Quantiles, Networks, Time

andrew p blake

2024-01-01

Table of contents

Genesis

This book is about economics and central banking. It is the product of an initiative taken by the Bank of England in the early 1990s. It was decided that the Bank should create a forum where the central bankers of the world could gather to commune, discuss, and above all, learn together. Central banking had long been associated with grey men in grey suits, pondering deeply the impenetrable decisions of high finance, fuelled by cigar smoke and mystique. In truth, this Capraesque view was substantially out of date. Central bankers were no longer monochrome, or exclusively male, and needed skills their forebears could have barely imagined – and had most decidedly different priorities.

The timing, of course, was not incidental, and the initiative turned out to be a prescient one. This was at a major historical turning point, one that signalled a burgeoning new world order, as the Iron Curtain crumbled, the European experiment gathered momentum, and industrial might continued an inexorable shift eastwards. Economic policy had shifted too. Monetary policy was beginning a new and – as it turned out – lasting phase. There was indeed much to learn, and so the Centre for Central Banking Studies was instituted. More than thirty years later it remains a key forum for learning, discussion and networking, just as intended. Activities have evolved to include many more of the disparate areas of responsibility that now involve the central banking community, and with a truly global reach.

Various histories of the CCBS exist, e.g. Hammond (2006).

Placeholders abound.

Figure 1: Puppet, Yogyakarta

Figure 2: Abu Dhabi, United Arab Emirates

Figure 3: Villa Sterne, Pretoria

Figure 4: Insadong, Seoul

Figure 5: Old Town, Montevideo

Part I

Methods matter

Chapter 1

Introduction

This is a book created using Quarto and includes all examples as executable code.

See Andrew P. Blake and Mumtaz (2017).

1.1 Reading is good for you

For me, the best (although slightly dated) text is Hastie, Tibshirani, and Friedman (2009) The Elements of Statistical Learning and the best source for the mathematics, with an easy-reading version by some of the same authors James et al. (2021) Introduction to Statistical Learning.

I also rather like Boehmke and Greenwell (2019) Hands-On Machine Learning with R which is something of a cookbook rather than a technical manual but with wide scope. Taddy (2019) is more elementary.

On text, just read Silge and Robinson (2017) Text Mining with R: A Tidy Approach and then Hvitfeldt and Silge (2021) Supervised Machine Learning for Text Analysis in R. That's it.

Two books I would solidly recommend to make us all into better statisticians and not just econometricians are Gelman, Hill, and Vehtari (2019) Regression and Other Stories, and McElreath (2020) Statistical Rethinking.

Chapter 2

Non-econometric methods for econometricians

2.1 Selected ML and Dataviz techniques in R

Econometricians are used to handling data, performing analysis and reporting results. But somewhere along the line data became big and unstructured, analysis was now machines learning about something and outputs became visualisations.

This online seminar takes some big(ish) datasets, sets the machines on them and draws some great graphs. If you ever wondered what use a tree was for forecasting or why everything is a network (including neural ones), or wanted to draw a map with your house in it, or to understand a document without the bother of reading it (or a few other things besides) then you might find something to interest you. All done in R.

Specifically, this seminar is designed to introduce some of the key methods used outside of econometrics that econometricians will find very useful in their work in a central bank. This includes some important machine learning techniques as a gateway to others, particularly tree-based methods and neural networks, as well as text processing and map making. All the way through there is an emphasis on the network properties of many of these techniques. We make extensive use of the `tidyverse`, including `ggplot2` and `tidytext`, and a number of statistics, machine learning, geographical data and other packages.

The framework for each day is the following:

- Each day is divided into two two-hour sessions starting at 10.30 am and 2.00 pm GMT.

- The first hour of each will be an online presentation covering a particular topic (or topics) with a look at both techniques and code.
- After a quick break the second hour will be largely devoted to the code itself or resources to understand how to code the material.

We may run polls during the event to prioritize the topics covered in the webinars as it is not expected that everyone will be able to try out everything.

2.1.1 The code

All code and some of the data will be made available through the Juno portal. For each presentation the .Rmd (R markdown) file is supplied that creates the presentation, an HTML file of the presentation for you to step through which can be re-created from the .Rmd file, and a further .R file of the code that we use. Some additional code and data is included, including links to a number of videos that cover some additional aspects both in this file and in the presentations.

