VAR Model for Mexico’s GDP Growth and Inflation

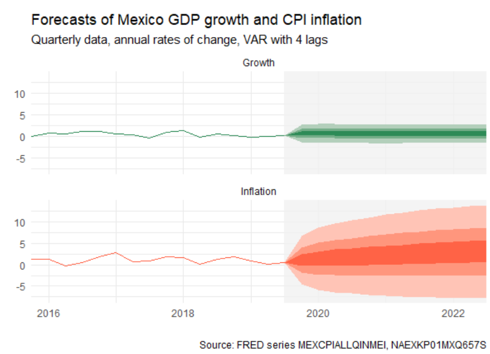
José María Álvarez Silva

02-Nov-2019

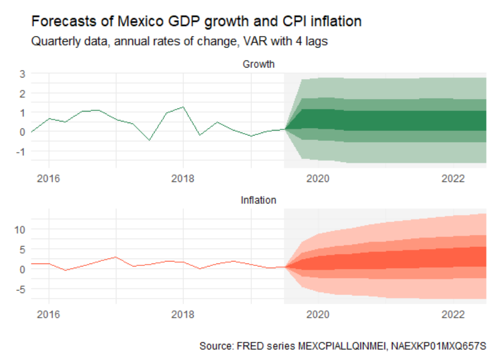
Objectives

Forecast Mexico’s GDP growth and inflation using a Vector auto-regresive or VAR model, generate a fanchart plot just as Central Banks arround the world do as a way of explaining uncertainty . All this work self-contained in chunks of code. This work is based on “Build-your-own fancharts in R” blog post by Andrew Blake.

Conlcusions



Due to the high levels of inflation (compared to GDP growth) of the Mexican economy, the inflation forecast shows a positive trend and wide forecast intervals due to uncertainty. On the other hand, the magnitude of GDP growth is undermined by the magnitude of inflation, which has greater magnitude and greater uncertainty.

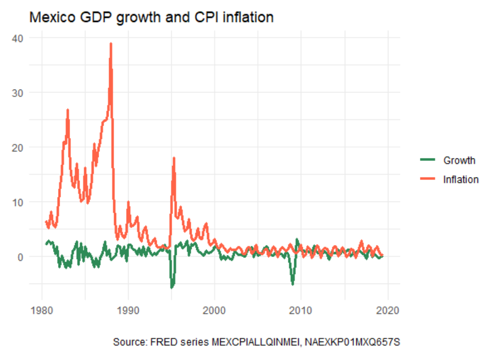


Without the scale, it seems that GDP growth shows little or no change in the forecast, which means that it is expected to have a behavior similar to what it had in the past.

Vector autoregression (VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series.

Excecutive Summary

Exploratory Data Analysis



As you can see in the chart, there have been son past events when inflation had a hike and GDP growth a downturn, having in some cases negative GDP growth. Here are some of the most important events that explain these events.

The macroeconomic policies of the 1970s left Mexico's economy highly vulnerable to external conditions. These turned sharply against Mexico in the early 1980s, and caused the worst recession since the 1930s, with the period known in Mexico as La Década Perdida, "the lost decade", i.e., of economic growth. By mid-1981, Mexico was beset by falling oil prices, higher world interest rates, rising inflation, a chronically overvalued peso, and a deteriorating balance of payments that spurred massive capital flight. This disequilibrium, along with the virtual disappearance of Mexico's international reserves—by the end of 1982 they were insufficient to cover three weeks' imports—forced the government to devalue the peso three times during 1982. The devaluation further fueled inflation and prevented short-term recovery. Cut off from additional credit, the government declared an involuntary moratorium on debt payments in August 1982, and the following month it announced the nationalization of Mexico's private banking system.

Due to the financial crisis that took place in 1982, the total public investment on infrastructure plummeted from 12.5% of GDP to 3.5% in 1989. After rising during the early years of Salinas' presidency, the growth rate of real GDP began to slow during the early 1990s. During 1993 the economy grew by a negligible amount, but growth rebounded to almost 4 percent during 1994, as fiscal and monetary policy were relaxed and foreign investment was bolstered by United States ratification of the North American Free Trade Agreement (NAFTA). The nuevo peso (new peso) was the result of hyperinflation in Mexico. In 1993, President Carlos Salinas de Gortari stripped three zeros from the peso, creating a parity of 1 new peso for 1000 of the old ones.

January 1, 1994, the North American Free Trade Agreement, signed by Mexico, the United States, and Canada went into effect. The collapse of the new peso in December 1994 and the ensuing economic crisis caused the economy to contract by an estimated 7 percent during 1995. Investment and consumption both fell sharply, the latter by some 10 percent.

The latest decline in GDP growth in the graph shows the effect of the global subprime crisis that affected the economy of all countries in the world, showing how connected the different economies are due to globalization.

VAR Model

VAR models generalize the univariate autoregressive model (AR model) by allowing for more than one evolving variable. All variables in a VAR enter the model in the same way: each variable has an equation explaining its evolution based on its own lagged values, the lagged values of the other model variables, and an error term. VAR modeling does not require as much knowledge about the forces influencing a variable as do structural models with simultaneous equations: The only prior knowledge required is a list of variables which can be hypothesized to affect each other intertemporally.

Before a VAR model can estimated some conditions need to checked, in order for it to work and so that we can trust the results. In statistics, the Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive model. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity.

- The DF test for GDP growth

```
        Augmented Dickey-Fuller Test

data:  Data$Growth
Dickey-Fuller = -9.1054, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
```

- The DF test for Inflation

```
        Augmented Dickey-Fuller Test

data:  Data$Inflation
Dickey-Fuller = -4.5122, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
```

Additionaly, the correlation between the variables is aproximately -30%.

Since both variables are stationary and relatively correlated we can proceed with the VAR modelling.

A VAR model is used to explain and forecast GDP growth explained by previous GDP growth and past inflation rates (in this case, the last four; that is, 4 lags); similarly for inflation.

Algebraically a VAR with m lags is:

$$Y_t = \beta_0 + \sum_{i=1}^{m} \beta_i Y_{t-i} + \varepsilon_t$$

where Y_t is a vector of growth and inflation in each period.

	Growth	Inflation
Growth_01	0.214085704	-0.18339705
Growth_02	0.065439664	0.22668919
Growth_03	-0.185920008	-0.11017714
Growth_04	0.005855782	-0.05407754
Inflation_01	-0.064959245	0.82083493
Inflation_02	0.018076456	-0.07978091
Inflation_03	0.009571780	0.07038020
Inflation_04	0.028331103	0.09247628
constant	0.528814757	0.51350587

Using the vars package, the estimated model is a VAR with only one lag for both Growth and Inflation based on information Criteria.

The estimation of the coefficients as estimated before:

	Growth	Inflation
Growth_01	0.28404783	-0.003601816

	Growth	Inflation
Inflation_01	-0.01560747	0.873311805
constant	0.48941839	0.614990402

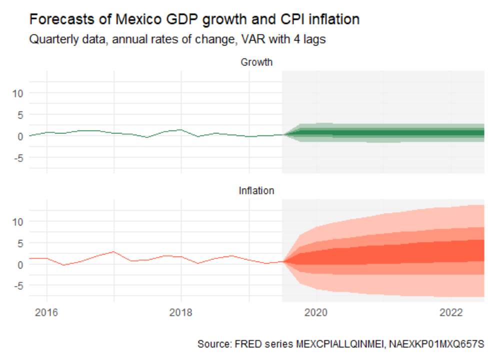
The estimation of the coeficientes with vars package:

	Growth	Inflation
Growth_01	0.28405	-0.003602
Inflation_01	-0.01561	0.873312
constant	0.48942	0.614990

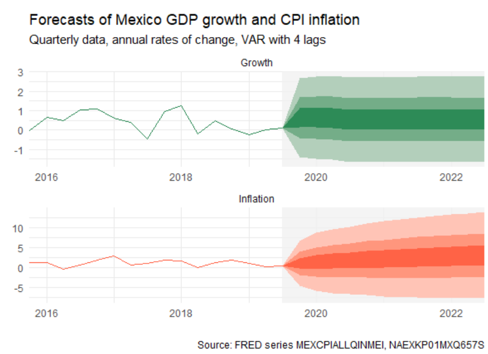
Analysing both cases (i.e.)

