Trabalho - Econometria IV

Guilherme Luz, Guilherme Masuko, Caio Garzeri

August 2023

```r
library(lubridate)  # for handling dates
library(randomForest)  # Random Forest implementation of the original Fortran code by Brieman (2001)
library(ranger)  # Faster implementation of Random Forest
```

Question 3

Item D

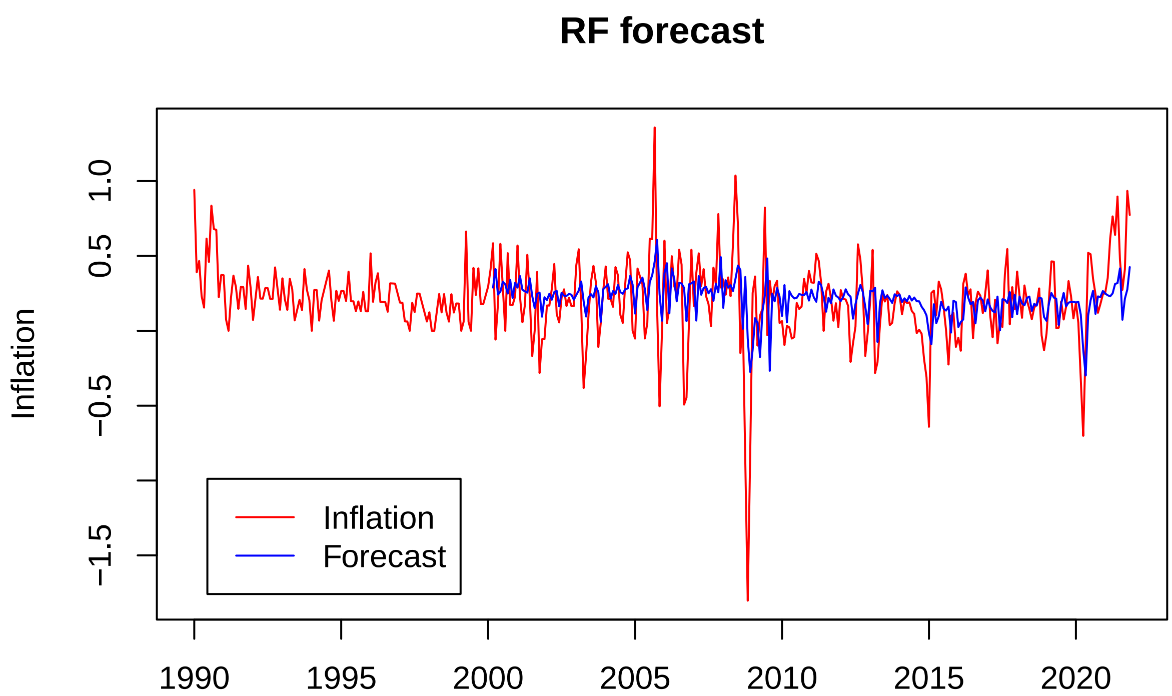
In order to include the lags of the variables as covariates, we need to do an embedding process. explicar. (We do this inside the rolling window loop to avoid 'cheating').

After this process, we can use the usual IID bootstrap, since we are interested in direct forecasting.

```r
# Embedding function that creates n_lags of all variables
# of a given data frame
my_embed = function(df, n_lags = 4) {
    Lags = list()
    Lags[[1]] = df %>%
        select(-contains("date"))
    if (n_lags >= 1) {
        for (i in 1:n_lags) {
            Lags[[i + 1]] = df %>%
                select(-contains("date")) %>%
                mutate_all(function(x) lag(x, n = i))
        }
    }
    lagged_data = reduce(Lags, function(x, y) {
        bind_cols(x, y, .name_repair = ~make.unique(.x))
    })

    return(lagged_data)
}
```

```r
n_lags = 4

# Rolling window forecasting
rolling_window <- 492

# Random Forest parameters
p = (1+n_lags)*ncol(data) # number of variables
mtry = ((1/3)*p) %>% round() # number of variables randomly selected
num.trees = 500 # number of trees
min.bucket = 5 # minimal number of observations in each leave (terminal node)
```