Some data will need to be downloaded from original other sites if all the examples are to be followed. All code is additionally available at https://github.com/andrewpeterblake/R2020 or https://github.com/andrewpeterblake/R2021 or through the QR codes below.

Figure 2.1: 2020 GitHub repositories for historic CCBS courses

Figure 2.2: 2021

2.2 HOW TO ENSURE RSTUDIO FINDS THE CODE

To use the code, in particular so that R Studio finds the data files etc, create a directory for each topic, (e.g. Trees, ANN etc) and copy the contents from the zip file or GitHub. Then create a new project in R Studio that uses that directory as its home directory, using "File/New Project" in the drop down menu. Opening files within a project sets the home directory to that directory, so everything (including the sub-directories) can be found.

2.3 Typical program structure

2.3.1 Day 1: Trees and maps

2.3.1.1 Trees

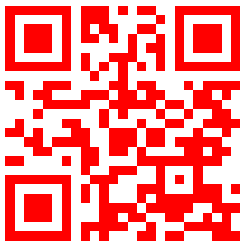
- Classification and regression trees
- Econometrics strikes back: Bootstrap/bagging and Boosting/Model selection
- Random forests
- Visualising decision trees
- Use example: House prices

The presentations for this are `Trees.html` and `LondonHP.html`; The two programs `TreeCancer.R` and `TreeNW.R` are the use examples.

2.3.1.2 Maps

- How to draw a map in R
- A guide to some resources
- Choropleths
- Use examples: Climate change, regional data, postcode wrangling

The presentation for this is `MapAER.html` (see also `Weatherpretty.html`); The program `MapAERcode.R` is the main map drawing code, I've included `ZAF.R` as as short simple way and source for two countries, and the directory `Trendz` contains the program (`app.R`) and data for the weather example.



I've included an additional video (red QR code) for more about Shiny. This uses unemployment data from the Survey of Professional Forecasters. The code we look at is for climate change data World Bank data.

A comprehensive treatment of maps is Lovelace, Nowosad, and Muenchow (2019) *Geocomputation in R*, but it is quite a lot to assimilate all at once.

2.3.2 Day 2: Networks

2.3.2.1 Neural networks

- What is an ANN? Deep learning?
- Function approximation via a network
- Data: fit, validate, test
- Network architecture
- Use examples: House prices revisited

The presentation for this is `IntroANN.html`; The program `ANN.R` replicates the ANN estimation. The data used is the same as for Day 1.

2.3.2.2 Networks (real ones)

- DAGs and ANNs as network graphs
- Incidence matrices
- Measuring connectivity: Degree and betweenness
- Plotting with `igraph`
- Use examples: Industry inter-relationships

Figure 2.3: Coding club Network examples

Figure 2.4: R-Bloggers article

The presentation used for the first part of this is `DAG.html` and the program `Draw_DAG_ANN.R` draws the ANN examples from Day 2 Session 1 as well as some of the DAG examples. The example is modified from Cunningham (2021) *Causal Inference: The Mixtape*, which is a great read with R code. The pdf `HandShake3.pdf` is the source of the director network graphs, and `Graph101a.R` is a subset of the analytical work on the corruption data set as described in the post *Graph Theory 101* (purple QR code), which is the work of Marina Medina (blue QR code link to presentation site).

2.3.3 Day 3: Text

Text modelling, a 'tidytext' approach.

2.3.3.1 Session 1

- Data cleaning
- Sentiment
- Topic modelling

2.3.3.2 Session 2

- Parts-of-speech tagging
- Text regression
- Use examples: Central bank minutes, reports

Chapter 3

Kalman filtering

Estimating unobserved components

3.1 Literature

Alternatives to what follows can be found in Harvey (1989), Hamilton (1994), Kim and Nelson (1999), Durbin and Koopman (2001) or Triantafyllopoulos (2021). There are many books devoted to the Kalman filter as a casual Amazon search mostly from an engineering perspective. However econometricians since Harvey and Pierse (1984) have used it in a way somewhat different from standard engineering applications. We will cover the filter and then look at a simple example if filtering, then develop a maximum likelihood estimation approach.