Conlcusions

Vector autoregression (VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series.

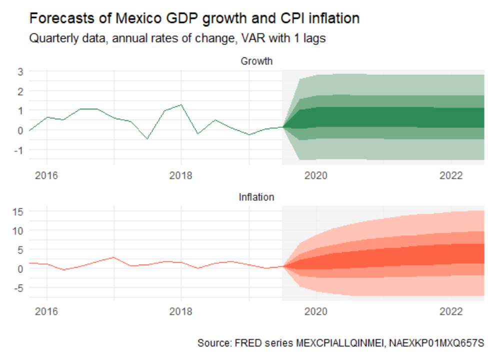


Due to the high levels of inflation (compared to GDP growth) of the Mexican economy, the inflation forecast shows a positive trend and wide forecast intervals due to uncertainty. On the other hand, the magnitude of GDP growth is undermined by the magnitude of inflation, which has greater magnitude and greater uncertainty.

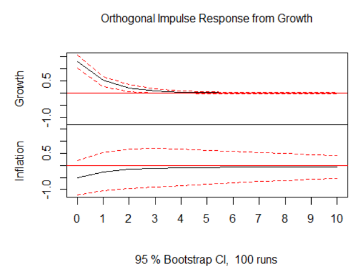


Without the scale, it seems that GDP growth shows little or no change in the forecast, which means that it is expected to have a behavior similar to what it had in the past.

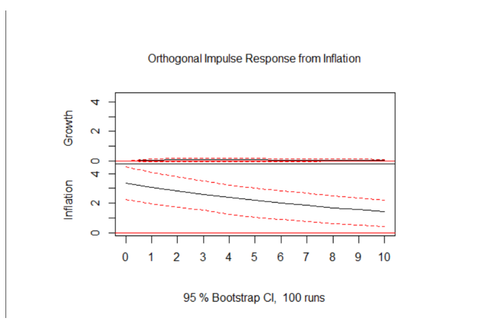
The model with only one lag:



Finaly, both series seem to have some influence on the other series based on the coefficients of the model. But looking closely at the Impulse responses, growth creates a response on itself and on Inflation. The first impulse generates an impulse in the short term that inmediately tends to fall toward the stable level; while the inlfation decreases in the short term tending towards stable level but with a permanent level.



The Impulse responses created by Inflation affects both growth and on Inflation. The first impulse generates its not crearly apreciable (due to scale) but it has a permanent effect on Growth; while the inlfation increases in the short term and continues to be above the stable level with a downward trend.



References

- https://bankunderground.co.uk/2019/11/19/build-your-own-fancharts-in-r/
- https://en.wikipedia.org/wiki/Vector_autoregression
- https://www.stlouisfed.org/
- https://fred.stlouisfed.org/series/MEXCPIALLQINMEI
- https://fred.stlouisfed.org/series/NAEXKP01MXQ657S
- https://en.wikipedia.org/wiki/Dickey%E2%80%93Fuller_test

Anexos

```r
library(tidyverse)
library(quantmod)   # For data access
library(ggplot2)
library(tseries)
```

```r
series = c('MEXCPIALLQINMEI', 'NAEXKP01MXQ657S') # FRED codes for US GDP growth and CPI
CPI    = getSymbols(series[1], src = 'FRED', auto.assign = FALSE)
Growth = getSymbols(series[2], src = 'FRED', auto.assign = FALSE)
```

The next bit of code stores the data series in Data along with the date, calculates the annual inflation rate and then keeps only what's necessary.

```r
Data = inner_join(tibble(Date = time(Growth), Growth = coredata(Growth)),
                  tibble(Date = time(CPI), CPI = coredata(CPI)), by = c("Date")) %>%
  mutate(Inflation = 100 * (CPI/lag(CPI) - 1)) %>%
  select(Date, Growth, Inflation) %>%
  drop_na() # Drop missing obs to balance dataset
```

Stationary

```r
adf.test(Data$Growth, k = 0)
```

```
## Warning in adf.test(Data$Growth, k = 0): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Data$Growth
## Dickey-Fuller = -9.1054, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(Data$Inflation, k = 0)
```

```
## Warning in adf.test(Data$Inflation, k = 0): p-value smaller than printed p-
## value
```

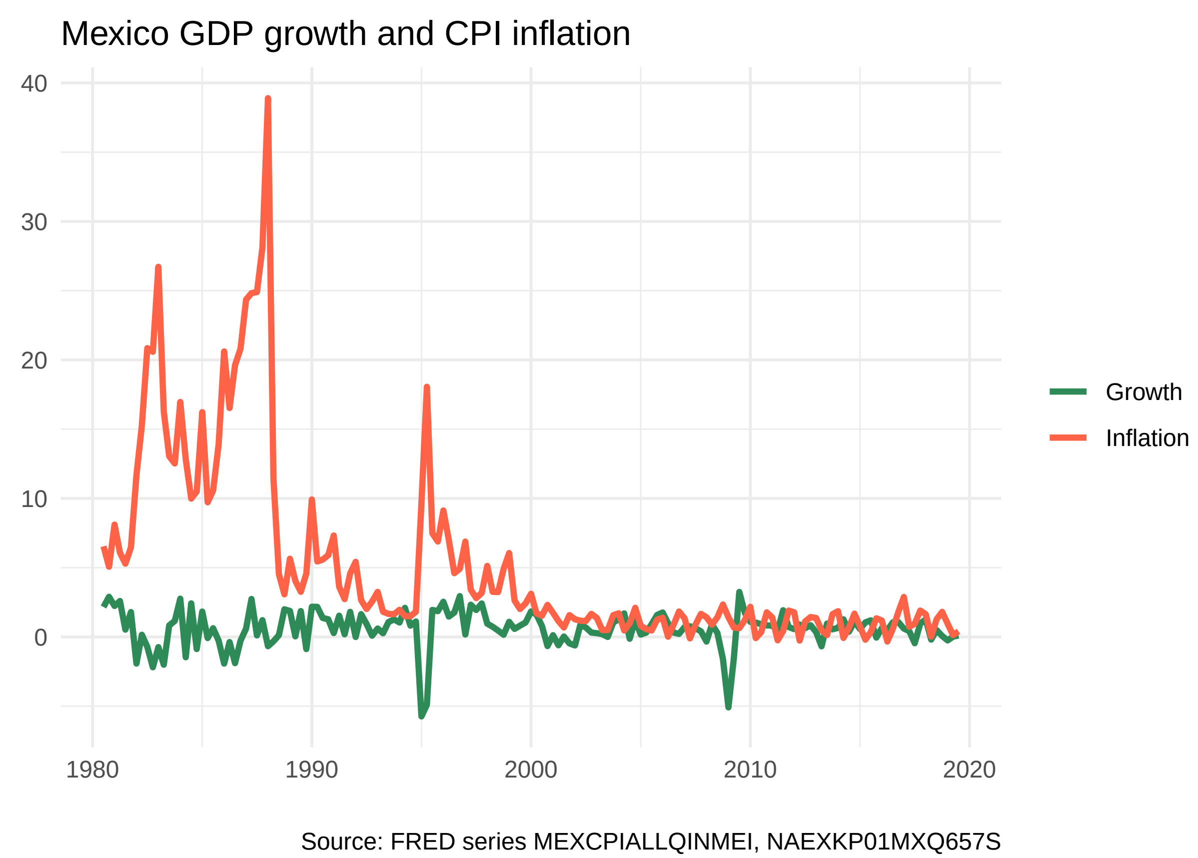
```
##
##  Augmented Dickey-Fuller Test
##
## data:  Data$Inflation
## Dickey-Fuller = -4.5122, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

Correlation

```r
cor(Data$Growth, Data$Inflation)
```

```
##            [,1]
## [1,] -0.2110312
```

```r
centre_colour = c("seagreen","tomato") # Colours for time series/centre of fancharts
tail_colour   = "gray95"               # Colour for the tails, used later but defined here
pivot_longer(Data, cols = -Date, names_to = "Variables", values_to = "Values") %>%
  ggplot() +
  geom_line(aes(x = Date, y = Values, group = Variables, colour = Variables),
            size = 1.1,
            show.legend = TRUE) +
  scale_colour_manual(values = centre_colour) +
  theme_minimal() +
  theme(legend.title = element_blank()) +
  labs(title = "Mexico GDP growth and CPI inflation", x = "", y = "",
       caption = paste0("Source: FRED series ", paste(series, collapse = ", ")))
```



Algebraically a VAR with m lags is:

$$Y_t = \beta_0 + \sum_{i=1}^{m} \beta_i Y_{t-i} + \varepsilon_t$$

where Y_t is a vector of growth and inflation in each period.

