homework_6

Harinath Reddy

2022-11-24

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(tsibble)
```

```
##
## Attaching package: 'tsibble'

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

```
library(fable)
```

```
## Loading required package: fabletools
```

```
library(feasts)
library(tidyr)
library(ggplot2)
```

```
library(IDDA)
data(state.ts)
```

```
## Using `DATE` as index variable.
Virginia.ts <- state.ts %>%
  dplyr::filter(State == "Virginia") %>%
  dplyr::select(Death, Y.Death) %>%
  mutate(YDA_Death = lag(Y.Death)) %>%
  dplyr::filter(!is.na(YDA_Death))
```

```
## Adding missing grouping variables: `State`
```

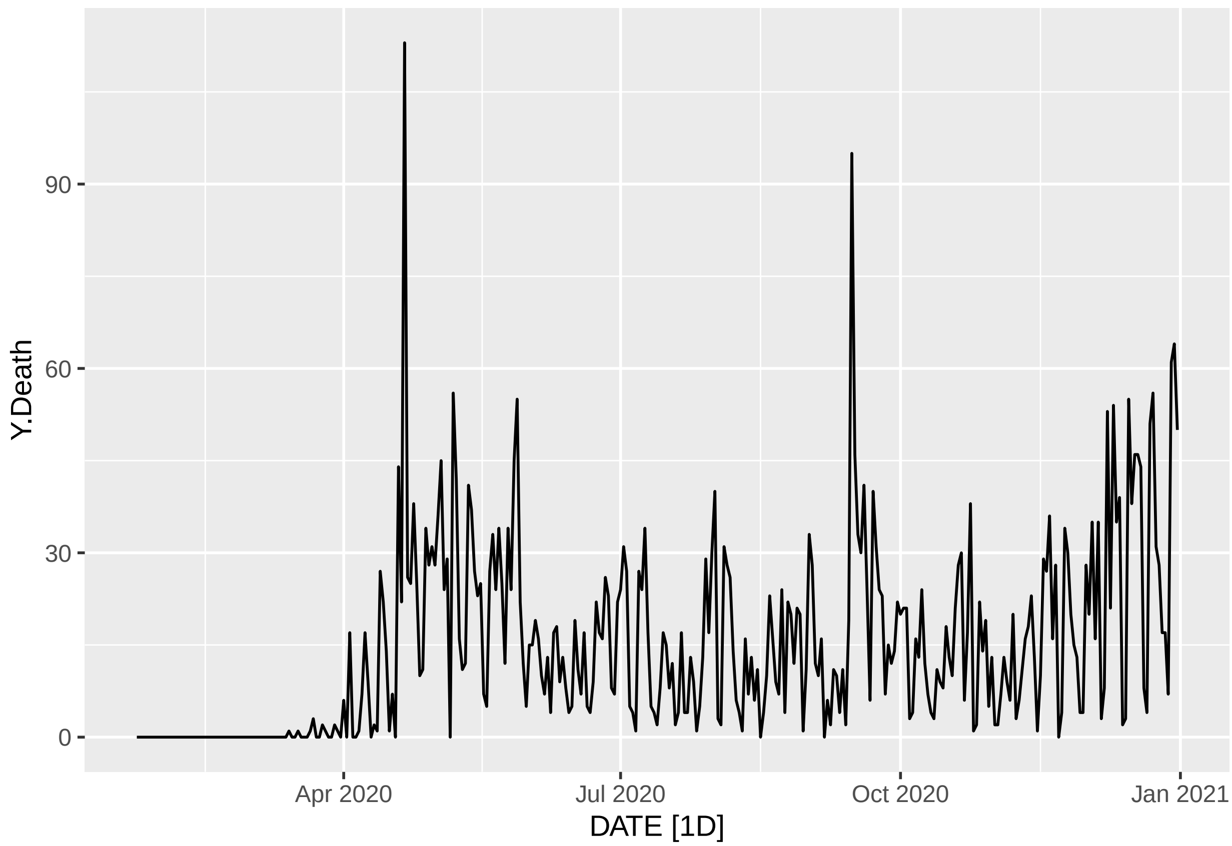
```
head(Virginia.ts )
```

```
## # A tsibble: 6 x 5 [1D]
## # Key:       State [1]
## # Groups:    State [1]
##   State    Death Y.Death DATE       YDA_Death
```

```
##    <chr>    <int>    <int> <date>          <int>
## 1 Virginia     0        0 2020-01-24         0
## 2 Virginia     0        0 2020-01-25         0
## 3 Virginia     0        0 2020-01-26         0
## 4 Virginia     0        0 2020-01-27         0
## 5 Virginia     0        0 2020-01-28         0
## 6 Virginia     0        0 2020-01-29         0
```