3.2 Reminder of a state-space model

Consider the trend-cycle model

$$y_t = \chi_t + \tau_t + \varepsilon_t \tag{3.1}$$

where the cycle equation is

$$\chi_t = c + \rho_1 \chi_{t-1} + \rho_2 \chi_{t-2} + v_{1t} \tag{3.2}$$

and the trend equation is

$$\tau_t = \tau_{t-1} + v_{2t} \tag{3.3}$$

In state space this can be written

$$y_t = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \chi_t \\ \chi_{t-1} \\ \tau_t \end{bmatrix} + [1]\varepsilon_t \tag{3.4}$$

$$\begin{bmatrix} \chi_t \\ \chi_{t-1} \\ \tau_t \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \rho_1 & \rho_2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \chi_{t-1} \\ \chi_{t-2} \\ \tau_{t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{1t} \\ v_{2t} \end{bmatrix} \tag{3.5}$$

3.2.1 A useful class of models

We need a framework that nests this type of model (and many more). State-space models are one such framework, amenable to classical (and Bayesian) estimation. Quite a lot of apparatus required before we can apply maximum likelihood.

Key points:

- Many quantities routinely used to build models and analyse policy are *unobservable*.
- Econometricians face a major problem estimating such models: everything unobserved needs estimating simultaneously (states and parameters).
- Fortunately there is a method we can use: the ***Kalman filter*** (Kalman (1960)).
- Has the useful spin-off that we can also use it to calculate the value of the likelihood function.

3.3 What is a filter?

Imagine we had a time-varying parameter model where we estimate β_t; this becomes a time series so we have a lot of parameters to estimate: we outline an estimation method here, which we will then characterise as the *Kalman Filter*.

Simple observation and transition equations

$$y_t = H_t\beta_t + e_t, \qquad var(e_t) = R \tag{3.6}$$

$$\beta_t = \mu + F\beta_{t-1} + v_t, \qquad var(v_t) = Q \tag{3.7}$$

where β_t is a vector of stochastic variables, y_t a vector of measurements and the data forms an information set such that $\psi_T = \{y_T, y_{T-1}, ..., y_1\}$.

Jazwinski (1970) defines three type of estimation problem

- **Smoothing** is the problem of estimating β_k for any $k < T$

- **Filtering** is the problem of estimating β_k for $k = T$
- **Prediction** is the problem of estimating β_k for any $k > T$

> "The object of filtering is to update our knowledge of the system each time a new observation y_t is brought in." (Durbin and Koopman (2001))

Filtering is specifically this: perhaps we have some estimates already of β_t for $t = 1...t-1$, then given the new period-t observation of y_t how should we estimate a new value of β_t? This immediately implies a recursive structure to estimation problems, consistent with on-line (or real-time) estimation. The Kalman filter uses the period t news available from observed y_t to update our estimates of β_t using the regression lemma.

3.3.1 Forecasting

Consider the simple first-order VAR model

$$\beta_t = F\beta_{t-1} + v_t, \quad v_t \sim N(0, Q) \tag{3.8}$$

We can use to make the *conditional* forecast

$$\beta_{t|t-1} = F\beta_{t-1} \tag{3.9}$$

where $\beta_{t|t-1} = E[\beta_t|\psi_{t-1}]$ and ψ_{t-1} is the information set available at time $t-1$.

If ψ_{t-1} includes β_{t-1} we can straightforwardly forecast next period (and the next-but-one period etc) using the model. This is a standard forecasting exercise given any estimated (or even calibrated) economic model.

3.3.2 Uncertainty

How can we assess the associated forecast uncertainty? The forecast covariance $P_{t|t-1} = var(\beta_t|\psi_{t-1})$ is given by

$$\begin{aligned} P_{t|t-1} &= E\left[(\beta_t - \beta_{t|t-1})(\beta_t - \beta_{t|t-1})'\right] && (3.10)\\ &= E\left[(F\beta_{t-1} + v_t - F\beta_{t-1|t-1})\left(\beta'_{t-1}F' + v'_t - \beta'_{t-1|t-1}F'\right)\right] && (3.11)\\ &= E\left[F\left(\beta_{t-1} - \beta_{t-1|t-1}\right)\left(\beta_{t-1} - \beta_{t-1|t-1}\mu_z\right)' F'\right] + E[v_t v'_t] \\ &= FP_{t-1|t-1}F' + Q && (3.12) \end{aligned}$$

where $P_{t-1|t-1} = var(\beta_{t-1}|\psi_{t-1})$. The forecast error variance depends on the previous error variance; that value depends on the information set.