```r
m     = 4  # maximum lag in VAR
Datal = Data %>%
   pivot_longer(cols = -Date, names_to = "Names", values_to = "Values") %>%
   mutate(lag_value = list(0:m)) %>%
   unnest(cols = lag_value) %>%
   group_by(Names, lag_value) %>%
   mutate(Values = lag(Values, unique(lag_value))) %>%
   ungroup() %>%
   mutate(Names = if_else(lag_value == 0, Names,                # No suffix at lag 0
   paste0(Names, "_", str_pad(lag_value, 2, pad = "0")))) %>% # All other lags
   select(-lag_value) %>%                                       # Drop the redundant lag index
   pivot_wider(names_from = Names, values_from = Values) %>%
   slice(-c(1:m)) %>%                                           # Remove missing lagged initial values
   mutate(constant = 1)                                         # Add column of ones at end
```

Now select the lagged values (those with a suffix) and constant as explanatory variables and the rest (except for the date) as dependent ones using a regular expression match. These are put in the matrices X and Y respectively.

```r
s = paste(paste0(str_pad(1:m, 2, pad = "0"), "$"), collapse = "|")
X = data.matrix(select(Datal, matches(paste0(s, "|constant"))))
Y = data.matrix(select(Datal, -matches(paste0(s, "|constant|Date"))))
```

The VAR is easy to estimate by solving for the unknown β's using:

```r
(bhat = solve(crossprod(X), crossprod(X,Y)))
```

```
##                    Growth   Inflation
## Growth_01     0.214085704 -0.18339705
## Growth_02     0.065439664  0.22668919
## Growth_03    -0.185920008 -0.11017714
## Growth_04     0.005855782 -0.05407754
## Inflation_01 -0.064959245  0.82083493
## Inflation_02  0.018076456 -0.07978091
## Inflation_03  0.009571780  0.07038020
## Inflation_04  0.028331103  0.09247628
## constant      0.528814757  0.51350587
```

A nice feature of calculating bhat this way is that it automatically labels the output for ready interpretation. An econometrician would spend some time evaluating the statistical model, but let's just press ahead.

Forecast

Simulating the model to calculate the forecasts and the forecast error variances is done in a loop. A first-order representation of the VAR works best, with the small complication that the parameters need to be re-ordered.

```r
nv    = ncol(Y) # Number of variables
nf    = 12      # Periods to forecast
nb    = 16      # Periods of back data to plot, used later
```

```
v     = crossprod(Y - X %*% bhat)/(nrow(Y) - m * nv - 1)              # Calculate error variance
bhat2 = bhat[rep(seq(1, m * nv, m), m) + rep(seq(0, m - 1), each = nv),] # Reorder for simulation
A     = rbind(t(bhat2), diag(1, nv * (m - 1), nv * m))               # First order form - A
B     = diag(1,nv*m,nv)                                               # First order form - B
cnst  = c(t(tail(bhat,1)), rep(0,nv * (m - 1)))                       # First order constants
# Simulation loop
Yf      = matrix(0, nv * m, nf + 1)                 # Stores forecasts
Yf[,1] = c(t(tail(Y, m)[m:1,]))                     # Lagged data
Pf      = matrix(0, nv, nf + 1)                     # Stores variances
P       = matrix(0, nv * m, nv * m)                 # First period state covariance

for (k in 1:nf) {
  Yf[, k + 1] = cnst + A %*% Yf[,k]
  P = A %*% P %*% t(A) + B %*% v %*% t(B)
  Pf[, k + 1] = diag(P)[1:nv]
}
```

Fancharts

There are packages to plot fancharts too. The fanplot package actually has Bank of England fancharts built in but not in the tidyverse, although for the tidy-minded there is ggfan. But it isn't hard to do it from scratch.

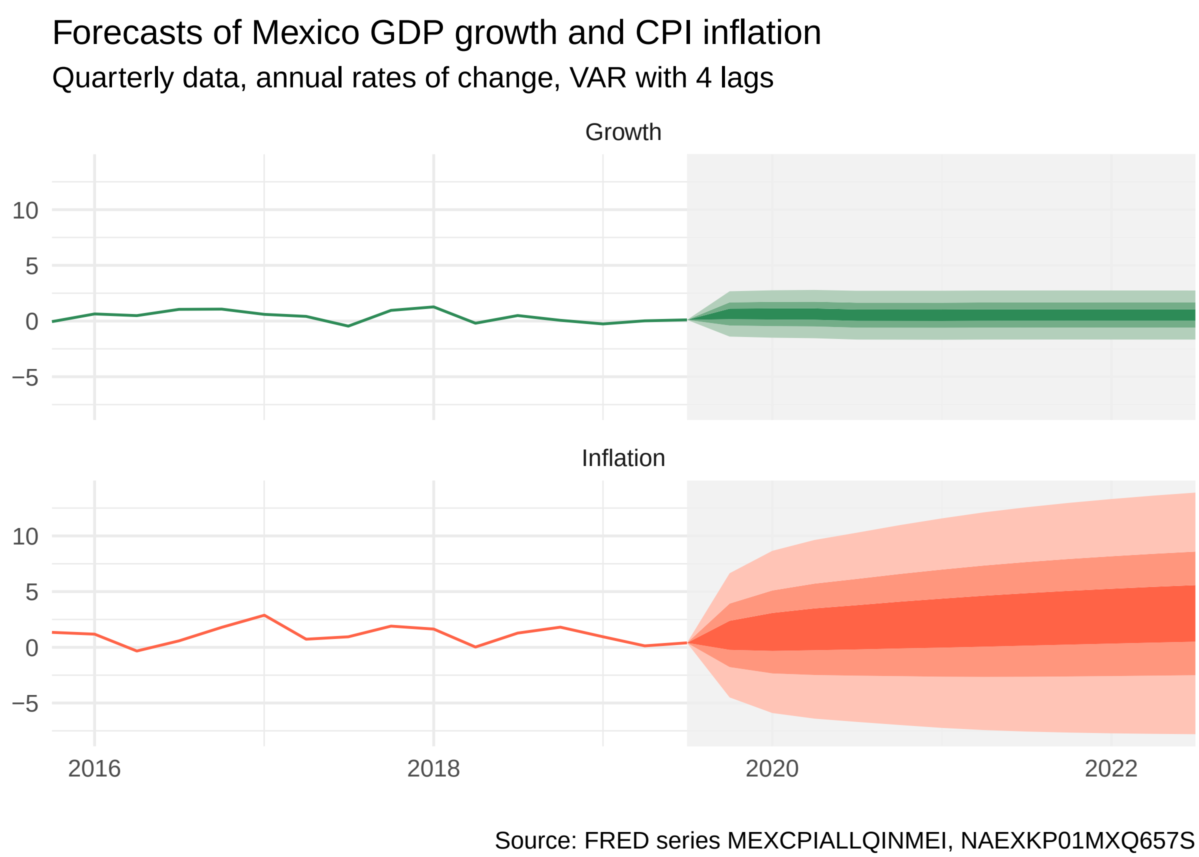
Each forecast fanchart is built up of five shaded areas, with the darkest shade representing an area expected to contain the outcome 30% of the time. Two lighter adjacent areas are the next 15% probability bands below and above the central area, and then another 15% probability bands outside these are shaded lighter still. The edges of these bands are forecast quantiles, evaluated using the values below. Starting at the bottom, each selected forecast quantile is a lower edge of a polygon and next higher quantile the upper edge. The upper coordinates need to be reversed so the perimeter lines join to make the right side of the polygon. Creating this series for each polygon and each variable is done in the code segment below in the curly-bracketed bit `{bind_rows(...)}`. And as everything in a single data frame is convenient, a last step binds in the historical data.