Creatiung the Training Data

```
# time series plot death counts in Florida
Virginia.ts %>%
  autoplot(Y.Death)
```



set training data from NOV 28 to DEC 04

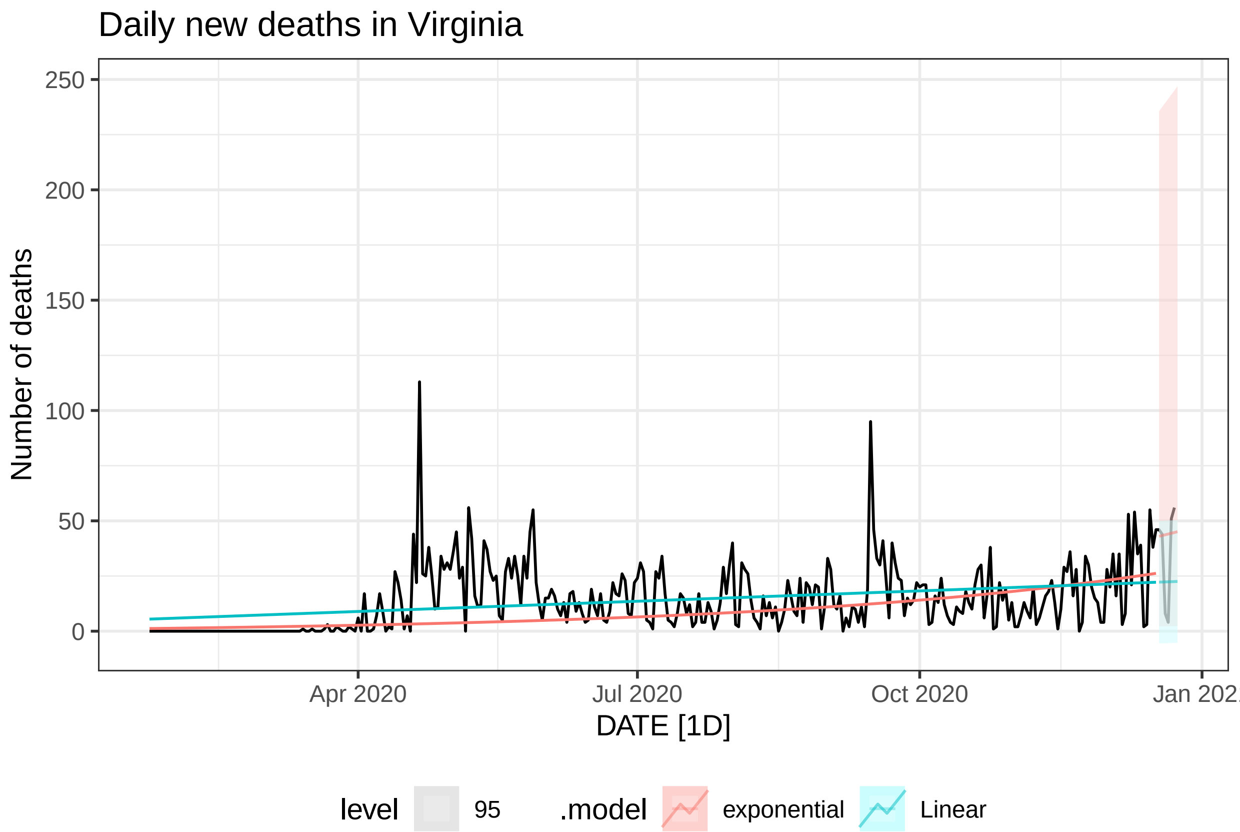
```
train <- Virginia.ts %>%
  filter_index("2020-01-23" ~ "2020-12-17
             ")
n <- nrow(train)
```

```
fit_trends <- train %>%
  model(
    Linear = TSLM(Y.Death ~ trend()),
    exponential = TSLM(log(Y.Death + 1) ~ trend()),
```

```
  )

fc_trends <- fit_trends %>% fabletools::forecast( h = 7)
```

