Metrics PSet 3

0 Set up environment, load packages and data

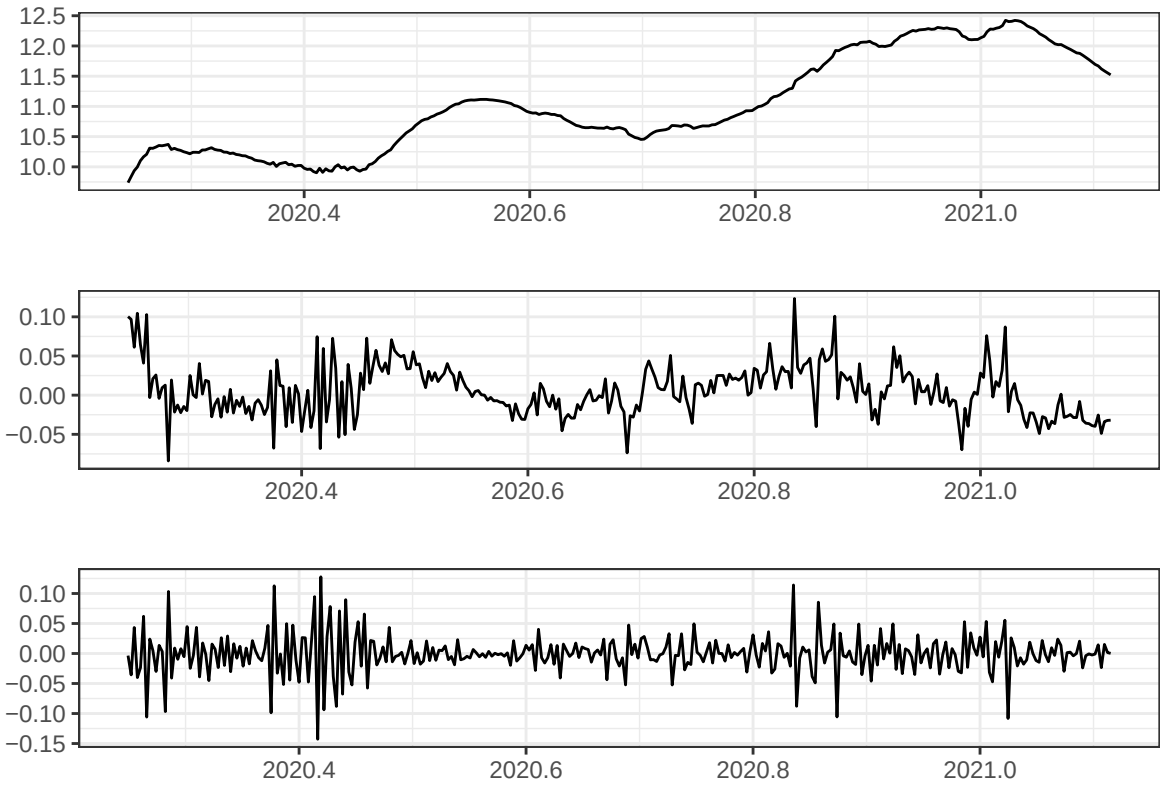
```
rm(list = ls())
set.seed(1)
init_dir <- getwd()
# Set paths
root <- paste0("/Users/tombearpark/Documents/princeton/",
               "1st_year/term2/ECO518_Metrics2/")
dir <- paste0(root, "sims/exercises/3_AR/")
out <- paste0(dir, "out/")
setwd(dir)

# Load packages
if(!require(IDex2019))
  install.packages("IDex2019", repos = NULL, type = "source")
```

```
## Loading required package: IDex2019
```

```
if(!require(pacman)) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
pacman::p_load(IDex2019, tidyverse, ggfortify, patchwork, zoo)
theme_set(theme_bw())

# I source this one separately, as I added an option to this function
# to not output his default plot
source(paste0(dir, "/IDex2019/R/fcastBand.R"))
# Load data, and plot a simple time series
load("logcovidcases.RData")
df <- cfd19ts
# Save in a convenient format for later plotting
plot_df <- df %>% as.data.frame() %>%
  mutate(date = row_number() - length(df))


# Required for the RMD file to run nicely
setwd(init_dir)
# Note - doesn't look very stationary! Neither do the differences
autoplot(df) /
autoplot(diff(df)) /
autoplot(diff(df, differences = 2))
```