```
qu     = c(.05, .2, .35, .65, .8, .95)  # Chosen quantiles ensures 30% of the distribution each colour
nq     = length(qu)
fdates = seq.Date(tail(Data$Date,1), by = "quarter", length.out = nf + 1) # Forecast dates

forecast_data = tibble(Date     = rep(fdates, 2),
                       Variable = rep(colnames(Data)[-1], each = (nf + 1)),
                       Forecast = c(t(Yf[1:nv,])),
                       Variance = c(t(sqrt(Pf)))) %>%
  bind_cols(map(qu, qnorm, .$Forecast, .$Variance)) %>%          # Calculate quantiles
  select(-c("Forecast", "Variance")) %>%
  {bind_rows(select(., -(nq + 2)),                               # Drop last quantile
             select(., -3) %>%                                   # Drop first quantile
               arrange(Variable, desc(Date)) %>%                 # Reverse order
               rename_at(-(1:2), ~ paste0("V",1:(nq - 1))) )} %>%  # Shift names of reversed ones
  pivot_longer(cols = -c(Date, Variable), names_to = "Area", values_to = "Coordinates") %>%
  unite(VarArea, Variable, Area, remove = FALSE) %>%               # Create variable to index polygons
  bind_rows(pivot_longer(tail(Data,nb), cols = -Date, names_to = "Variable", values_to = "Backdata"), .)
```

That's pretty much it. Shaded rectangles made using geom_rect indicate the forecast region, the filled polygons plotted using geom_polygon define the different bands and historical data is added using geom_line.

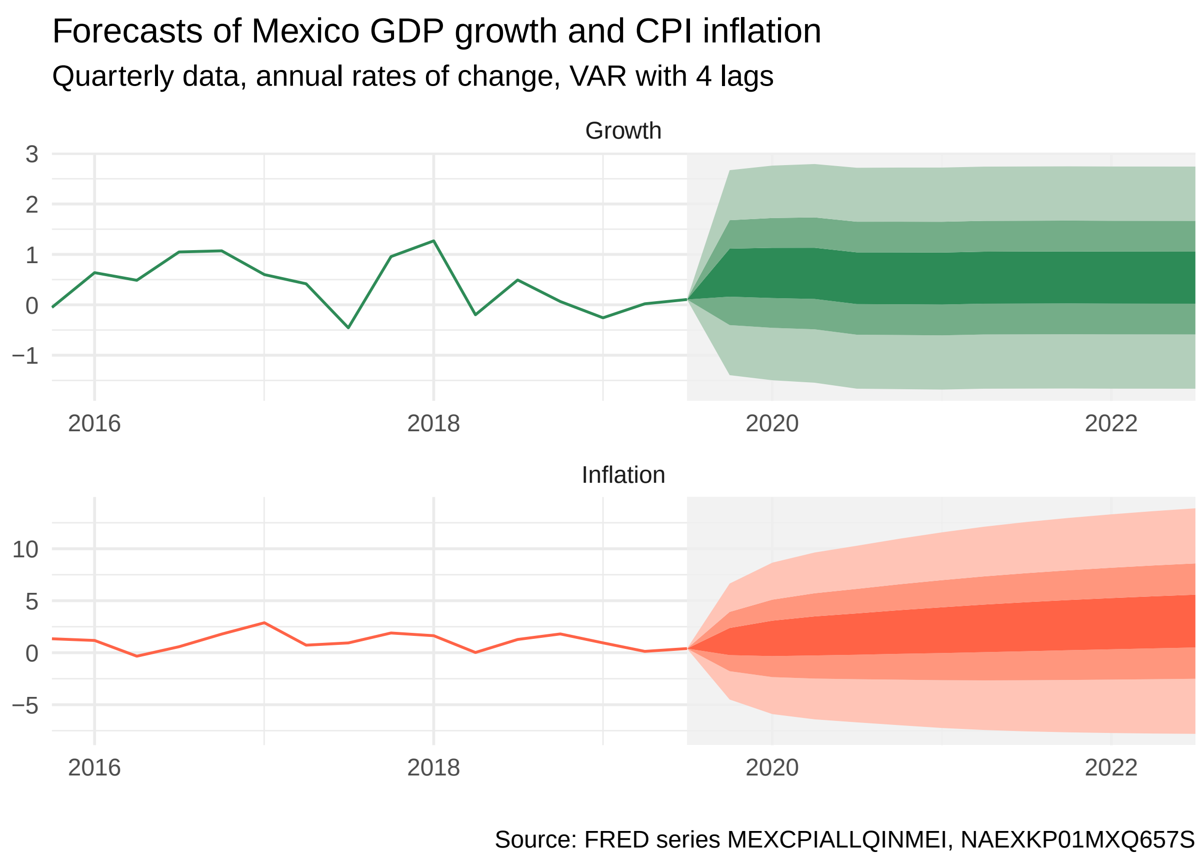
A bit of formatting, apply facet_wrap() and we're done.

```
# Band colours 'ramp' from the centre to the tail colour
band_colours = colorRampPalette(c(rbind(tail_colour, centre_colour), tail_colour),
                                space = "Lab")(nv * nq + 1)[-seq(1, nv * nq + 1, nq)]

ggplot(forecast_data) +
  geom_rect(aes(xmin = Date[nv*nb], xmax = max(Date), ymin = -Inf, ymax = Inf),
            fill = tail_colour,
            alpha = .2) +
  geom_polygon(aes(x = Date, y = Coordinates, group = VarArea, fill = VarArea)) +
  scale_fill_manual(values = band_colours) +
  geom_line(aes(x = Date, y = Backdata, group = Variable, colour = Variable)) +
  scale_colour_manual(values = centre_colour) +
  scale_x_date(expand = c(0,0)) +
  theme_minimal() +
  theme(legend.position = "none") +
  facet_wrap(~ Variable, ncol = 1) +
  labs(title = "Forecasts of Mexico GDP growth and CPI inflation",
       subtitle = paste("Quarterly data, annual rates of change, VAR with", m, "lags"),
       caption = paste("Source: FRED series", paste(series, collapse = ", ")), x = "", y = "")
```




```
# Band colours 'ramp' from the centre to the tail colour
band_colours = colorRampPalette(c(rbind(tail_colour, centre_colour), tail_colour),
                                space = "Lab")(nv * nq + 1)[-seq(1, nv * nq + 1, nq)]
```