Making Predictions

```
Virginia.ts %>%
  dplyr::filter(DATE < train$DATE[n] + 7 ) %>%
  autoplot(Y.Death) +
  geom_line(data = fitted(fit_trends),
            aes(y = .fitted, color = .model)) +
  autolayer(fc_trends, alpha = 0.5, level = 95) +
  labs(y = "Number of deaths",
       title = "Daily new deaths in Virginia") +
  theme_bw() +
  theme(legend.position = "bottom")
```




```
lm_fit <- train%>%
  model(lm = TSLM(Y.Death ~ log(YDA_Death + 1)))
report(lm_fit)
```

```
## Series: Y.Death
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

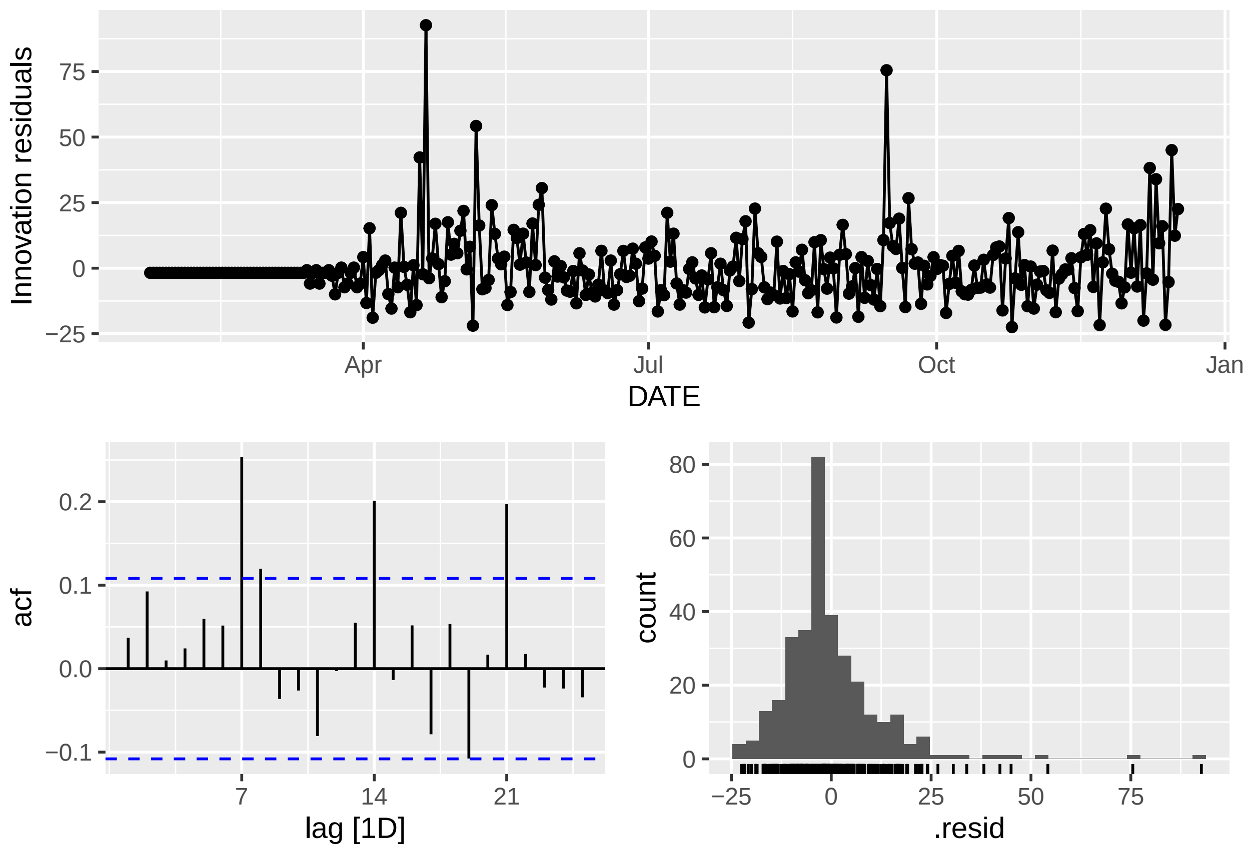
```
## -22.471  -7.293  -1.772   4.036  92.657
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.7724     1.2915   1.372    0.171
## log(YDA_Death + 1)   5.9227     0.5335  11.102   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.69 on 327 degrees of freedom
## Multiple R-squared: 0.2737,  Adjusted R-squared: 0.2715
## F-statistic: 123.2 on 1 and 327 DF, p-value: < 2.22e-16
```

Fitting the ETS Model

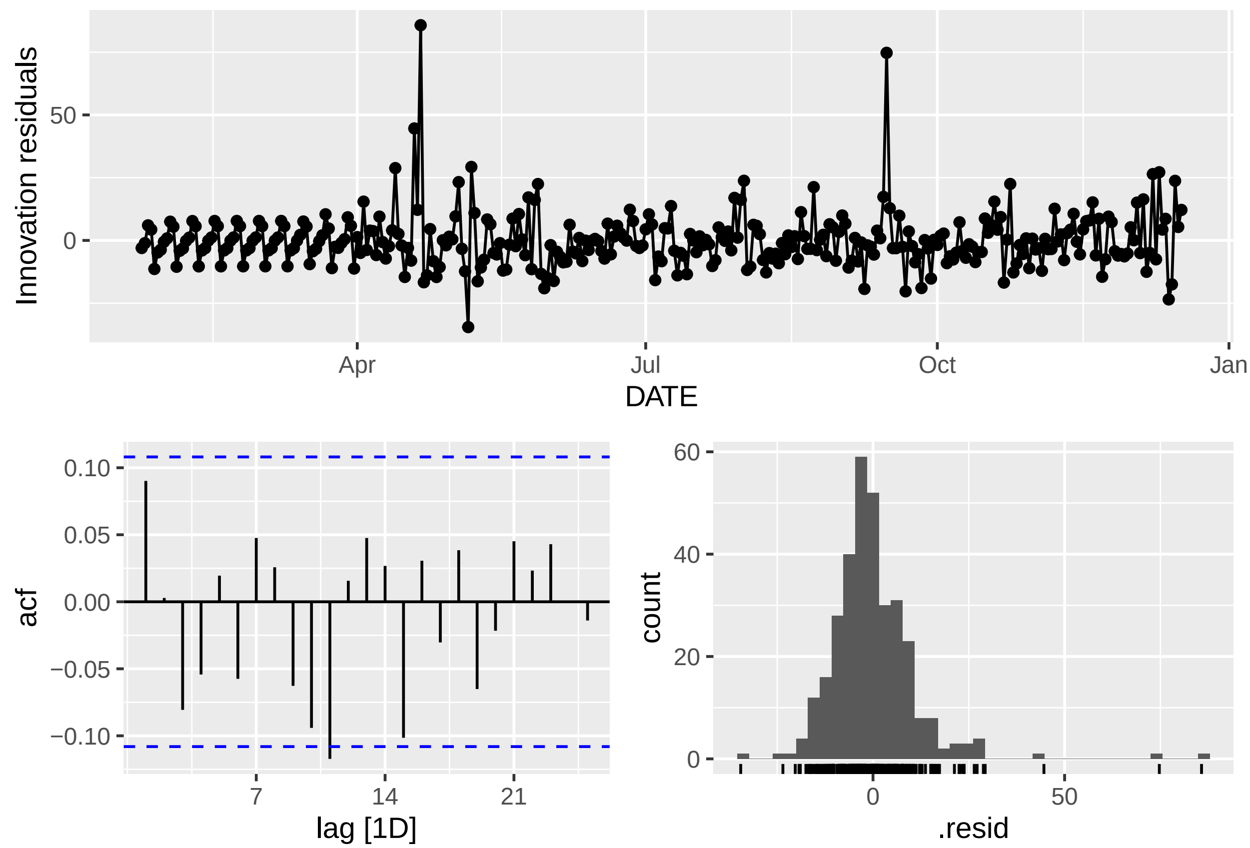
```
ets_fit <-train %>%
  model(ETS(Y.Death ~ error("A") + trend("A") + season("A"), opt_crit =
"mse"))
report(ets_fit)
```