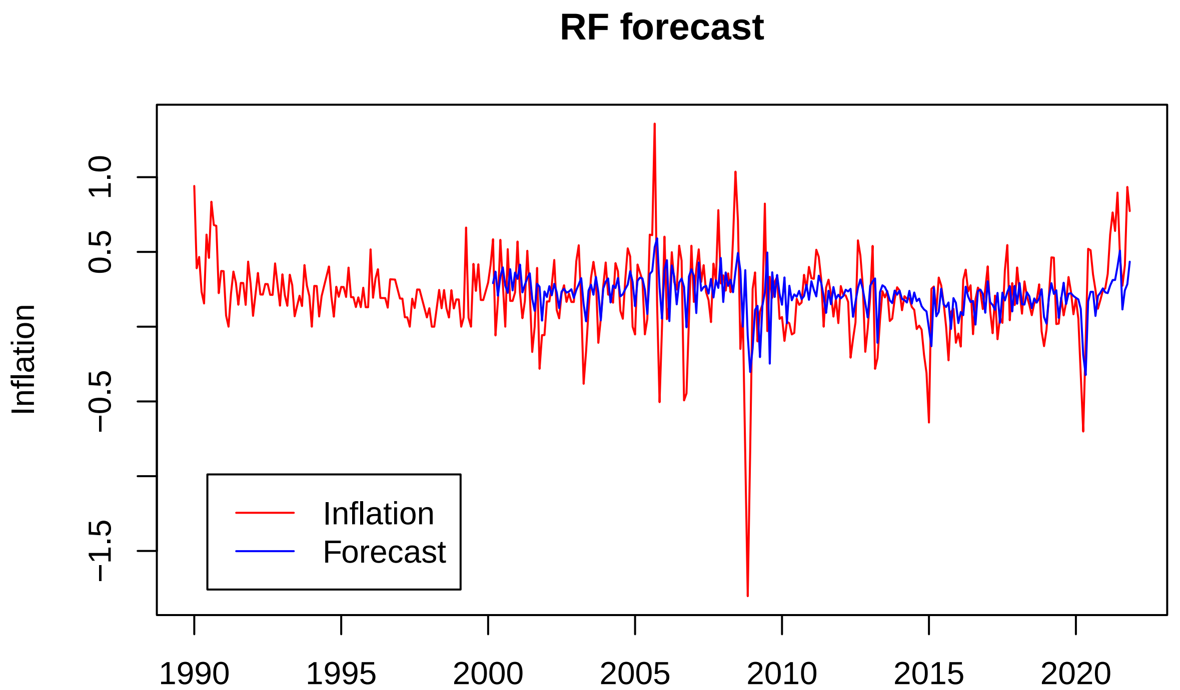
```r
set.seed(1430)

forecast1 = list()

for(a in 1:(length(inflation)-rolling_window)){
  # get the window for training the model
  train = data[a:(a+rolling_window-1), ]
  # embed
  RF_data = my_embed(train, n_lags = n_lags)
  # bind the embeded columns with the one-step-ahead inflation
  RF_data = bind_cols(inflation.ahead = lead(inflation[a:(a+rolling_window-1)]), RF_data)

  # Random forest estimation
  RF = ranger(inflation.ahead ~.,
              data = RF_data %>% na.omit(),
              oob.error = T,
              # Parameters below are set previously
              mtry = mtry,
              num.trees = num.trees,
              min.bucket = min.bucket)

  # Prediction
  new = RF_data %>% select(-inflation.ahead) %>% tail(1)
  forecast1[a] = predict(RF, data = new)
}

forecast1 = forecast1 %>% unlist() %>%
  ts(start = start(inflation)+c(0,rolling_window), frequency =  frequency(inflation) )
```