1. Fit a 9th order linear AR model to the data.

Estimation of the model can be done with a call to rfvar3(). By default rfvar3() uses an improper prior that shrinks toward persistence. You can omit that prior by setting lambda=NULL, mu=NULL. You can choose whether or not to use the prior.

```
# ?rfvar3
AR9 <- rfvar3(
  ydata = df,
  lags = 9,
  xdata = NULL,
  const = TRUE,
  breaks = NULL,
  lambda = NULL,
  mu = NULL,
  ic = NULL,
  sigpar = NULL)

# Check I can get the same thing in base R
baseR_AR <- ar(df, aic = FALSE, order.max = 9, method = "ols")
data.frame(sims = AR9$By[,,seq(1, dim(AR9$By)[3])],
                        baseR = baseR_AR$ar)
```

```
##           sims        baseR
## 1  1.233095111  1.233095111
## 2  0.018438252  0.018438252
## 3 -0.065427731 -0.065427731
```

```
## 4 -0.076003362 -0.076003362
## 5 -0.002085195 -0.002085195
## 6 -0.041047493 -0.041047493
## 7 -0.445553401 -0.445553401
## 8  0.531587672  0.531587672
## 9 -0.154845942 -0.154845942
```

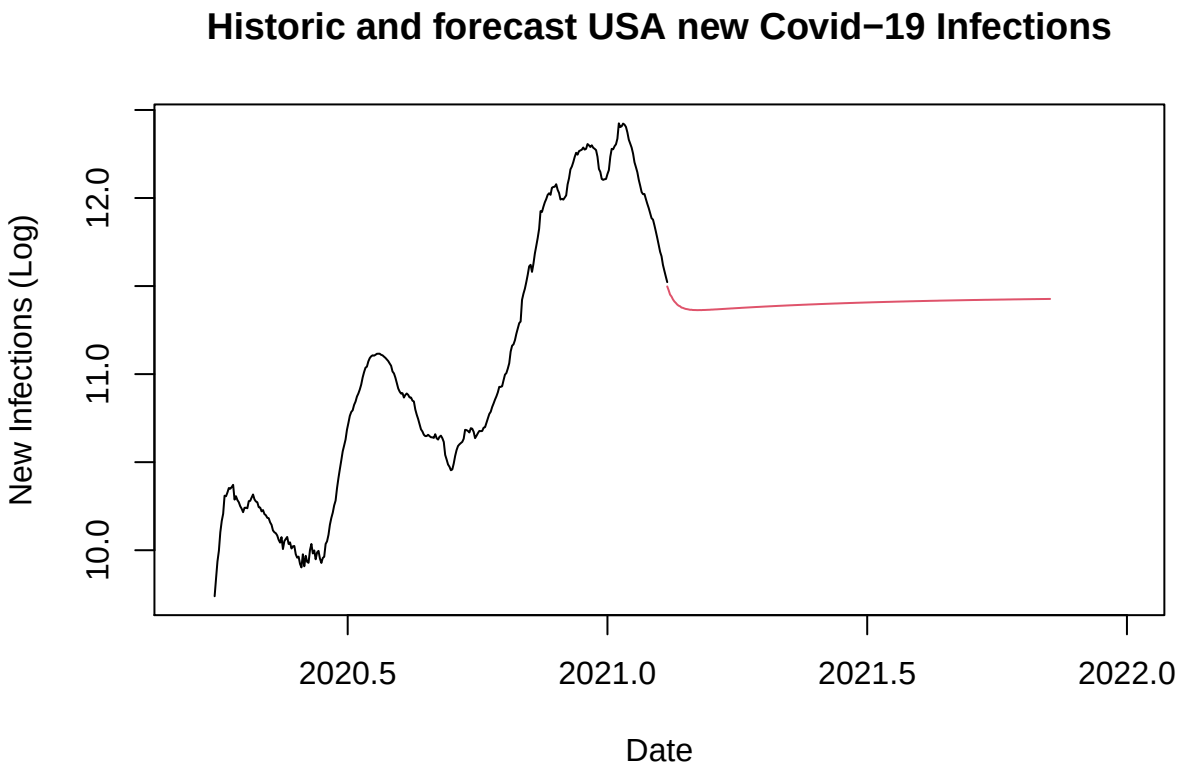
2. Forecast the next 270 days, using the estimates