```
## Series: Y.Death
## Model: ETS(A,A,A)
##   Smoothing parameters:
##     alpha = 0.2261085
##     beta  = 0.0001000967
##     gamma = 0.0001110448
##
##   Initial states:
##      l[0]       b[0]     s[0]    s[-1]   s[-2]     s[-3]    s[-4]      s[-5]
##  2.033307 0.09267903 3.281893 3.369762 7.46685 -7.164426 -7.38868 -0.4560142
##      s[-6]
##  0.8906152
##
##   sigma^2:  130.4125
##
##      AIC     AICc      BIC
## 3522.176 3523.163 3567.729
```

```
residual_plot_linear_model <- gg_tsresiduals(lm_fit)

residual_plot_linear_model
```




```
residual_plot_ets_model <- gg_tsresiduals(ets_fit)

residual_plot_ets_model
```



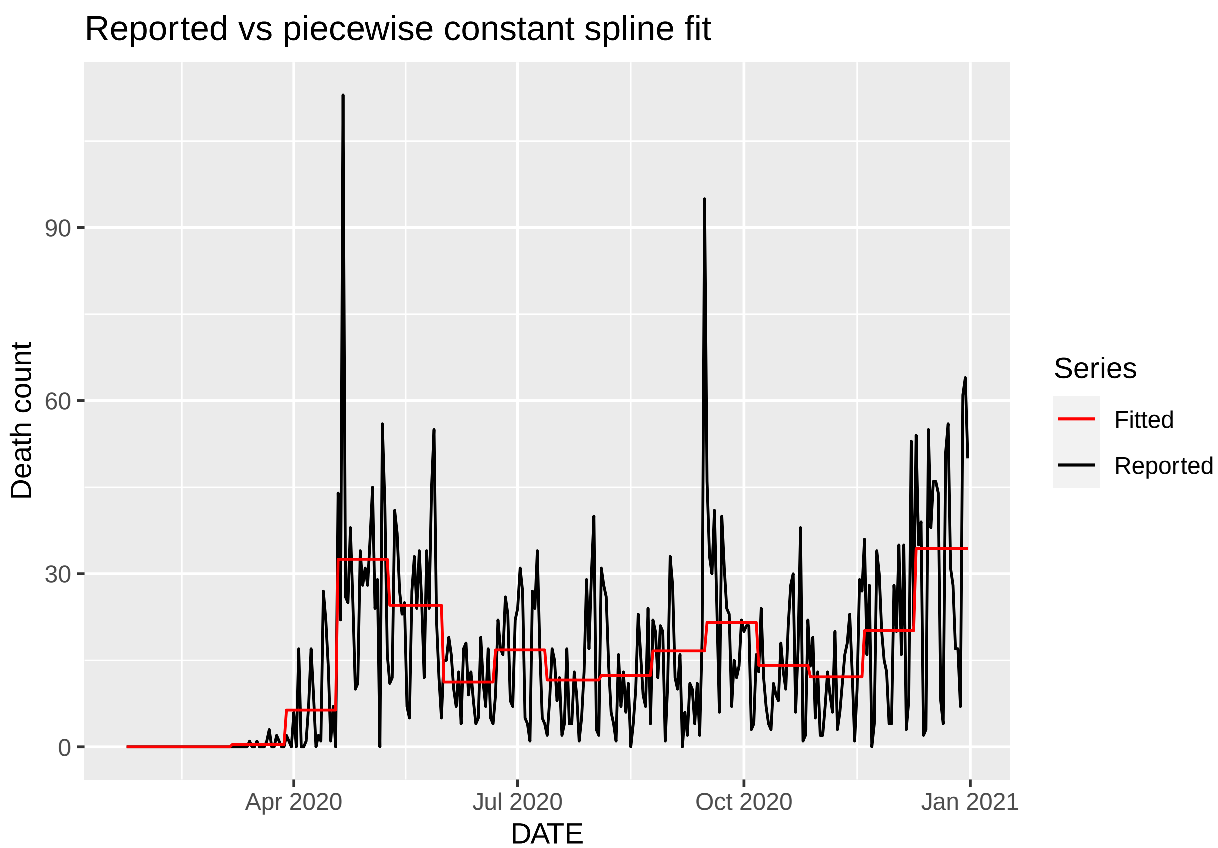
Yes the residuals appear to be reasonably normally distributed

Piecewise constant spline regression model with 15 interior knots