Using 4 lags of all variables

```
n_lags = 0

# Rolling window forecasting
rolling_window <- 492

# Random Forest parameters
p = (1+n_lags)*ncol(data) # number of variables
mtry = ((1/3)*p) %>% round() # number of variables randomly selected
num.trees = 500 # number of trees
min.bucket = 5 # minimal number of observations in each leave (terminal node)


set.seed(1430)

forecast1 = list()

for(a in 1:(length(inflation)-rolling_window)){
  # get the window for training the model
  train = data[a:(a+rolling_window-1), ]
  # embed
  RF_data = my_embed(train)
  # bind the embeded columns with the one-step-ahead inflation
  RF_data = bind_cols(inflation.ahead = lead(inflation[a:(a+rolling_window-1)]), RF_data)


  # get the window for training the model
  train = data[a:(a+rolling_window-1), ] %>% select(-CPIAUCSL)
  train_cpi = data[a:(a+rolling_window-1), ] %>% select(CPIAUCSL)
```

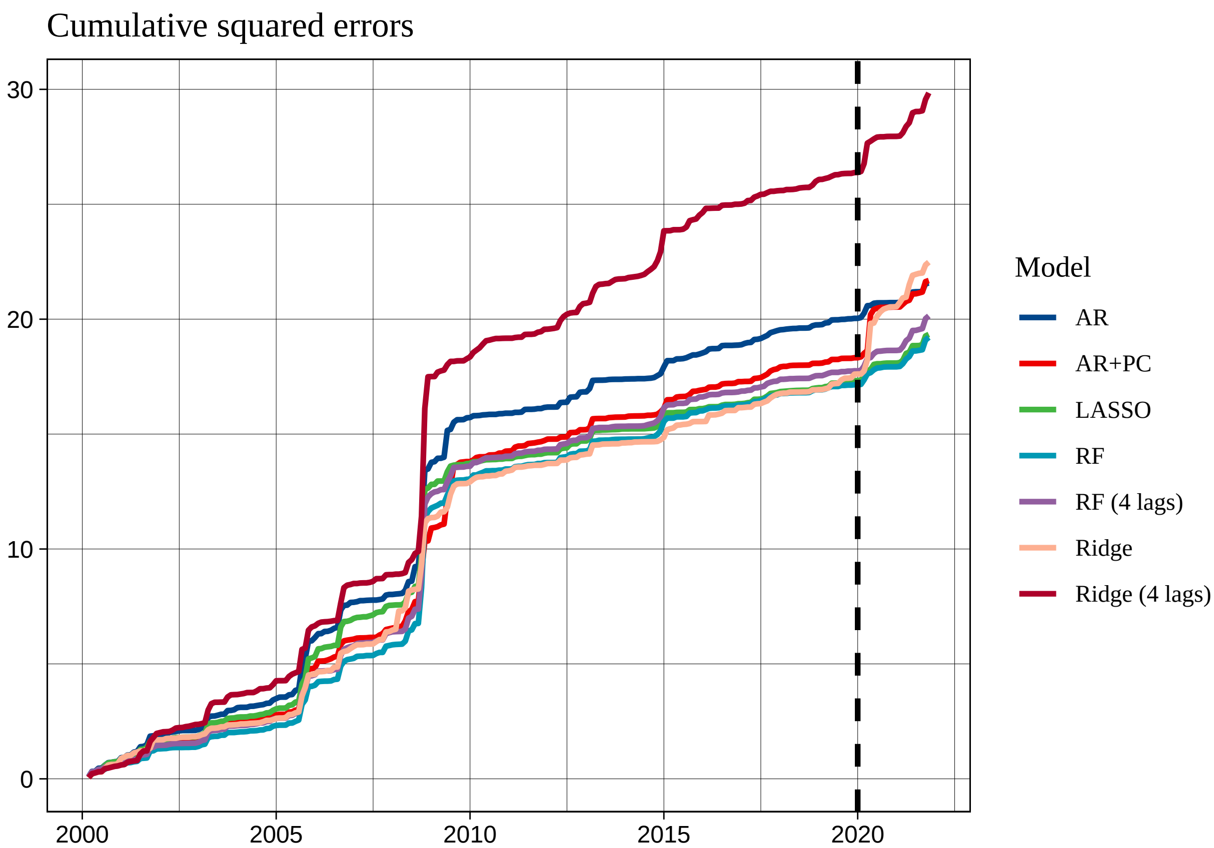
```
  # embed
  RF_data = my_embed(train, n_lags = n_lags)
  cpi_lags = my_embed(train_cpi, n_lags = 4)
  # bind the embeded columns with the one-step-ahead inflation
  RF_data = bind_cols(inflation.ahead = lead(inflation[a:(a+rolling_window-1)]),
                      cpi_lags, RF_data)

  # Random forest estimation
  RF = ranger(inflation.ahead ~.,
              data = RF_data %>% na.omit(),
              oob.error = T,
              # Parameters below are set previously
              mtry = mtry,
              num.trees = num.trees,
              min.bucket = min.bucket)

  # Prediction
  new = RF_data %>% select(-inflation.ahead) %>% tail(1)
  forecast1[a] = predict(RF, data = new)
}

forecast1 = forecast1 %>% unlist() %>%
  ts(start = start(inflation)+c(0,rolling_window), frequency =  frequency(inflation) )
```