The y0 agument of the forecasting commands should be dimensioned as a 9 x 1 matrix. It should also be a time series object with start date and frequency. Check that this is true with str(y0). If not,use y0 <-ts(matrix(y0, ncol=1), end=c(2021,42), freq=365). The c(2021, 42) specifies day 42 (i.e.February11) of 2021.

```
# Format initial condition
y0 <- df[(length(df) - 8):length(df)]
y0 <- ts(matrix(y0, ncol=1),
        end=c(2021,42),
        freq=365)
str(y0)
```

```
##  Time-Series [1:9, 1] from 2021 to 2021: 11.8 11.8 11.7 11.7 11.7 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "Series 1"
```

```
# A forecast using a single set of parameter values can be generated with
# fcast()

f <- fcast(y0 = y0,          # values of time series to forecast from
           By = AR9$By,      # fitted AR coefficients from part 1
           Bx = AR9$Bx,
           xdata = NULL,     # exogenous variables (we don't have any)
           horiz = 270,      # forecast horizon
           const = TRUE,
           shocks = NULL
           )

# Plot!
f <- window(f, start = c(2021,43))
ts.plot(df, xlim = (c(2020.2, 2022)),
        main = "Historic and forecast USA new Covid-19 Infections",
        ylab = "New Infections (Log)", xlab = "Date")
points(f, type = "l", col = 2)
```



3.

Sample 1000 draws from the posterior distribution of AR coefficients and

residual variance to generate a forecast with error bands around the

forecast that reflects (only) the uncertainty about the parameters of

the model. (Make these 90% and 68% bands.)

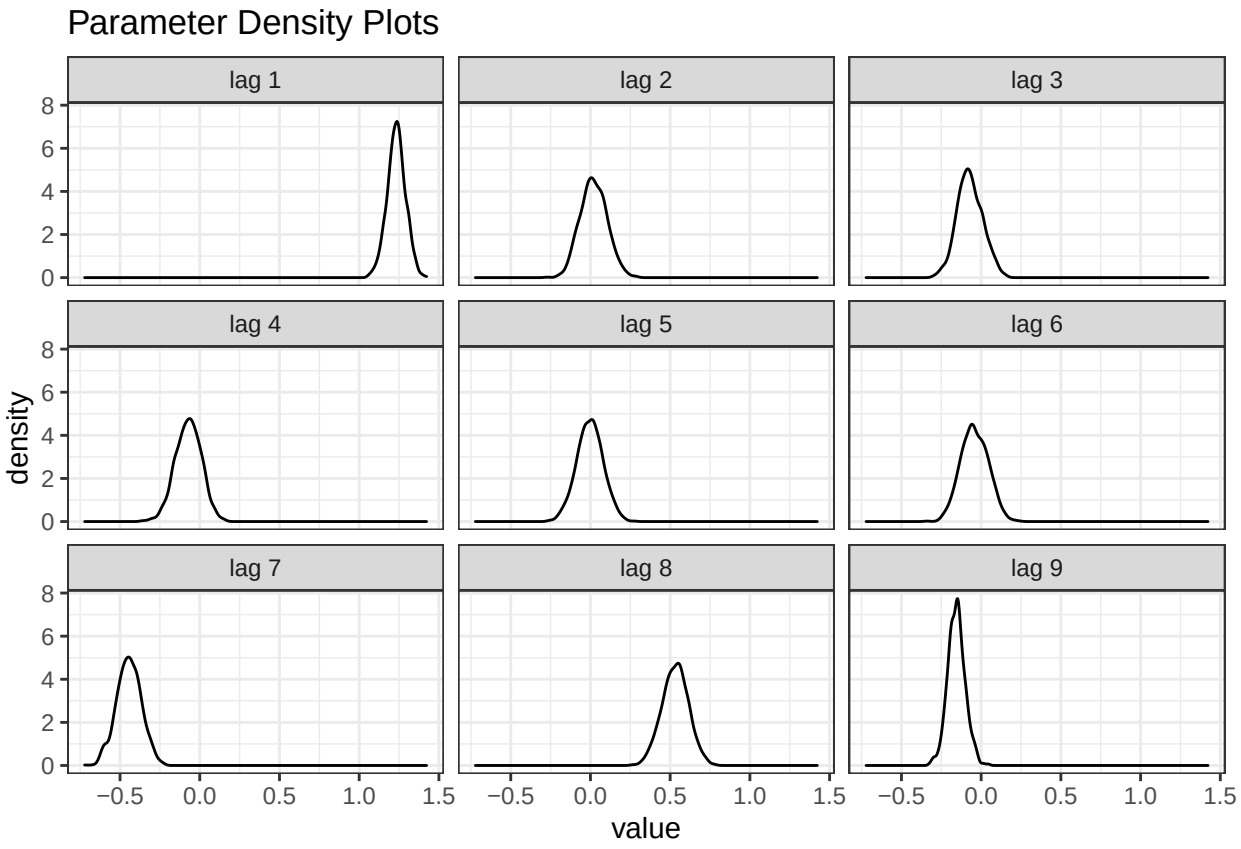
Generation of draws from the posterior for the parameters can then be done with a call to postdraw()...

```
# ?postdraw
draws <-
  postdraw(
    AR9,               # fitted AR model
    n = 1000,          # number of draws
    nosigprior = TRUE  # specify that we don't have a jeffrey's prior
    )
```

```
# Convert to a format I know how to work with
draws_df <-
  draws$By[,,1:9, 1:1000] %>%
  t() %>%
  as_tibble()
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
```

```
# Plot the parameter pdfs
draws_df %>%
  pivot_longer(cols = V1:V9, names_to = "lag") %>%
  mutate(lag = paste0("lag ", substr(lag, 2, 2))) %>%
  ggplot() +
  geom_density(aes(x = value)) +
  facet_wrap(~lag) +
  ggtitle("Parameter Density Plots")
```