```r
ggplot(forecast_data) +
  geom_rect(aes(xmin = Date[nv*nb], xmax = max(Date), ymin = -Inf, ymax = Inf),
            fill = tail_colour,
            alpha = .2) +
  geom_polygon(aes(x = Date, y = Coordinates, group = VarArea, fill = VarArea)) +
  scale_fill_manual(values = band_colours) +
  geom_line(aes(x = Date, y = Backdata, group = Variable, colour = Variable)) +
  scale_colour_manual(values = centre_colour) +
  scale_x_date(expand = c(0,0)) +
  theme_minimal() +
  theme(legend.position = "none") +
  facet_wrap(~ Variable, ncol = 1, scales = "free") +
  labs(title = "Forecasts of Mexico GDP growth and CPI inflation",
       subtitle = paste("Quarterly data, annual rates of change, VAR with", m, "lags"),
       caption = paste("Source: FRED series", paste(series, collapse = ", ")), x = "", y = "")
```



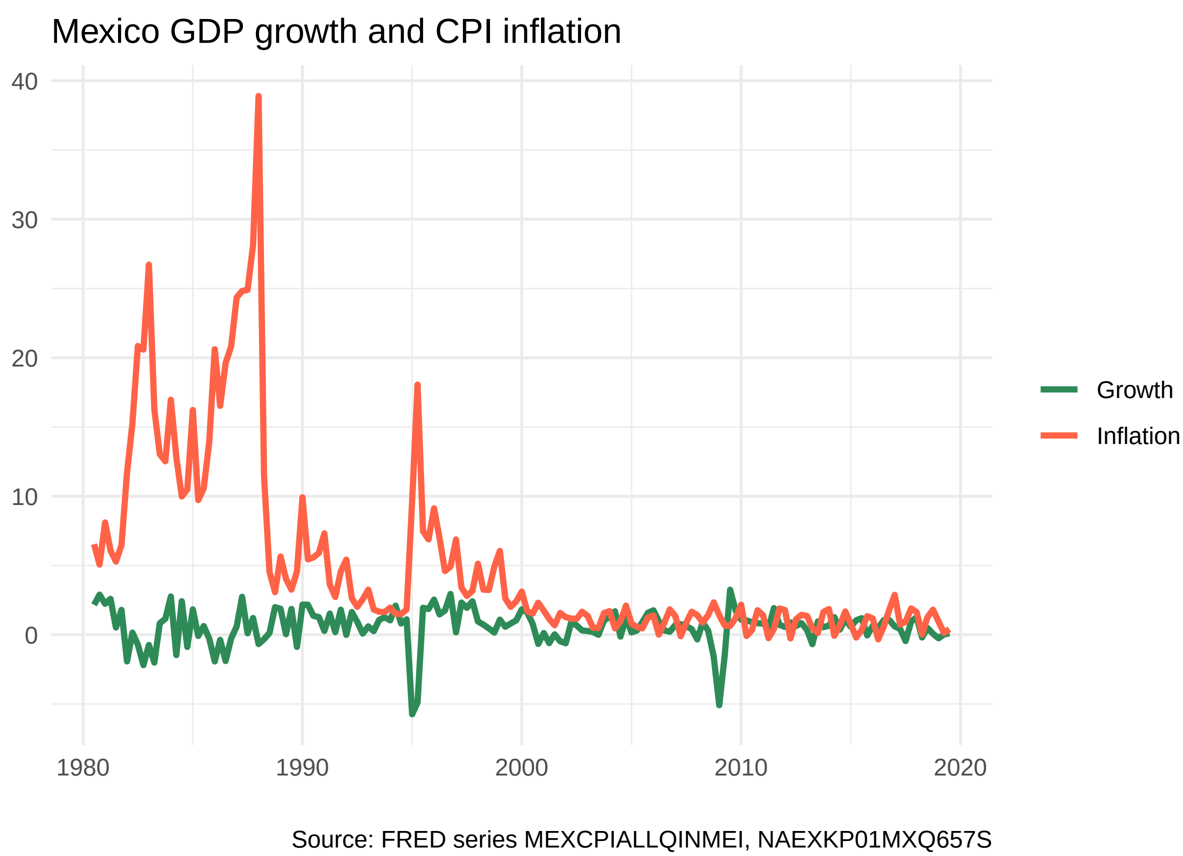
Estimating the VAR model with R packages.

```r
series = c('MEXCPIALLQINMEI', 'NAEXKP01MXQ657S') # FRED codes for US GDP growth and CPI
CPI    = getSymbols(series[1], src = 'FRED', auto.assign = FALSE)
Growth = getSymbols(series[2], src = 'FRED', auto.assign = FALSE)
```

The next bit of code stores the data series in Data along with the date, calculates the annual inflation rate and then keeps only what's necessary.

```
Data = inner_join(tibble(Date = time(Growth), Growth = coredata(Growth)),
                  tibble(Date = time(CPI), CPI = coredata(CPI)), by = c("Date")) %>%
  mutate(Inflation = 100 * (CPI/lag(CPI) - 1)) %>%
  select(Date, Growth, Inflation) %>%
  drop_na() # Drop missing obs to balance dataset
```

```
centre_colour = c("seagreen","tomato") # Colours for time series/centre of fancharts
tail_colour   = "gray95"              # Colour for the tails, used later but defined here
pivot_longer(Data, cols = -Date, names_to = "Variables", values_to = "Values") %>%
  ggplot() +
  geom_line(aes(x = Date, y = Values, group = Variables, colour = Variables),
            size = 1.1,
            show.legend = TRUE) +
  scale_colour_manual(values = centre_colour) +
  theme_minimal() +
  theme(legend.title = element_blank()) +
  labs(title = "Mexico GDP growth and CPI inflation", x = "", y = "",
       caption = paste0("Source: FRED series ", paste(series, collapse = ", ")))
```




```
m     = 1   # maximum lag in VAR
Datal = Data %>%
   pivot_longer(cols = -Date, names_to = "Names", values_to = "Values") %>%
```

```
  mutate(lag_value = list(0:m)) %>%
  unnest(cols = lag_value) %>%
  group_by(Names, lag_value) %>%
  mutate(Values = lag(Values, unique(lag_value))) %>%
  ungroup() %>%
  mutate(Names = if_else(lag_value == 0, Names,              # No suffix at lag 0
  paste0(Names, "_", str_pad(lag_value, 2, pad = "0")))) %>% # All other lags
  select(-lag_value) %>%                                      # Drop the redundant lag index
  pivot_wider(names_from = Names, values_from = Values) %>%
  slice(-c(1:m)) %>%                                          # Remove missing lagged initial values
  mutate(constant = 1)                                        # Add column of ones at end
```

Now select the lagged values (those with a suffix) and constant as explanatory variables and the rest (except for the date) as dependent ones using a regular expression match. These are put in the matrices X and Y respectively.

```
s = paste(paste0(str_pad(1:m, 2, pad = "0"), "$"), collapse = "|")
X = data.matrix(select(Datal, matches(paste0(s, "|constant"))))
Y = data.matrix(select(Datal, -matches(paste0(s, "|constant|Date"))))
```

The VAR is easy to estimate by solving for the unknown β's using:

```
(bhat = solve(crossprod(X), crossprod(X,Y)))
```

```
##                   Growth    Inflation
## Growth_01     0.28404783 -0.003601816
## Inflation_01 -0.01560747  0.873311805
## constant      0.48941839  0.614990402
```

A nice feature of calculating bhat this way is that it automatically labels the output for ready interpretation. An econometrician would spend some time evaluating the statistical model, but let's just press ahead.

Forecast

Simulating the model to calculate the forecasts and the forecast error variances is done in a loop. A first-order representation of the VAR works best, with the small complication that the parameters need to be re-ordered.

```
nv    = ncol(Y) # Number of variables
nf    = 12      # Periods to forecast
nb    = 16      # Periods of back data to plot, used later

v     = crossprod(Y - X %*% bhat)/(nrow(Y) - m * nv - 1)              # Calculate error variance
bhat2 = bhat[rep(seq(1, m * nv, m), m) + rep(seq(0, m - 1), each = nv),] # Reorder for simulation
A     = rbind(t(bhat2), diag(1, nv * (m - 1), nv * m))                # First order form - A
B     = diag(1,nv*m,nv)                                               # First order form - B
cnst  = c(t(tail(bhat,1)), rep(0,nv * (m - 1)))                       # First order constants
# Simulation loop
Yf     = matrix(0, nv * m, nf + 1)                # Stores forecasts
Yf[,1] = c(t(tail(Y, m)[m:1,]))                   # Lagged data
Pf     = matrix(0, nv, nf + 1)                    # Stores variances
```

```
P      = matrix(0, nv * m, nv * m)                # First period state covariance

