1.9555(2.6029)     -1.3408(1.3687)
## Equation ln_newvacc2             -0.0413(0.0690)     -0.0289(0.0363)
##                                  new_deaths_per_million2 -5 ln_newvacc2 -5
## Equation vacc_rate2              0.0005(0.0026)              -0.0359(0.0607)
## Equation new_deaths_per_million2 -0.5723(0.0928)***          2.1121(2.1631)
## Equation ln_newvacc2             -0.0057(0.0025)*            0.1258(0.0574)*
##                                  vacc_rate2 -6      new_deaths_per_million2 -6
## Equation vacc_rate2              0.0530(0.0385)     0.0008(0.0026)
## Equation new_deaths_per_million2 0.6368(1.3717)     -0.4207(0.0919)***
## Equation ln_newvacc2             0.0377(0.0364)     -0.0044(0.0024).
##                                  ln_newvacc2 -6      vacc_rate2 -7
## Equation vacc_rate2              0.0260(0.0611)      0.8339(0.0385)***
## Equation new_deaths_per_million2 -0.4835(2.1763)     0.1879(1.3734)
## Equation ln_newvacc2             -0.0273(0.0577)     -0.0920(0.0364)*
##                                  new_deaths_per_million2 -7 ln_newvacc2 -7
## Equation vacc_rate2              0.0013(0.0025)              -0.1731(0.0579)**
## Equation new_deaths_per_million2 0.2666(0.0885)**            -8.4957(2.0624)***
## Equation ln_newvacc2             -0.0040(0.0023).            -0.2140(0.0547)***
##                                  vacc_rate2 -8       new_deaths_per_million2 -8
## Equation vacc_rate2              -0.6335(0.0708)***  0.0014(0.0023)
## Equation new_deaths_per_million2 -0.8960(2.5227)     0.2513(0.0832)**
## Equation ln_newvacc2             0.1328(0.0669)*     -0.0009(0.0022)
##                                  ln_newvacc2 -8     vacc_rate2 -9
## Equation vacc_rate2              0.0943(0.0587)     -0.1466(0.0847).
## Equation new_deaths_per_million2 5.7444(2.0922)**   -0.2818(3.0197)
## Equation ln_newvacc2             0.1826(0.0555)**   -0.1748(0.0801)*
##                                  new_deaths_per_million2 -9 ln_newvacc2 -9
## Equation vacc_rate2              0.0002(0.0020)              0.0183(0.0591)
## Equation new_deaths_per_million2 0.1272(0.0713).             2.1074(2.1051)
## Equation ln_newvacc2             0.0001(0.0019)              -0.0532(0.0558)
##                                  vacc_rate2 -10     new_deaths_per_million2 -10
## Equation vacc_rate2              0.0995(0.0653)     -0.0024(0.0016)
## Equation new_deaths_per_million2 0.5803(2.3280)     0.0265(0.0564)
## Equation ln_newvacc2             0.0235(0.0618)     -0.0014(0.0015)
##                                  ln_newvacc2 -10
## Equation vacc_rate2              -0.0188(0.0314)
```

```
## Equation new_deaths_per_million2 -1.5942(1.1173)
## Equation ln_newvacc2             -0.0328(0.0296)
```

```
model2var <- vec2var(ctest2e, r = 2);
serial.test(model2var, type = "BG");          # strong evidence of serial correlation
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object model2var
## Chi-squared = 91.074, df = 45, p-value = 5.819e-05
```