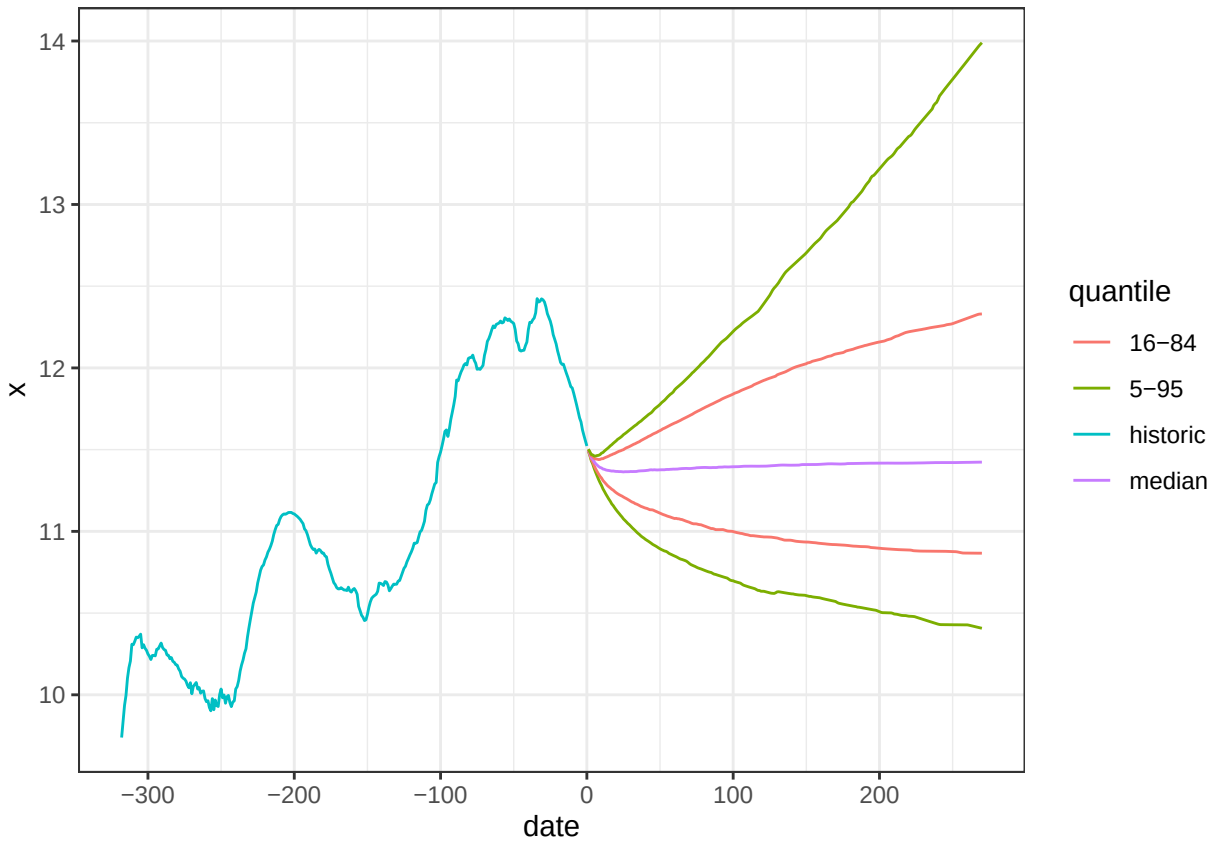
A set of draws from the posterior on the forecast can be generated with fcastBand(). This program will make plots showing error bands and will return an array of sampled forecasts, unsorted. It can show uncertainty based on parameter values alone (with whichs=0 or, with whichs left at its default value, based on both parameter and future shock uncertainty.

```
# ?fcastBand

p_vals <- c(5, 16, 50, 84, 95)
```

```r
fc_draws <-
  fcastBand(
    pdout = draws,
    y0,
    horiz = 270,
    pctiles = p_vals,
    # whichv = NULL,
    whichs = 0,
    main = "Parameter Uncertainty Forecast Bands",
    file = paste0(out, "3_param_only_uncertainty.pdf"),
    xdata = NULL,
    const = TRUE,
    plot = FALSE
  )

# Function for converting the results of fcastBand to a dataframe
format_fc <- function(fc_draws){
  fc_draws$fc %>%
    as.data.frame.table() %>%
    mutate(date = rep(1:279, 1000)) %>%
    group_by(Var3) %>%
      mutate(draw = cur_group_id()) %>%
    ungroup() %>%
    select(-c(Var1, var, Var3), value = Freq) %>%
    filter(date > 9) %>%
    mutate(date = rep(1:270, 1000))
}

# Function to take quantiles across draws at each date, return a nice plot
plot_quantiles <- function(forecast_bands, p_vals, plot_df, title=NULL){
  forecast_bands %>%
    group_by(date) %>%
    summarise(x = quantile(value, p_vals / 100),
              q = p_vals/ 100) %>%
    ungroup() %>%
    mutate(quantile = case_when(
      q %in% c(0.05, 0.95) ~ "5-95",
      q %in% c(0.16, 0.84) ~ "16-84",
      q == 0.5 ~ "median"
    )) %>%
    bind_rows(plot_df %>% mutate(
              q = 0.5001, quantile = "historic")) %>%
    ggplot() +
    geom_line(aes(x = date, y = x, group = q, color = quantile))
}

fc_draws <- format_fc(fc_draws)
p <- plot_quantiles(fc_draws, p_vals, plot_df)
```

```
## `summarise()` has grouped output by 'date'. You can override using the `.groups` argument.
```

```
p
```