Using 4 lags of CPI and no lags of other variables

Item F

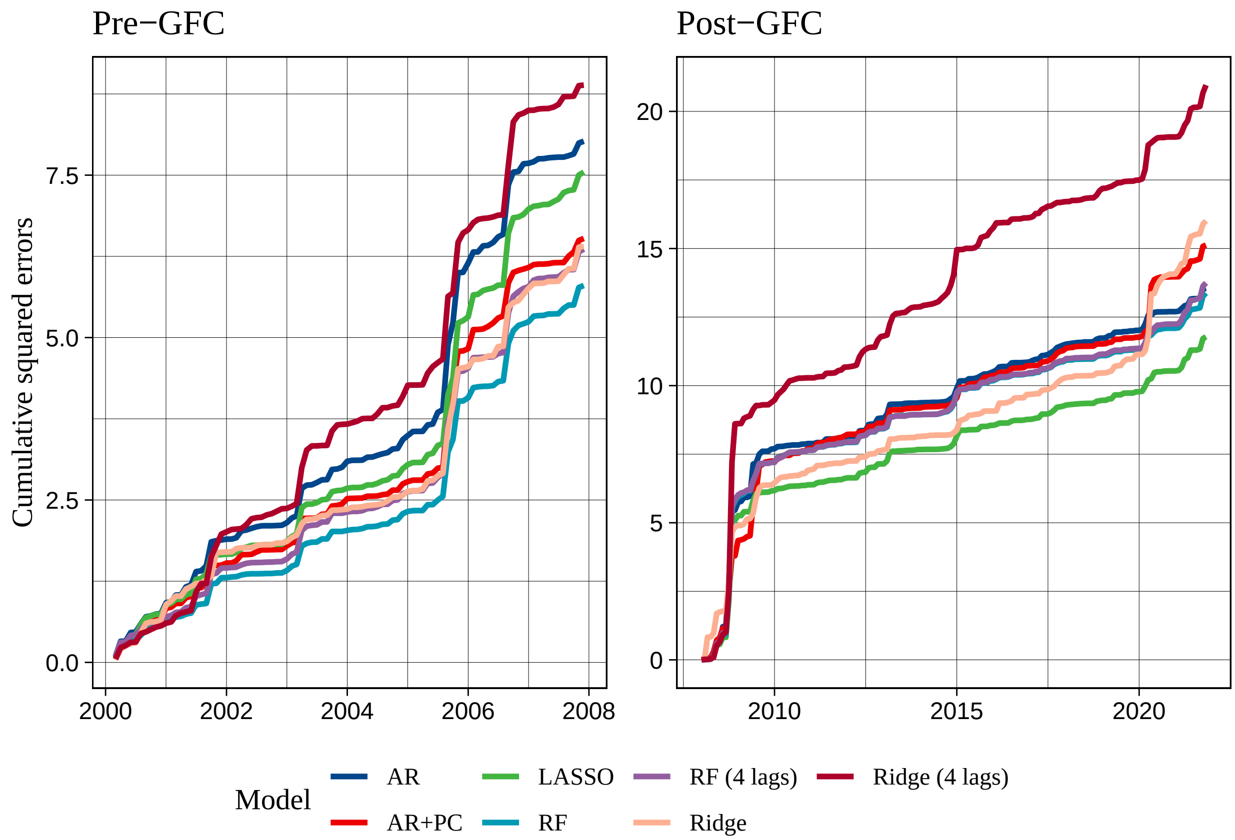
```r
# Forecasting error
error = inflation - forecasts
cum_error = error %>%
    data.frame() %>%
    mutate_all(function(x) {
        (x^2) %>%
            cumsum()
    }) %>%
    bind_cols(date = zoo::as.Date.yearmon(time(error))) %>%
    setNames(c("AR", "AR+PC", "Ridge (4 lags)", "Ridge", "LASSO",
        "RF (4 lags)", "RF", "date"))
```




```r
cum_error %>%
    tail(1) %>%
    select(-date) %>%
    mutate_all(~./tail(cum_error, 1)$AR) %>%
    t() %>%
    data.frame() %>%
    setNames("MSE") %>%
    kbl(digits = 4, booktabs = T)
```