If β_{t-1} forms part of the information set ψ_{t-1} then

$$P_{t-1|t-1} = var\left(\beta_{t-1}|\psi_{t-1}\right) = 0 \tag{3.13}$$

and there is no uncertainty other than from the disturbance terms and $P_t = Q$.

If β_{t-1} does not form part of the information set ψ_{t-1} but β_{t-2} does then $P_{t-1|t-1} = Q$ and $P_{t|t-1} = FQF' + Q$. This can be continued backwards; the unconditional (steady-state) covariance of β_t is the limit $P = FPF' + Q$. We can easily calculate error bands for β_t using the appropriate information set.

3.3.3 Prediction error

We can turn this around, as it must be the *prediction errors* are given by

$$\begin{aligned}
\eta_{t|t-1} &= y_t - E[y_t|\psi_{t-1}] && (3.14)\\
&= y_t - E[H_t\beta_t + e_t|\psi_{t-1}] && (3.15)\\
&= y_t - H_t\beta_{t|t-1} && (3.16)
\end{aligned}$$

where $\eta_{t|t-1}$ is uncorrelated with ψ_{t-1}. So the 'news' over that contained in y_t above ψ_{t-1} is captured by $\eta_{t|t-1}$. It will be that $\eta_{t|t-1} \sim N(0, \Sigma_{\eta\eta})$; we need to find an expression for the covariance.

3.3.4 Current-data predictions

Now we find $E[\beta_t|\psi_t]$ – the best prediction of the unknown coefficient vector given *current* information. Using the regression lemma we know that

$$\begin{aligned}
E[\beta_t|\psi_t] &= E[\beta_t|\psi_{t-1}, \eta_{t|t-1}] && (3.17)\\
&= E[\beta_t|\psi_{t-1}] + \Sigma_{\beta\eta}\Sigma_{\eta\eta}^{-1}\eta_{t|t-1} && (3.18)\\
&= \beta_{t|t-1} + \Sigma_{\beta\eta}\Sigma_{\eta\eta}^{-1}\eta_{t|t-1} && (3.19)
\end{aligned}$$

because ψ_{t-1} and $\eta_{t|t-1}$ are uncorrelated and $\eta_{t|t-1}$ is mean zero.

Similarly, we can find $P_{t|t}$ as the best prediction of the variance of β_t given ψ_t. Using the regression lemma we know that

$$\begin{aligned}
P_{t|t} &= E[(\beta_t - \beta_{t|t})(\beta_t - \beta_{t|t})'|\psi_{t-1}, \eta_{t|t-1}] && (3.20)\\
&= E[(\beta_t - \beta_{t|t})(\beta_t - \beta_{t|t})'|\psi_{t-1}] - \Sigma_{\beta\eta}\Sigma_{\eta\eta}^{-1}\Sigma_{\eta\beta} && (3.21)\\
&= P_{t|t-1} - \Sigma_{\beta\eta}\Sigma_{\eta\eta}^{-1}\Sigma_{\eta\beta} && (3.22)
\end{aligned}$$

3.3.5 Estimated model covariances

All we need do is plug the relevant expressions into the regression lemma. So, what is $\Sigma_{\beta\eta}$?

$$\begin{aligned}
\Sigma_{\beta\eta} &= E[(\beta_t - \beta_{t|t-1})\eta'_{t|t-1}] && (3.23)\\
&= E\left[(\beta_t - \beta_{t|t-1})(y_t - H_t\beta_{t|t-1})'\right] && (3.24)\\
&= E\left[(\beta_t - \beta_{t|t-1})(H_t\beta_t + e_t - H_t\beta_{t|t-1})'\right] && (3.25)\\
&= E[(\beta_t - \beta_{t|t-1})(\beta_t - \beta_{t|t-1})'H'_t] + E[(\beta_t - \beta_{t|t-1})e'_t] && (3.26)\\
&= P_{t|t-1}H'_t && (\Sigma_{\beta\eta})
\end{aligned}$$

as $E[(\beta_t - \beta_{t|t-1})e'_t] = 0$.