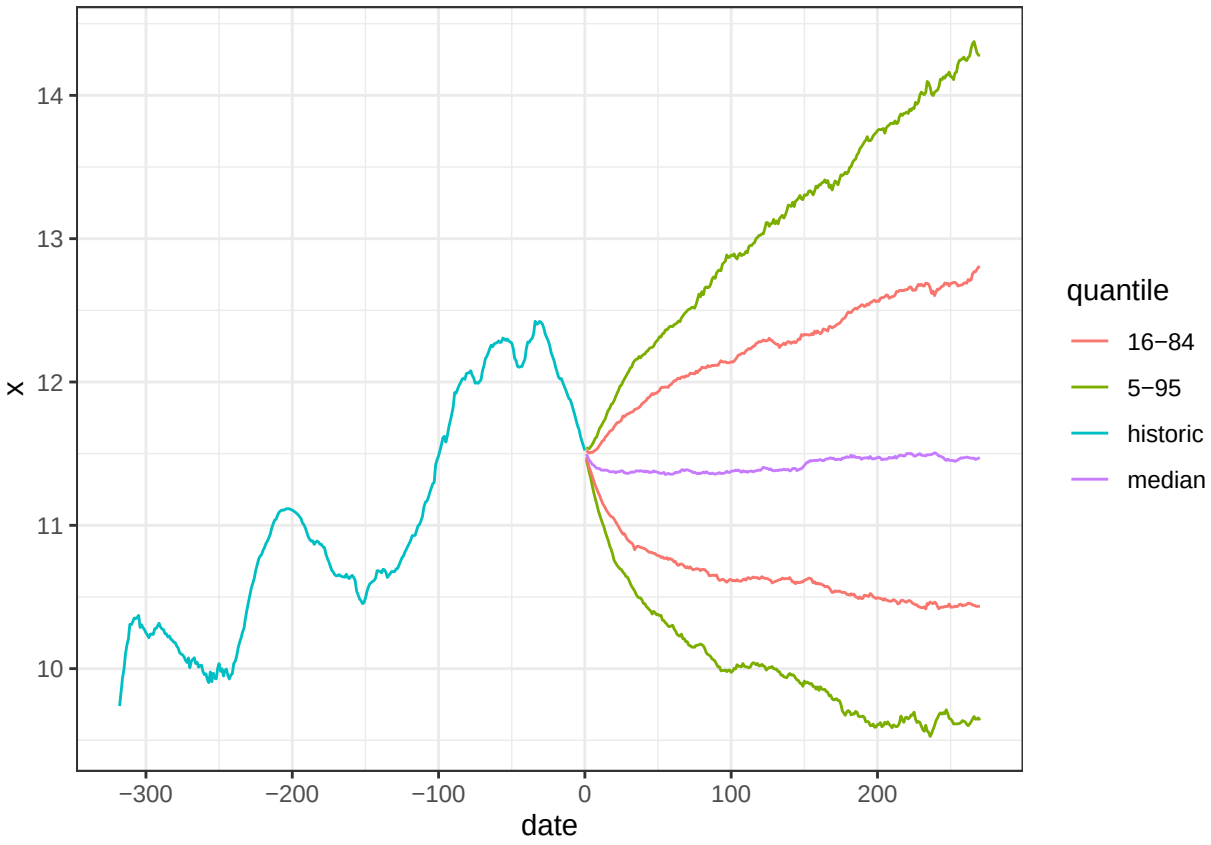
(4) Generate error bands that include effects both of uncertainty about the