```
n <- nrow(Virginia.ts)
t <- 1:n
y <- Virginia.ts$Y.Death

# Knots
N <- 15
knots <- 1 + (n-1)/(N+1) * (0:N)

# Piecewise constant spline basis
t.rep <- matrix(rep(t, N), n, N)
knot.L <- matrix(rep(knots[-(N + 1)], each = n), n, N)
knot.R <- matrix(rep(knots[-1], each = n), n, N)
B <- 1*((knot.L <= t.rep) & (t.rep < knot.R))
X <- cbind(B, knots[N] < t & t <= n)

M <- t(X) %*% X
beta <- solve(M) %*% t(X) %*% y
yhat <- X %*% beta
Virginia.ts$pcs_preds <- yhat
```

```
# plot of reported vs piecewise constant spline fit
pcs_p <- Virginia.ts %>%
```

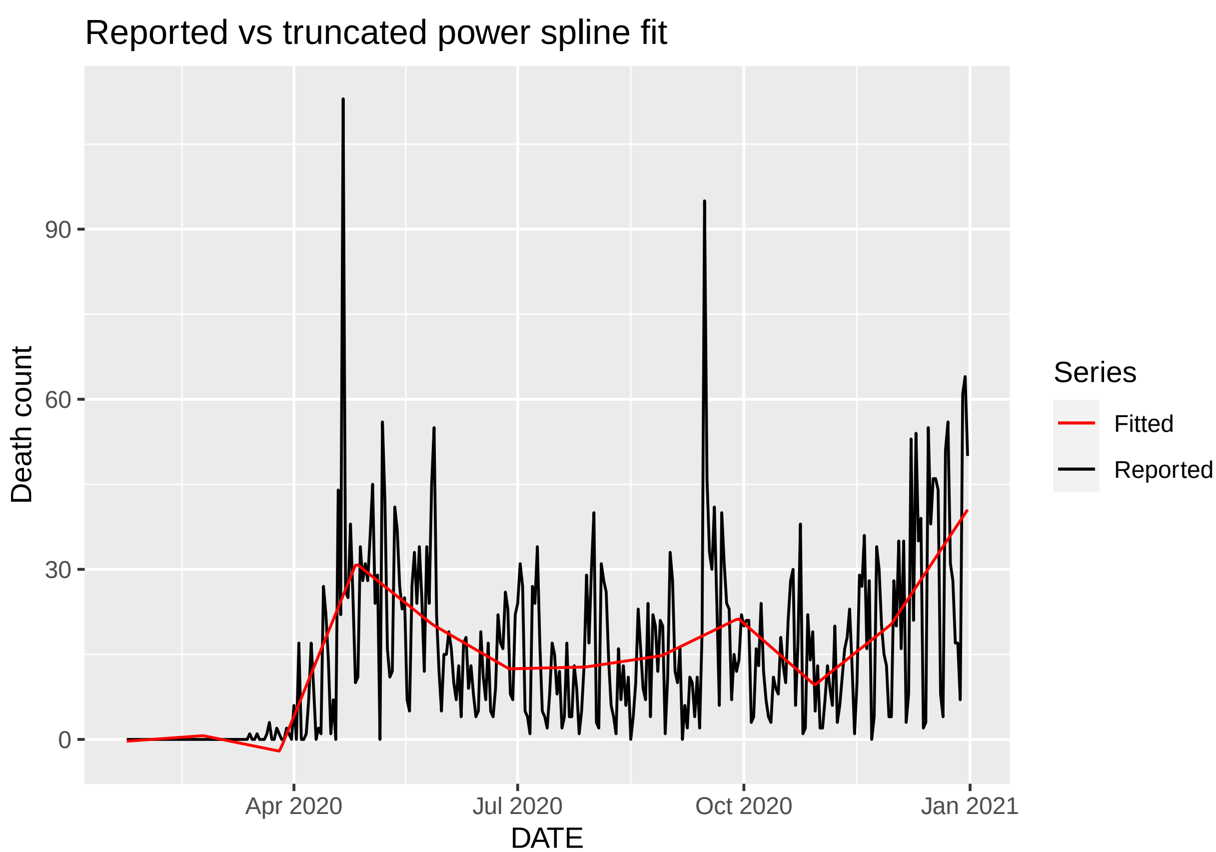
```
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Y.Death, color = "Reported")) +
  geom_line(aes(y = pcs_preds, color = "Fitted")) +
  scale_color_manual(
    values = c(Reported = "black", Fitted = "red")
  ) +
  labs(y = "Death count",
       title = "Reported vs piecewise constant spline fit") +
  guides(color = guide_legend(title = "Series"))

pcs_p
```



Truncated Power Spline

```
y <- Virginia.ts$Y.Death
n <- nrow(Virginia.ts)
t <- 1:n

# knots
N <- 10
knots <- 1 + (n-1)/(N+1) * (0:N)

# truncated power spline basis functions
X <- matrix(1, n, N + 2)
X[, 2] <- t
```

```r
t.rep <- matrix(rep(t, N), n, N)
tmp <- t.rep - matrix(rep(knots[2:(N + 1)], each = n), n, N)
X[, 3:(N+2)] <- tmp * (tmp > 0)