What is $\Sigma_{\eta\eta}$?

$$\begin{aligned}
\Sigma_{\eta\eta} &= E[(y_t - H_t\beta_{t|t-1})(y_t - H_t\beta_{t|t-1})'] && (3.27)\\
&= E[(H_t\beta_t + e_t - H_t\beta_{t|t-1})(H_t\beta_t + e_t - H_t\beta_{t|t-1})'] && (3.28)\\
&= E[(H_t\beta_t - H_t\beta_{t|t-1})(H_t\beta_t - H_t\beta_{t|t-1})'] + E[e_te'_t] && (3.29)\\
&= E[H_t(\beta_t - \beta_{t|t-1})(\beta_t - \beta_{t|t-1})'H'_t] + R && (3.30)\\
&= H_tP_{t|t-1}H'_t + R && (3.31)\\
&= f_{t|t-1} && (\Sigma_{\eta\eta})
\end{aligned}$$

where we define $f_{t|t-1} = E[\eta_{t|t-1}\eta'_{t|t-1}]$. Now we're ready.

3.4 The Kalman filter

The equations of the filter are

- the conditional expectation depending on ψ_{t-1};
- an update that uses $\eta_{t|t-1}$ to obtain the best t-period prediction now based on ψ_t.

These must be of the form

$$\begin{aligned}
E[\beta_t|\psi_{t-1}] &= \mu + FE[\beta_t|\psi_{t-1}] && (3.32)\\
E[P_t|\psi_{t-1}] &= FE[P_{t-1}|\psi_{t-1}]F' + Q && (3.33)\\
E[\beta_t|\psi_t] &= E[\beta_t|\psi_{t-1}] + \Sigma_{\beta\eta}\Sigma^{-1}_{\eta\eta}\eta_{t|t-1} && (3.34)\\
E[P_t|\psi_t] &= E[P_t|\psi_{t-1}] - \Sigma_{\beta\eta}\Sigma^{-1}_{\eta\eta}\Sigma_{\eta\beta} && (3.35)
\end{aligned}$$

These are specifically

$$\begin{aligned}
\beta_{t|t-1} &= \mu + F\beta_{t-1|t-1} && \text{(Predicted } \beta\text{)} \\
P_{t|t-1} &= FP_{t-1|t-1}F' + Q && \text{(Predicted } P\text{)} \\
\eta_{t|t-1} &= y_t - H_t\beta_{t|t-1} && \text{(Prediction error)} \\
f_{t|t-1} &= H_tP_{t|t-1}H_t' + R && \text{(Pred. err. variance)} \\
\beta_{t|t} &= \beta_{t|t-1} + P_{t|t-1}H_t'f_{t|t-1}^{-1}\eta_{t|t-1} && \text{(Updated } \beta\text{)} \\
P_{t|t} &= P_{t|t-1} - P_{t|t-1}H_t'f_{t|t-1}^{-1}H_tP_{t|t-1} && \text{(Updated } P\text{)}
\end{aligned}$$

The filter evaluates these recursively, beginning from β_0, P_0.

Treatment of these initial condition reflects knowledge/model

- Stationary models can use the steady-state
- Non-stationary models use something which is often (confusingly) called a *diffuse prior* (zero mean, large variance)

3.4.1 Kalman filter trick

For ***known*** initial conditions – say $\beta_0 \sim N(\mu_0, P_0)$ – the likelihood of a state-space model with T observations of m variables is

$$\log L(\theta|y) = \sum_{t=1}^{T} \log\left(p\left(\theta|\psi_{t-1}, \theta\right)\right) \tag{3.36}$$

$$= -\Phi - \frac{1}{2}\sum_{t=1}^{T}\left(\log(\det(f_{t|t-1})) + \eta_{t|t-1}'f_{t|t-1}^{-1}\eta_{t|t-1}|\theta\right) \tag{3.37}$$

where $\Phi = \frac{Tm}{2}\log(2\pi)$ and θ are all the non-state parameters to be estimated. We can use the Kalman filter to obtain $\eta_{t|t-1}$ and $f_{t|t-1}$ as they are the *prediction error* and its *variance*. This is the ***prediction error decomposition*** of the log-likelihood.