model parameters and uncertainty about future disturbance terms.

```
fc_draws_full <-
  fcastBand(
    pdout = draws,
    y0,
    horiz = 270,
    pctiles = p_vals,
    main = "Parameter and Shock Uncertainty Forecast Bands",
    file = paste0(out, "3_param_and_shock_uncertainty.pdf"),
    xdata = NULL,
    const = TRUE,
    plot = FALSE
  )
fc_draws_full <- format_fc(fc_draws_full)
q <- plot_quantiles(fc_draws_full, p_vals, plot_df, title= "FULL")
```

```
## `summarise()` has grouped output by 'date'. You can override using the `.groups` argument.
```

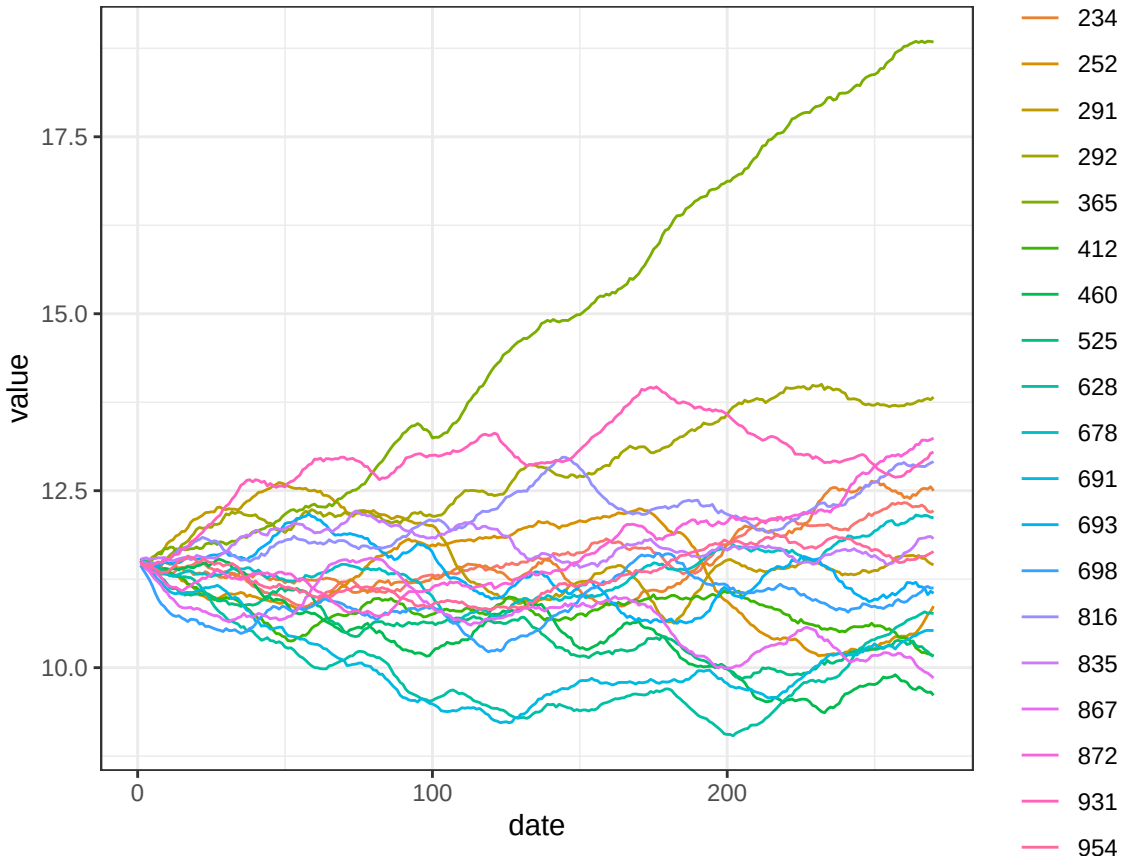
```
q
```



(5) Plot, on a single graph, 20 of the randomly drawn forecast time series that include the effects of both kinds of uncertainty.