# truncated power spline fit
M <- t(X) %*% X
beta <- solve(M) %*% t(X) %*% y
yhat <- X %*% beta
Virginia.ts$tps_preds <- yhat

# plot of reported vs truncated power spline fit
tps_p <- Virginia.ts %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Y.Death, color = "Reported")) +
  geom_line(aes(y = tps_preds, color = "Fitted")) +
  scale_color_manual(
    values = c(Reported = "black", Fitted = "red")
  ) +
  labs(y = "Death count",
       title = "Reported vs truncated power spline fit") +
  guides(color = guide_legend(title = "Series"))

tps_p
```



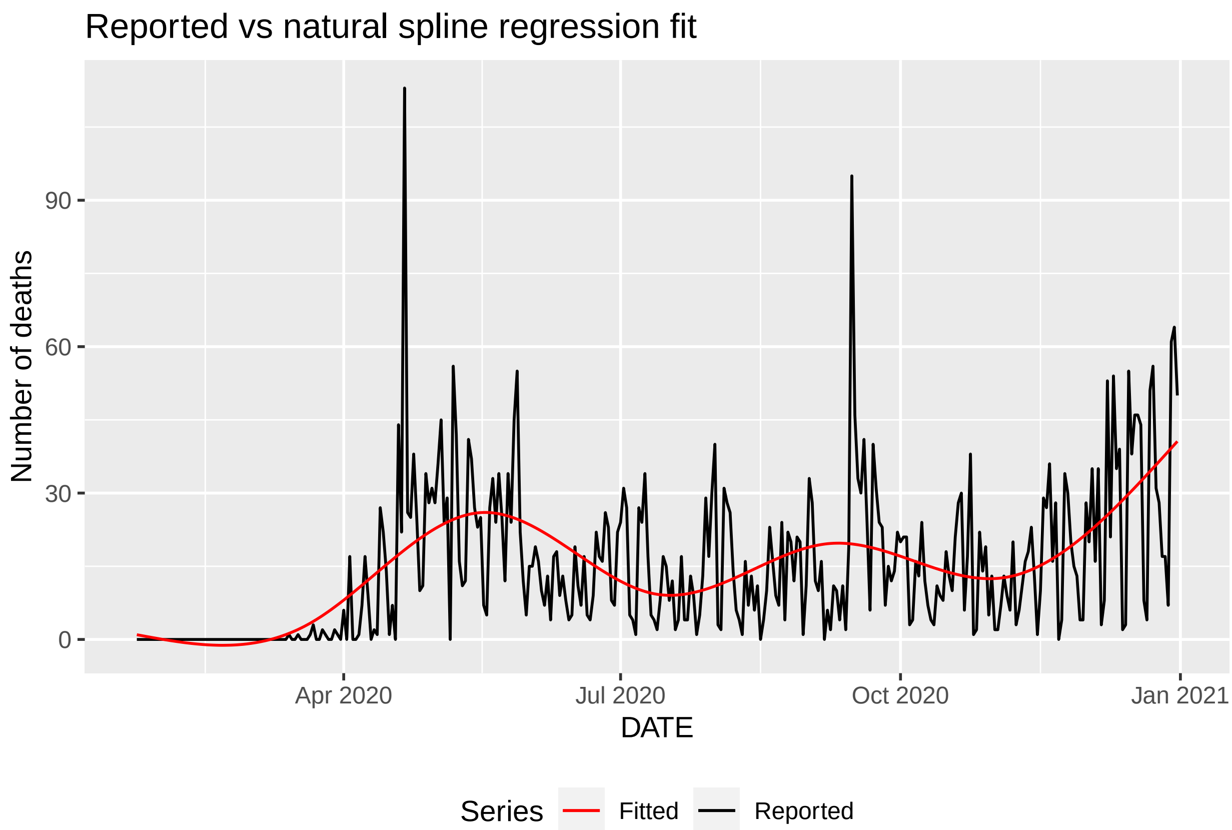
Natural spline regression model with 8 interior knots.