```
vars::arch.test(model2var);                   # strong evidence of heteroskedastic residuals
```

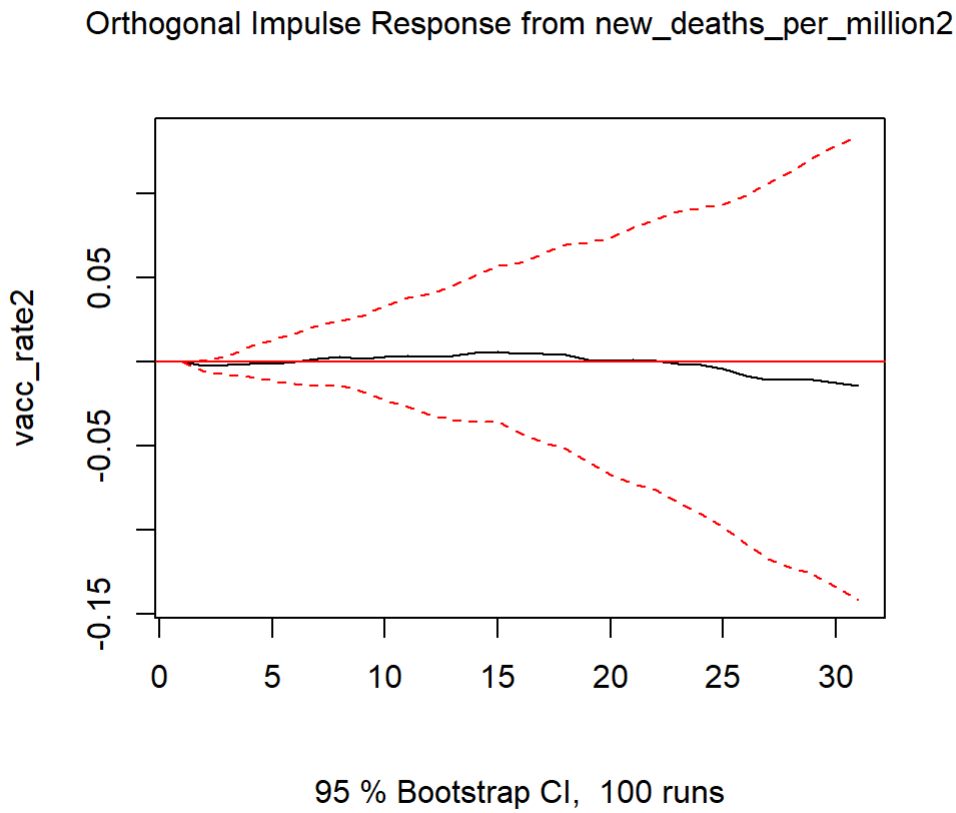
```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object model2var
## Chi-squared = 330.28, df = 180, p-value = 6.056e-11
```