(This is a different way of visualizing forecast uncertainty.)

```
sample_index <- sample(1:1000, 20)

fc_draws_full %>%
  filter(draw %in% sample_index)  %>%
  ggplot() +
  geom_line(aes(x = date, y = value, color = as.factor(draw)))
```



(6) Using the same draws from the posterior and future shocks, evaluate the posterior probability that the rate of new infections is smaller at the end of the forecast period than at the start.

```
fc_draws_full %>%
  filter(date %in% c(1, 270)) %>%
  pivot_wider(names_from = date, names_prefix = "date") %>%
  mutate(increased = ifelse(date1<date270, 1, 0)) %>%
  summarise(p = mean(increased))
```

```
## # A tibble: 1 x 1
##       p
##   <dbl>
## 1 0.491
```

(7) Using the unconditional joint pdf of the initial conditions implied by the point-estimate of the parameters, calculate how many standard

deviations from the process mean is the initial observation.

If your point estimates imply an unstable root, you won't be able to

do this part, so start by calculating the roots and checking whether

they are all in the stable region.

```
coef_vec <-
  AR9$By[,,] %>%
  unlist()

roots <-
  coef_vec %>%
  as.vector() %>%
  polyroot() %>%
  Mod()
roots
```

```
## [1] 1.178137 1.034736 1.034736 1.178137 1.017086 1.691496 1.691496 1.841393
```

We can see they are all outside the unit circle!

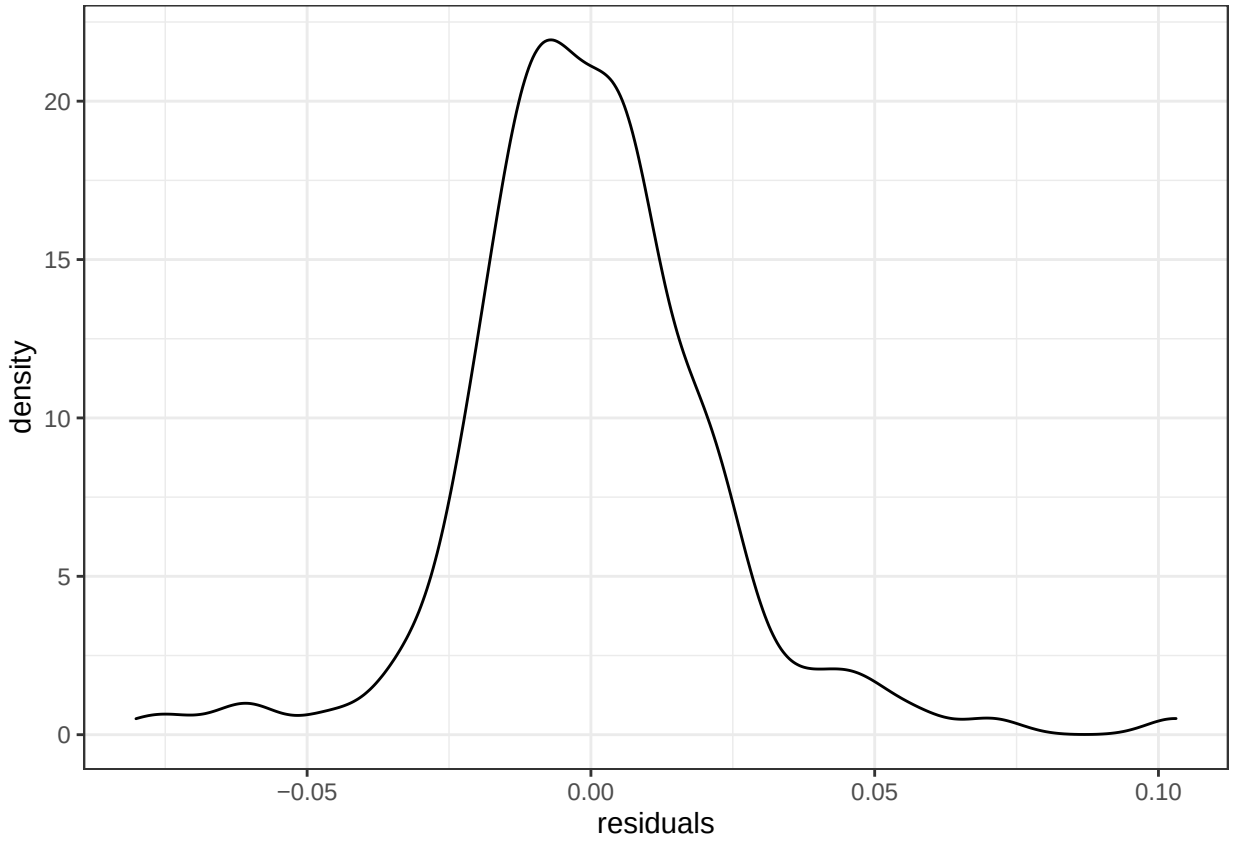
```
# Calculate distance
process_mean <- mean(df)
acf <- ARMAacf(ar = coef_vec, ma = NULL, lag.max = 9, pacf = TRUE)
(process_mean - df[1]) / sqrt(acf[1])
```

```
## [1] 1.287204
```