```
library(splines)
n <- nrow(Virginia.ts)
t <- 1:n
ns_fit <- lm(Y.Death ~ ns(t, df = 6), data = Virginia.ts)
summary(ns_fit)
```

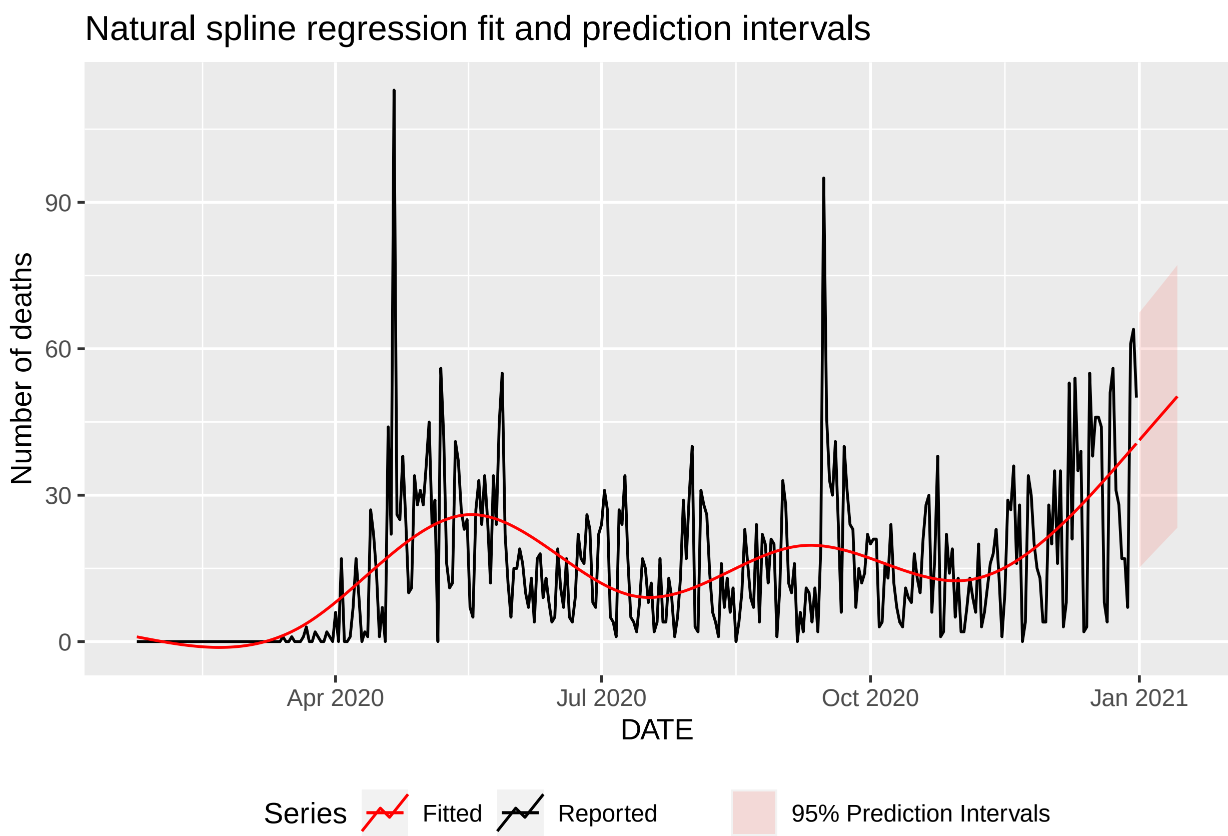
```
##
## Call:
## lm(formula = Y.Death ~ ns(t, df = 6), data = Virginia.ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.524  -7.406  -0.625   5.634  94.647
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.9727     3.3739   0.288    0.773
## ns(t, df = 6)1   40.4653     4.2468   9.529  < 2e-16 ***
## ns(t, df = 6)2   -5.0013     5.4424  -0.919    0.359
## ns(t, df = 6)3   29.2251     4.8350   6.044 3.98e-09 ***
## ns(t, df = 6)4    2.1764     4.2141   0.516    0.606
## ns(t, df = 6)5   20.9701     8.5750   2.445    0.015 *
## ns(t, df = 6)6   43.3173     3.8593  11.224  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.83 on 336 degrees of freedom
## Multiple R-squared:  0.3427, Adjusted R-squared:  0.3309
## F-statistic: 29.19 on 6 and 336 DF,  p-value: < 2.2e-16
```

```
 Virginia.ts$ns_preds <- predict(ns_fit)

# Natural spline prediction and prediction intervals
h <- 14
t.new <- t[n] + (1:h)
ns_PI <- predict(ns_fit, newdata = data.frame(t = t.new),
                 interval = "prediction", level = 0.95)

ns_PI <- as.data.frame(ns_PI) %>%
  mutate(DATE = ( Virginia.ts$DATE)[n] + 1:h)

# Plot of reported vs natural spline fit
ns <- Virginia.ts%>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Y.Death, color = "Reported")) +
  geom_line(aes(y = ns_preds, color = "Fitted")) +
  scale_color_manual(
    values = c(Reported = "black", Fitted = "red")
  ) +
  labs(y = "Number of deaths",
       title = "Reported vs natural spline regression fit") +
  guides(color = guide_legend(title = "Series")) +
```

```
  theme(legend.position = "bottom")

ns
```




```
# Plot of natural spline fit and its prediction intervals
ns_int <- ns +
  geom_ribbon(
    mapping = aes(y = fit,
                  ymin = lwr,
                  ymax = upr,
                  fill = '95% Prediction Intervals'),
    data = ns_PI, alpha = 0.2) +
  geom_line(mapping = aes(y = fit, color = "Fitted"),
            data = ns_PI,
            key_glyph = "timeseries") +
  labs(title = "Natural spline regression fit and prediction intervals")  +
  guides(color = guide_legend(title = "Series"),
         fill =  guide_legend(title = "")) +
  theme(legend.position = "bottom")

ns_int
```



Smoothing spline regression model with knots automatically selected by the “mgcv” package.