```
normality.test(model2var);                    # strong evidence residuals are not normally distributed
```

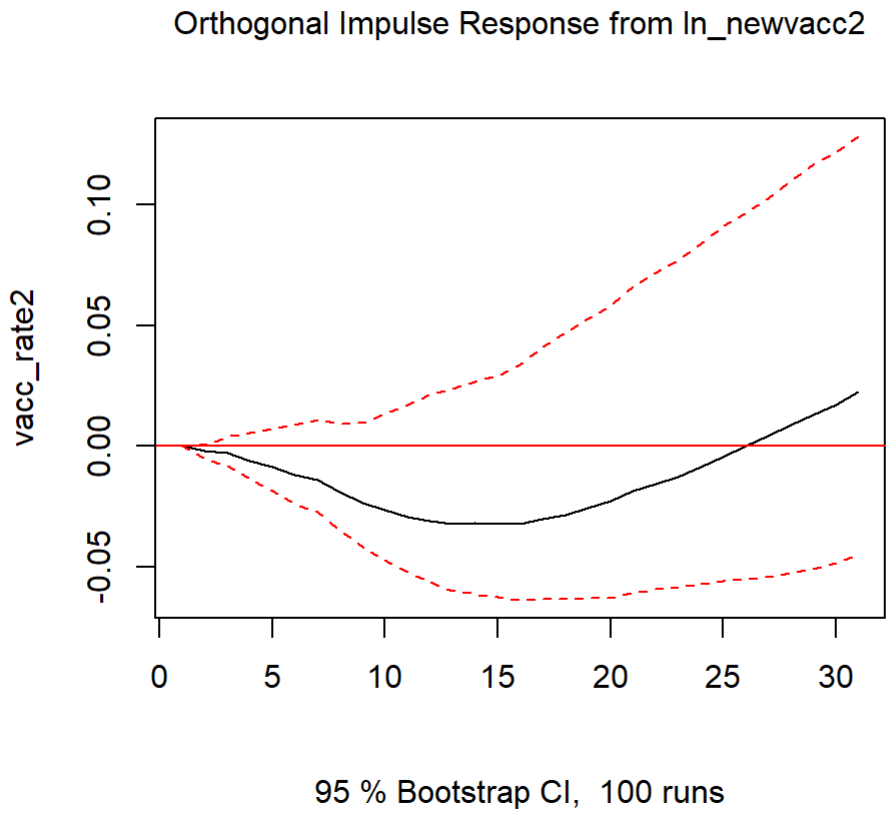
```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object model2var
## Chi-squared = 1622.3, df = 6, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object model2var
## Chi-squared = 48.177, df = 3, p-value = 1.953e-10
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object model2var
## Chi-squared = 1574.1, df = 3, p-value < 2.2e-16
```

Stage 2 Diagnostic Plots highly resemble Stage 1 Diagnostic Plots

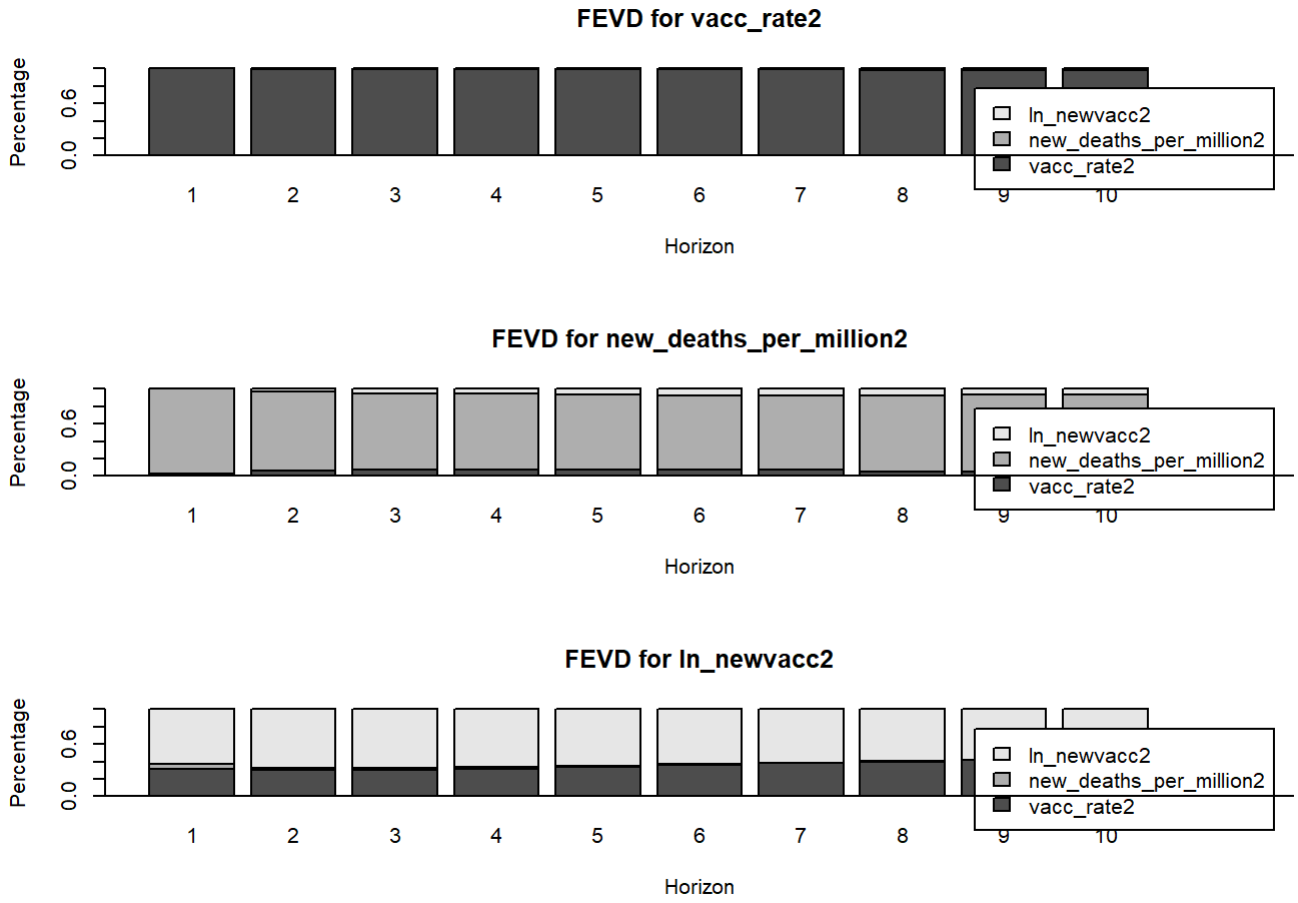
```
plot(irf(model2var, impulse = "new_deaths_per_million2", response = "vacc_rate2", n.ahead = 30, boot = TRUE));
```