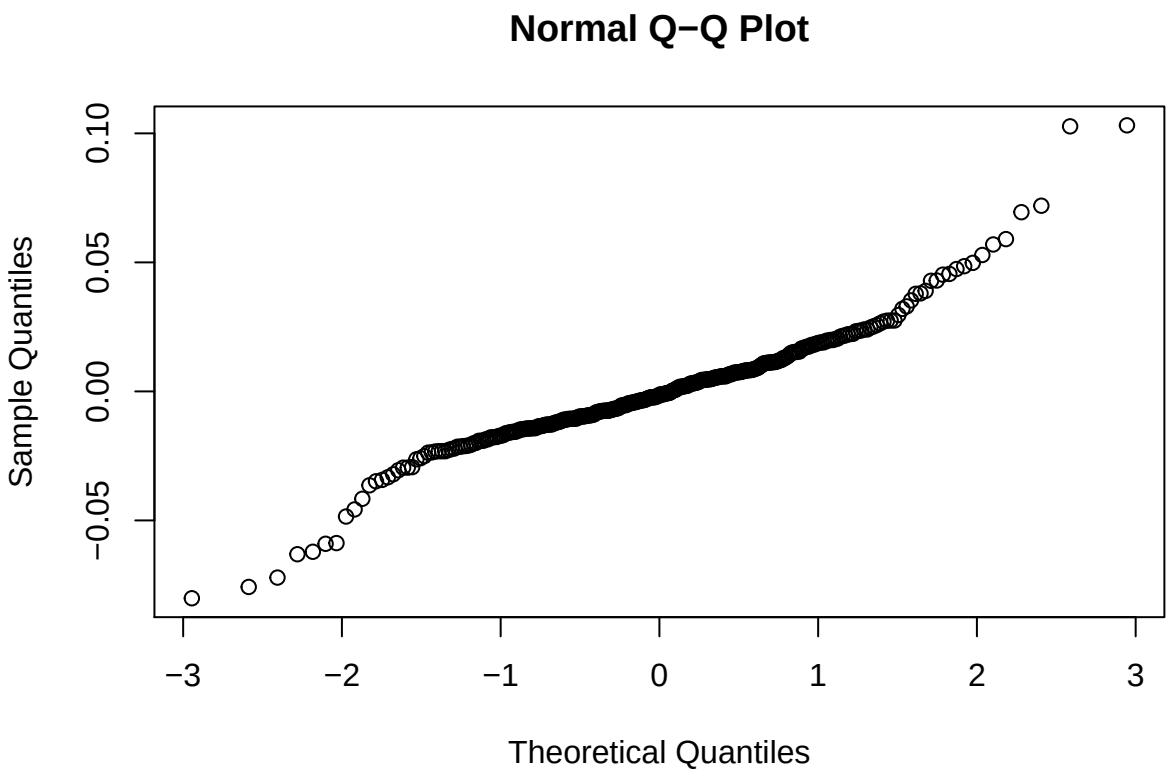
(8) Use a histogram of the residuals and a normal q-q plot of them to assess whether the normality assumption is a reasonable approximation.

```
residuals <-
  AR9$u %>%
  as.data.frame() %>%
  rename(residuals = 1)
```

```r
# Histogram of residuals
residuals %>%
  ggplot() +
  geom_density(aes(x = residuals))
```




```r
# QQplot
qqnorm(residuals$residuals)
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
0.10
0.05
0.00
−0.05
−3
−2
−1
0
1
2
3