for (k in 1:nf) {
  Yf[, k + 1] = cnst + A %*% Yf[,k]
  P = A %*% P %*% t(A) + B %*% v %*% t(B)
  Pf[, k + 1] = diag(P)[1:nv]
}
```

Fancharts

There are packages to plot fancharts too. The fanplot package actually has Bank of England fancharts built in but not in the tidyverse, although for the tidy-minded there is ggfan. But it isn't hard to do it from scratch.

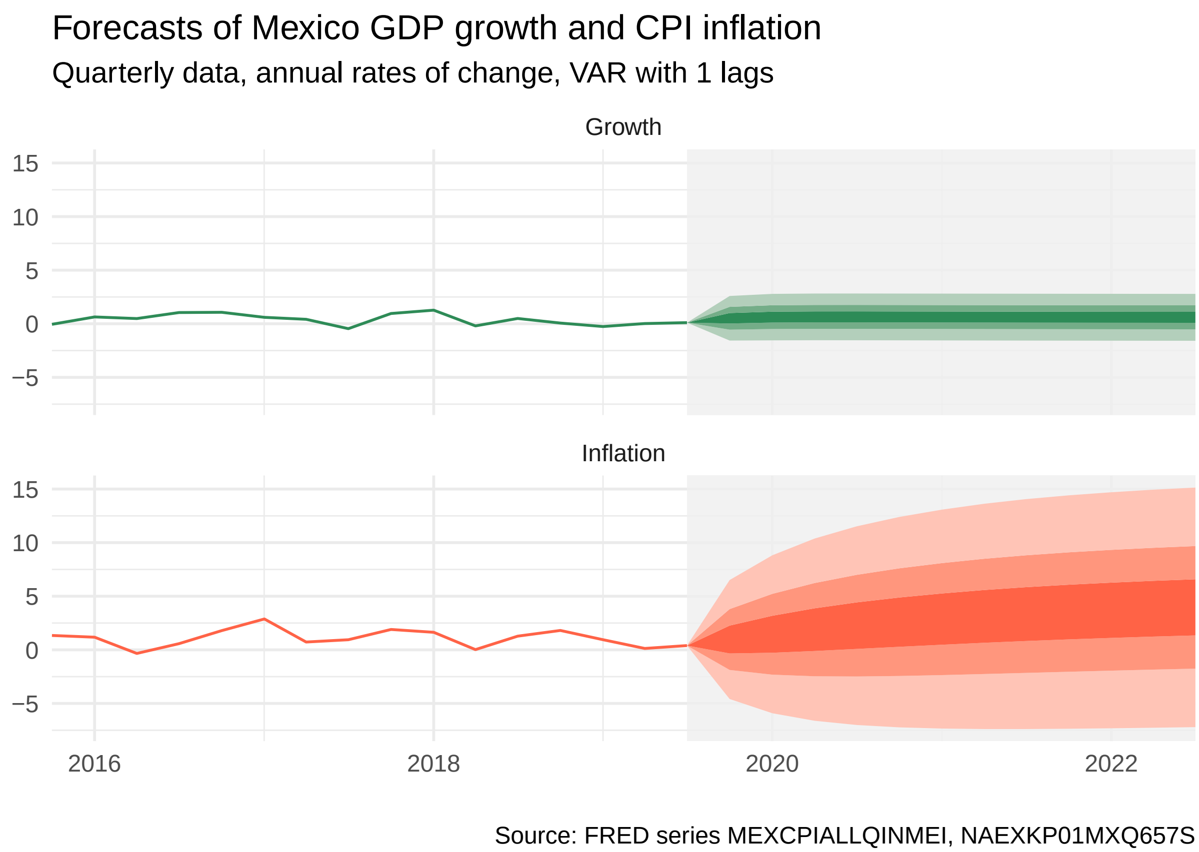
Each forecast fanchart is built up of five shaded areas, with the darkest shade representing an area expected to contain the outcome 30% of the time. Two lighter adjacent areas are the next 15% probability bands below and above the central area, and then another 15% probability bands outside these are shaded lighter still. The edges of these bands are forecast quantiles, evaluated using the values below. Starting at the bottom, each selected forecast quantile is a lower edge of a polygon and next higher quantile the upper edge. The upper coordinates need to be reversed so the perimeter lines join to make the right side of the polygon. Creating this series for each polygon and each variable is done in the code segment below in the curly-bracketed bit `{bind_rows(...)}`. And as everything in a single data frame is convenient, a last step binds in the historical data.

```
qu     = c(.05, .2, .35, .65, .8, .95)  # Chosen quantiles ensures 30% of the distribution each colour
nq     = length(qu)
fdates = seq.Date(tail(Data$Date,1), by = "quarter", length.out = nf + 1) # Forecast dates

forecast_data = tibble(Date     = rep(fdates, 2),
                       Variable = rep(colnames(Data)[-1], each = (nf + 1)),
                       Forecast = c(t(Yf[1:nv,])),
                       Variance = c(t(sqrt(Pf)))) %>%
  bind_cols(map(qu, qnorm, .$Forecast, .$Variance)) %>%         # Calculate quantiles
  select(-c("Forecast", "Variance")) %>%
  {bind_rows(select(., -(nq + 2)),                              # Drop last quantile
             select(., -3) %>%                                  # Drop first quantile
               arrange(Variable, desc(Date)) %>%                # Reverse order
               rename_at(-(1:2), ~ paste0("V",1:(nq - 1))) )} %>%  # Shift names of reversed ones
  pivot_longer(cols = -c(Date, Variable), names_to = "Area", values_to = "Coordinates") %>%
  unite(VarArea, Variable, Area, remove = FALSE) %>%              # Create variable to index polygons
  bind_rows(pivot_longer(tail(Data,nb), cols = -Date, names_to = "Variable", values_to = "Backdata"), .)
```

That's pretty much it. Shaded rectangles made using geom_rect indicate the forecast region, the filled polygons plotted using geom_polygon define the different bands and historical data is added using geom_line. A bit of formatting, apply facet_wrap() and we're done.

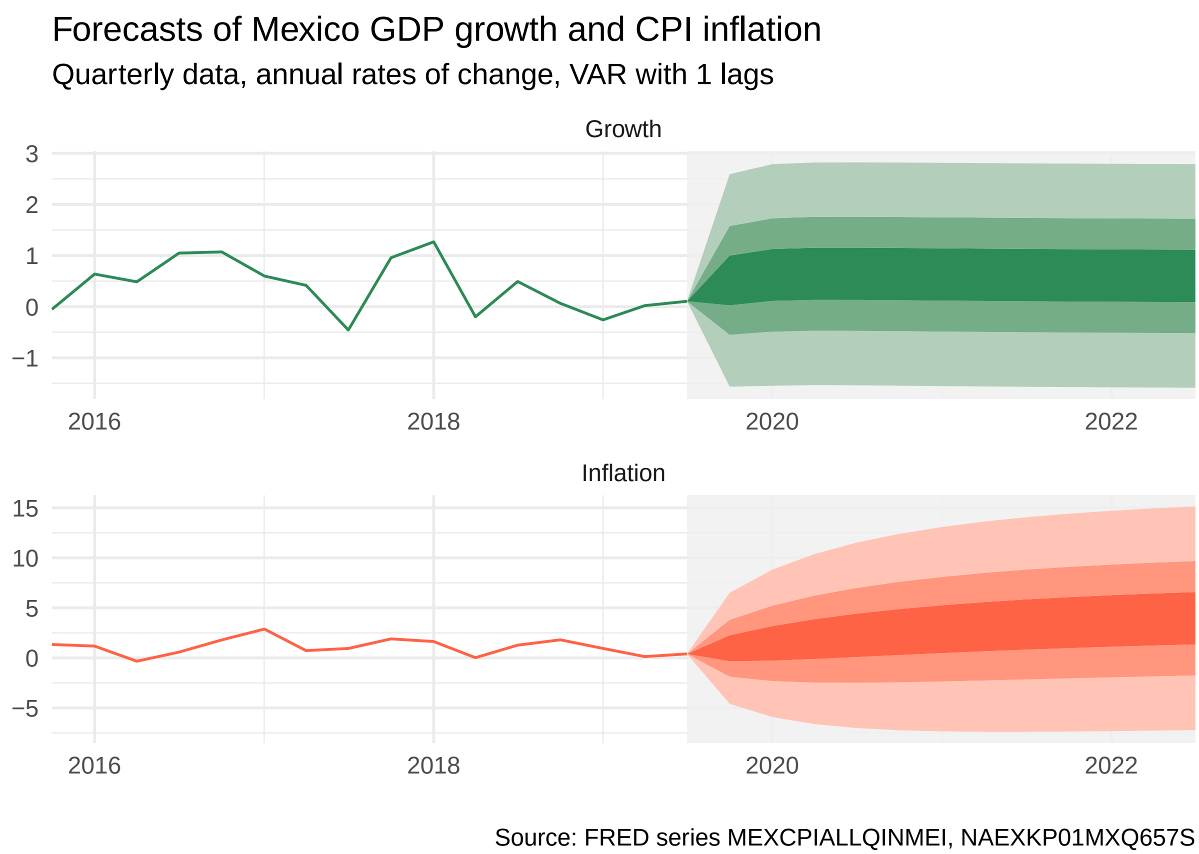
```
# Band colours 'ramp' from the centre to the tail colour
band_colours = colorRampPalette(c(rbind(tail_colour, centre_colour), tail_colour),
                                space = "Lab")(nv * nq + 1)[-seq(1, nv * nq + 1, nq)]

ggplot(forecast_data) +
  geom_rect(aes(xmin = Date[nv*nb], xmax = max(Date), ymin = -Inf, ymax = Inf),
            fill = tail_colour,
```