	MSE
AR	1.0000
AR+PC	1.0051
Ridge (4 lags)	1.3836
Ridge	1.0422
LASSO	0.8964
RF (4 lags)	0.9338
RF	0.8905

```
# Plot for subsample
split = 2008

plot_pre = cum_error %>%
    filter(year(date) < split) %>%
    gather(key = model, value = error, -date) %>%
    ggplot(aes(x = date, y = error, color = model)) + geom_line(size = 1) +
    scale_color_lancet() + labs(title = "Pre-GFC", color = "Model",
    y = "Cumulative squared errors", x = NULL)

plot_post = cum_error %>%
    filter(!(year(date) < split)) %>%
    mutate_at(vars(-date), ~. - first(.)) %>%
    gather(key = model, value = error, -date) %>%
    ggplot(aes(x = date, y = error, color = model)) + geom_line(size = 1) +
    scale_color_lancet() + labs(title = "Post-GFC", color = "Model",
    y = NULL, x = NULL)

lemon::grid_arrange_shared_legend(plot_pre, plot_post, ncol = 2)
```

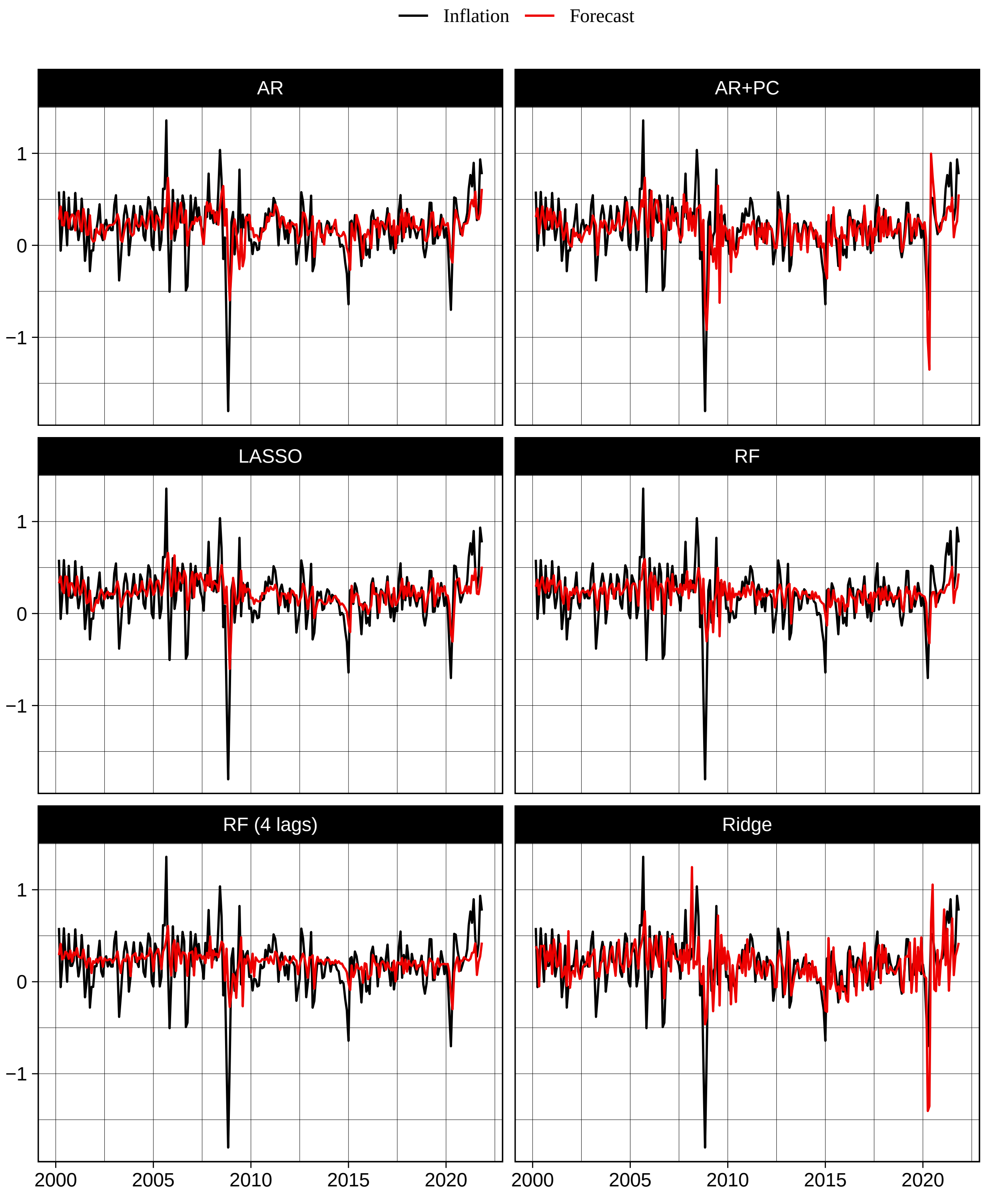



```
# Save
write.csv(forecasts, file = "output/forecasts.csv")
write.csv(cum_error, file = "output/cum_error.csv")
```

Item E

```
forecasts = forecasts %>%
    bind_cols(cum_error$date) %>%
    setNames(c("AR", "AR+PC", "Ridge (4 lags)", "Ridge", "LASSO",
        "RF (4 lags)", "RF", "date"))

# Inflation ts to data.frame
forecasts = data.frame(inflation = inflation) %>%
    bind_cols(date = zoo::as.Date.yearmon(time(inflation))) %>%
    inner_join(forecasts, by = "date")
```

```
# Plot forecasts

forecasts %>%
    select(-`Ridge (4 lags)`) %>%
    gather(key = model, value = pred, -date, -inflation) %>%
    gather(key = type, value = value, -date, -model) %>%
    ggplot(aes(x = date, y = value, color = type)) + geom_line() +
    labs(color = NULL, y = NULL, x = NULL) + scale_color_manual(values = c("black",
    "#ED0000FF"), labels = c("Inflation", "Forecast")) + facet_wrap(~model,
    ncol = 2) + theme(legend.position = "top")
```



Inflation
Forecast
AR
AR+PC
LASSO
RF
RF (4 lags)
Ridge
1
0
−1
2000
2005
2010
2015
2020