```
plot(irf(model2var, impulse = "ln_newvacc2", response = "vacc_rate2", n.ahead = 30, boot = TRUE));
```

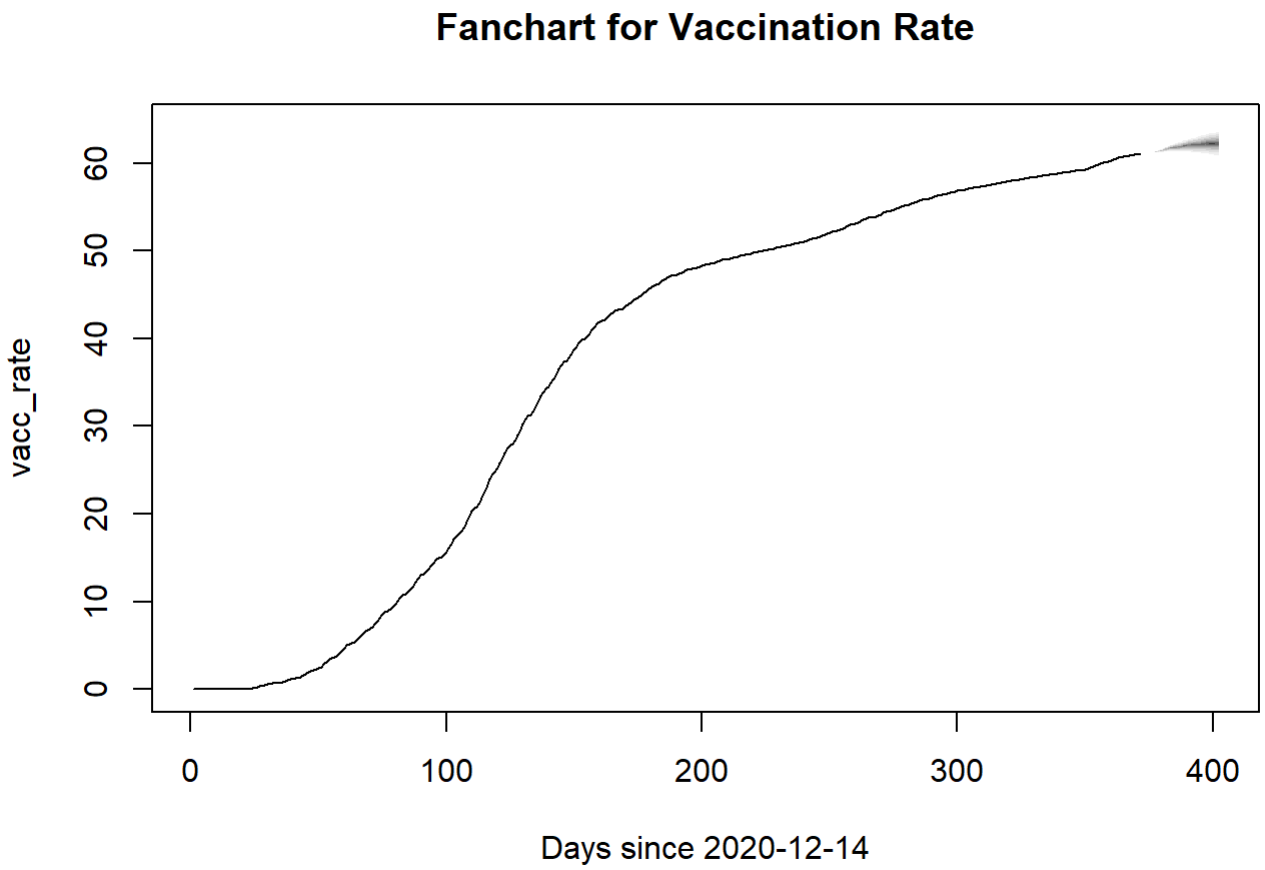



```
plot(fevd(model2var));
```



Forecast vaccination rate for the next 30 days.

```
forecast2 <- predict(model2var, n.ahead = 30, ci = 0.95);
fanchart(forecast2, names = "vacc_rate2", main = "Fanchart for Vaccination Rate", xlab = "Days since 2020-12-14", ylab = "va
cc_rate");
```




```
forecast2$fcst$vacc_rate2[,1];
```

```
##  [1] 61.04292 61.09892 61.15753 61.22740 61.28975 61.32338 61.38008 61.46426
##  [9] 61.55965 61.63755 61.71074 61.76560 61.78720 61.81791 61.87563 61.94092
## [17] 61.98913 62.02860 62.05117 62.04586 62.05329 62.08994 62.14005 62.17515
## [25] 62.20125 62.21158 62.19747 62.19558 62.22378 62.26570
```