```
            alpha = .2) +
  geom_polygon(aes(x = Date, y = Coordinates, group = VarArea, fill = VarArea)) +
  scale_fill_manual(values = band_colours) +
  geom_line(aes(x = Date, y = Backdata, group = Variable, colour = Variable)) +
  scale_colour_manual(values = centre_colour) +
  scale_x_date(expand = c(0,0)) +
  theme_minimal() +
  theme(legend.position = "none") +
  facet_wrap(~ Variable, ncol = 1) +
  labs(title = "Forecasts of Mexico GDP growth and CPI inflation",
       subtitle = paste("Quarterly data, annual rates of change, VAR with", m, "lags"),
       caption = paste("Source: FRED series", paste(series, collapse = ", ")), x = "", y = "")
```




```
# Band colours 'ramp' from the centre to the tail colour
band_colours = colorRampPalette(c(rbind(tail_colour, centre_colour), tail_colour),
                                space = "Lab")(nv * nq + 1)[-seq(1, nv * nq + 1, nq)]



ggplot(forecast_data) +
  geom_rect(aes(xmin = Date[nv*nb], xmax = max(Date), ymin = -Inf, ymax = Inf),
            fill = tail_colour,
            alpha = .2) +
  geom_polygon(aes(x = Date, y = Coordinates, group = VarArea, fill = VarArea)) +
  scale_fill_manual(values = band_colours) +
```

```r
  geom_line(aes(x = Date, y = Backdata, group = Variable, colour = Variable)) +
  scale_colour_manual(values = centre_colour) +
  scale_x_date(expand = c(0,0)) +
  theme_minimal() +
  theme(legend.position = "none") +
  facet_wrap(~ Variable, ncol = 1, scales = "free") +
  labs(title = "Forecasts of Mexico GDP growth and CPI inflation",
       subtitle = paste("Quarterly data, annual rates of change, VAR with", m, "lags"),
       caption = paste("Source: FRED series", paste(series, collapse = ", ")), x = "", y = "")
```




```r
library(vars)
```

```r
summary(Data[,c(2,3)])
```

```
##        Growth.V1          Inflation.V1
##  Min.   :-5.740336   Min.   :-0.33420
##  1st Qu.: 0.105385   1st Qu.: 1.17997
##  Median : 0.678469   Median : 1.89976
##  Mean   : 0.586658   Mean   : 5.11812
##  3rd Qu.: 1.268172   3rd Qu.: 6.05004
##  Max.   : 3.246991   Max.   :38.89938
```

```r
vY <- Data[,c(2,3)]
```

```r
#Seleccionar modelo
VARselect(vY)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      1      1      9
##
## $criteria
##                1         2         3         4         5         6
## AIC(n)  2.861967  2.881866  2.832282  2.861489  2.847436  2.840177
## HQ(n)   2.911561  2.964522  2.948000  3.010270  3.029280  3.055083
## SC(n)   2.984025  3.085296  3.117085  3.227665  3.294984  3.369097
## FPE(n) 17.496106 17.848475 16.986621 17.492914 17.253189 17.134660
##                7         8         9        10
## AIC(n)  2.842893  2.800960  2.781957  2.783011
## HQ(n)   3.090861  3.081991  3.096051  3.130167
## SC(n)   3.453185  3.492625  3.554994  3.637420
## FPE(n) 17.189839 16.494702 16.197641 16.231384
```

```r
#estimar
model.var = VAR(vY)
summary(model.var)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: Growth, Inflation
## Deterministic variables: const
## Sample size: 156
## Log Likelihood: -662.929
## Roots of the characteristic polynomial:
## 0.8734 0.284
## Call:
## VAR(y = vY)
##
##
## Estimation results for equation Growth:
## =======================================
## Growth = Growth.l1 + Inflation.l1 + const
##
##              Estimate Std. Error t value Pr(>|t|)
## Growth.l1     0.28405    0.07827   3.629 0.000387 ***
## Inflation.l1 -0.01561    0.01518  -1.028 0.305480
## const         0.48942    0.14147   3.460 0.000701 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.264 on 153 degrees of freedom
## Multiple R-Squared: 0.09771, Adjusted R-squared: 0.08591
```

```
## F-statistic: 8.284 on 2 and 153 DF,  p-value: 0.0003837
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Growth.l1 + Inflation.l1 + const
##
##               Estimate Std. Error t value Pr(>|t|)
## Growth.l1    -0.003602   0.208975  -0.017    0.986
## Inflation.l1  0.873312   0.040526  21.549   <2e-16 ***
## const         0.614990   0.377691   1.628    0.106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.374 on 153 degrees of freedom
## Multiple R-Squared: 0.7608,  Adjusted R-squared: 0.7577
## F-statistic: 243.3 on 2 and 153 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            Growth Inflation
## Growth     1.5971   -0.8255
## Inflation -0.8255   11.3837
##
## Correlation matrix of residuals:
##            Growth Inflation
## Growth     1.0000   -0.1936
## Inflation -0.1936    1.0000
```

```r
model.var1 = VAR(vY,type = "none")
summary(model.var1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: Growth, Inflation
## Deterministic variables: none
## Sample size: 156
## Log Likelihood: -671.355
## Roots of the characteristic polynomial:
## 0.9192 0.3994
## Call:
## VAR(y = vY, type = "none")
##
##
## Estimation results for equation Growth:
## =======================================
## Growth = Growth.l1 + Inflation.l1
##
##              Estimate Std. Error t value Pr(>|t|)
## Growth.l1     0.40428    0.07259   5.569 1.11e-07 ***
## Inflation.l1  0.01705    0.01230   1.386    0.168
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.308 on 154 degrees of freedom
## Multiple R-Squared: 0.1835,  Adjusted R-squared: 0.1729
## F-statistic:  17.3 on 2 and 154 DF,  p-value: 1.664e-07
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Growth.l1 + Inflation.l1
##
##              Estimate Std. Error t value Pr(>|t|)
## Growth.l1      0.1475     0.1883   0.783    0.435
## Inflation.l1   0.9143     0.0319  28.659   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.392 on 154 degrees of freedom
## Multiple R-Squared: 0.8439,  Adjusted R-squared: 0.8419
## F-statistic: 416.4 on 2 and 154 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Growth Inflation
## Growth     1.6473   -0.7439
## Inflation -0.7439   11.4055
##
## Correlation matrix of residuals:
##           Growth Inflation
## Growth     1.0000   -0.1716
## Inflation -0.1716    1.0000
```

```r
#causalidad de granger
causality(model.var1)
```

```
## Warning in causality(model.var1):
## Argument 'cause' has not been specified;
## using first variable in 'x$y' (Growth) as cause variable.
```