```
if(!require('mgcv')) install.packages('mgcv')
```

```
## Loading required package: mgcv

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:feasts':
##
##     ACF

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.
```

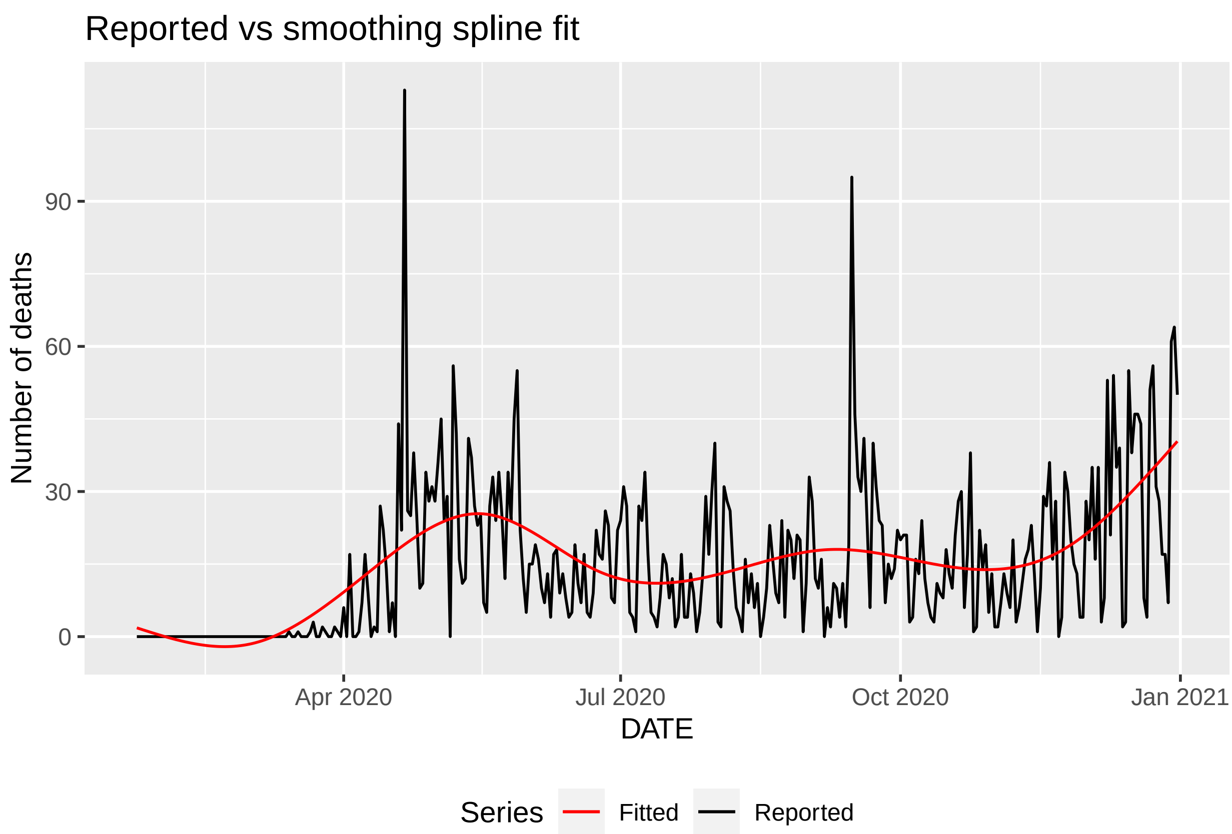
```
library(mgcv)
```

```
ss_fit <- gam(Y.Death ~ s(t, bs = "cr"), data = Virginia.ts)
summary(ss_fit)
```

```
##
## Family: gaussian
## Link function: identity
```

```
##
## Formula:
## Y.Death ~ s(t, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.6706     0.6927   21.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
## s(t) 7.733  8.585 19.87  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.331   Deviance explained = 34.6%
## GCV = 168.88  Scale est. = 164.58    n = 343
```

```r
Virginia.ts$ss_preds <- predict(ss_fit)

# Plot of reported vs smoothing spline fit
Virginia.ts %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Y.Death, color = "Reported")) +
  geom_line(aes(y = ss_preds, color = "Fitted")) +
  scale_color_manual(
    values = c(Reported = "black", Fitted = "red")) +
  labs(y = "Number of deaths",
       title = "Reported vs smoothing spline fit") +
  guides(color = guide_legend(title = "Series"))  +
  theme(legend.position = "bottom")
```


Reported vs smoothing spline fit
Number of deaths
90
60
30
0
Apr 2020
Jul 2020
Oct 2020
Jan 2021
DATE
Series
Fitted
Reported