```
## $Granger
##
##  Granger causality H0: Growth do not Granger-cause Inflation
##
## data:  VAR object model.var1
## F-Test = 0.61378, df1 = 1, df2 = 308, p-value = 0.434
##
##
## $Instant
##
```

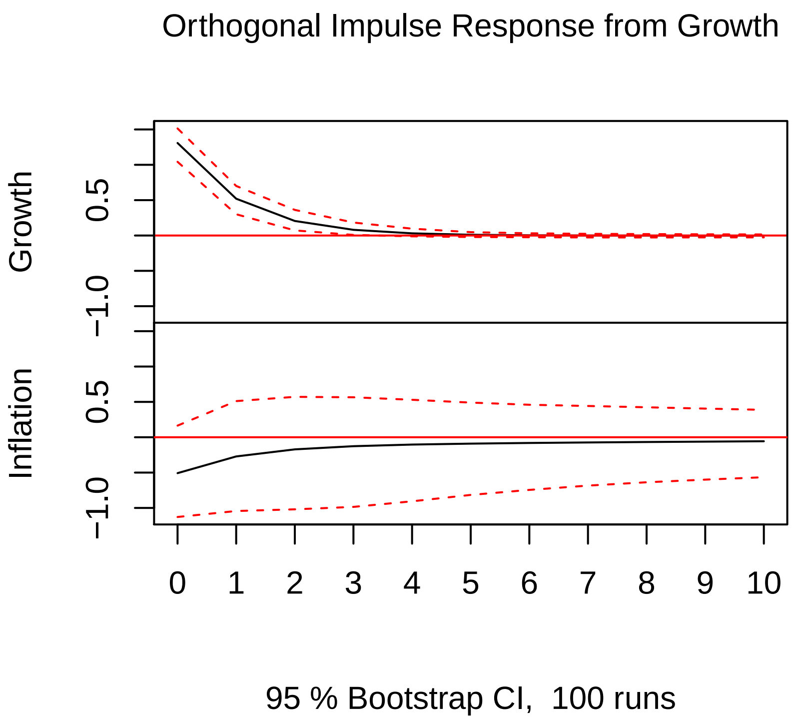
```
##  H0: No instantaneous causality between: Growth and Inflation
##
## data:  VAR object model.var1
## Chi-squared = 3.4192, df = 1, p-value = 0.06444
```

```r
#respuesta al impulso
model.ri = irf(model.var1)
model.ri
```

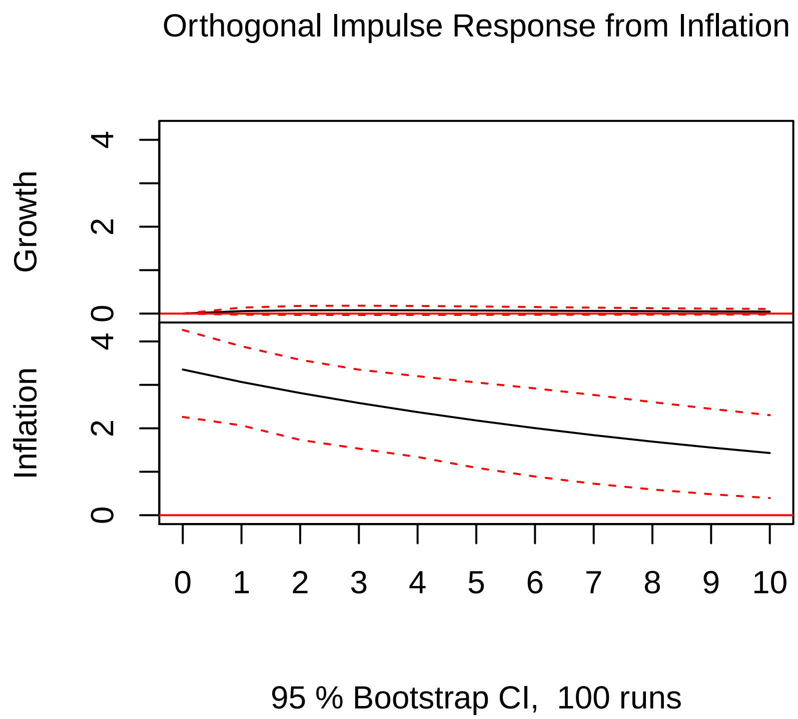
```
##
## Impulse response coefficients
## $Growth
##              Growth   Inflation
##  [1,]  1.307991864 -0.50777304
##  [2,]  0.520137818 -0.27137986
##  [3,]  0.205654214 -0.17142652
##  [4,]  0.080218816 -0.12641423
##  [5,]  0.030275203 -0.10375644
##  [6,]  0.010470310 -0.09040495
##  [7,]  0.002691244 -0.08111783
##  [8,] -0.000295313 -0.07377339
##  [9,] -0.001377475 -0.06749845
## [10,] -0.001707965 -0.06192053
## [11,] -0.001746453 -0.05686909
##
## $Inflation
##            Growth Inflation
##  [1,] 0.00000000  3.353799
##  [2,] 0.05719361  3.066555
##  [3,] 0.07541745  2.812347
##  [4,] 0.07844992  2.582600
##  [5,] 0.07575791  2.372976
##  [6,] 0.07109480  2.180910
##  [7,] 0.06593421  2.004606
##  [8,] 0.06084129  1.842640
##  [9,] 0.05602027  1.693796
## [10,] 0.05153291  1.556989
## [11,] 0.04738573  1.431237
##
##
## Lower Band, CI= 0.95
## $Growth
##              Growth  Inflation
##  [1,]  1.042125337 -1.1277686
##  [2,]  0.300462548 -1.0433507
##  [3,]  0.071736050 -1.0194777
##  [4,]  0.008275706 -0.9851614
##  [5,] -0.011746440 -0.9058714
##  [6,] -0.021421571 -0.8158265
##  [7,] -0.024889861 -0.7448815
##  [8,] -0.027355306 -0.6840603
##  [9,] -0.027895013 -0.6368303
## [10,] -0.026636976 -0.5995868
## [11,] -0.026598676 -0.5654842
```

```
##
## $Inflation
##            Growth Inflation
##  [1,]  0.00000000 2.2621017
##  [2,] -0.02455620 2.0673432
##  [3,] -0.03185476 1.7334290
##  [4,] -0.03294617 1.5320015
##  [5,] -0.03186618 1.3401301
##  [6,] -0.03008988 1.0918442
##  [7,] -0.02815399 0.8898024
##  [8,] -0.02625021 0.7254368
##  [9,] -0.02444319 0.5917080
## [10,] -0.02275523 0.4828730
## [11,] -0.02120613 0.3942616
##
##
## Upper Band, CI= 0.95
## $Growth
##           Growth Inflation
##  [1,] 1.51383868 0.1644403
##  [2,] 0.70072848 0.5110344
##  [3,] 0.36193505 0.5712013
##  [4,] 0.18306929 0.5651139
##  [5,] 0.09645588 0.5295597
##  [6,] 0.04747967 0.4900946
##  [7,] 0.03119125 0.4596363
##  [8,] 0.02336275 0.4420129
##  [9,] 0.01965505 0.4234839
## [10,] 0.01695164 0.4050508
## [11,] 0.01481646 0.3875273
##
## $Inflation
##          Growth Inflation
##  [1,] 0.0000000  4.264260
##  [2,] 0.1383832  3.888342
##  [3,] 0.1768409  3.576737
##  [4,] 0.1810929  3.350707
##  [5,] 0.1757916  3.200100
##  [6,] 0.1649273  3.056396
##  [7,] 0.1505229  2.919327
##  [8,] 0.1369786  2.768605
##  [9,] 0.1249332  2.603651
## [10,] 0.1143484  2.448536
## [11,] 0.1073953  2.302696
```

```r
plot(model.ri)
```



Orthogonal Impulse Response from Growth
Growth
Inflation
0.5
−1.0
0.5
−1.0
0
1
2
3
4
5
6
7
8
9
10
95 % Bootstrap CI, 100 runs




```r
##prediccion
predict(model.var1, n.ahead = 8, ci = 0.95)
```

```
## $Growth
##                fcst     lower    upper       CI
## [1,] 0.049403121 -2.514214 2.613020 2.563617
## [2,] 0.026453560 -2.734702 2.787609 2.761155
## [3,] 0.016744662 -2.777588 2.811078 2.794333
## [4,] 0.012367901 -2.790605 2.815341 2.802973
## [5,] 0.010161087 -2.797369 2.817692 2.807531
## [6,] 0.008857995 -2.802203 2.819919 2.811061
## [7,] 0.007949904 -2.806085 2.821985 2.814035
## [8,] 0.007230883 -2.809330 2.823791 2.816561
##
## $Inflation
##              fcst      lower     upper        CI
## [1,] 0.3800294  -6.268208  7.028267  6.648237
## [2,] 0.3547669  -8.623324  9.332858  8.978091
## [3,] 0.3282834 -10.212222 10.868788 10.540505
## [4,] 0.3026363 -11.392896 11.998168 11.695532
## [5,] 0.2785403 -12.309476 12.866556 12.588016
## [6,] 0.2561826 -13.038964 13.551330 13.295147
## [7,] 0.2355476 -13.628900 14.099995 13.864447
## [8,] 0.2165460 -14.111287 14.544379 14.327833
```