A maximum likelihood estimate maximizes $\log L(y|\theta)$ by choice of θ.

Part II

Quantiles

Chapter 4

Quantile regression

This shows how to manipulate data from the SPF in R.

```
library(readxl)
library(xts)
library(lubridate)
library(tidyverse)
library(quantmod)
library(quantreg)
```

4.1 Getting the data

We download the data and save it locally.

```
h <- "https://www.philadelphiafed.org/-/media/frbp/assets/surveys-and-data/survey-of-professio
f <- "meanlevel.xlsx"

download.file(paste0(h, f), destfile=f, mode="wb")
```

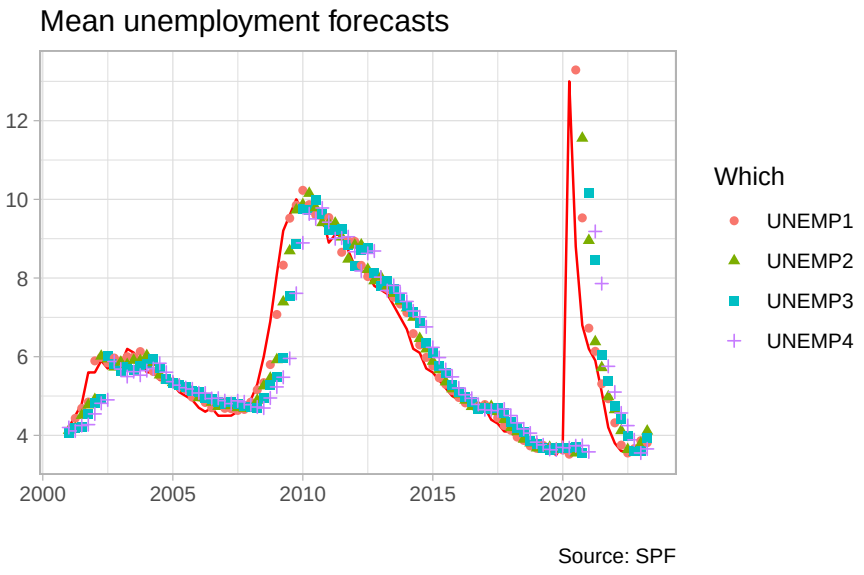
Retrieve the unemployment data for the average unemployment forecast.

```
UNEMP <- f %>%
  read_excel(na="#N/A", sheet="UNEMP") %>%
  mutate(Date=as.Date(as.yearqtr(paste(YEAR, QUARTER), format="%Y %q")))

Usel <- UNEMP %>%
  select(Date, UNEMP1, UNEMP3, UNEMP4, UNEMP5, UNEMP6) %>%
```

```
  mutate(UNRATE = lead(UNEMP1,1)) %>%
  select(Date, UNRATE,
         UNEMP1=UNEMP3, UNEMP2=UNEMP4, UNEMP3=UNEMP5, UNEMP4=UNEMP6) %>%
  mutate(UNEMP1 = lag(UNEMP1,1),
         UNEMP2 = lag(UNEMP2,2),
         UNEMP3 = lag(UNEMP3,3),
         UNEMP4 = lag(UNEMP4,4)) %>%
  pivot_longer(cols = -c(Date, UNRATE), names_to="Which", values_to="Val") %>%
  filter(year(Date) > 2000)
```

4.2 Plots

```
Usel %>%
  ggplot(aes(x=Date)) +
  geom_line(aes(y=UNRATE), colour="red") +
  geom_point(aes(y=Val, colour=Which, shape=Which)) +
  theme_light() +
  labs(title="Mean unemployment forecasts", x="", y="", caption="Source: SPF")
```



Chapter 5

Carter-Kohn

5.1 Using the Kalman Filter

Establish the usefulness of the Kalman Filter (and not just for state estimation). Refresh idea of maximum likelihood estimation in the context of state space models.

- Introduce *smoothing*
- Develop Gibbs sampling by the Carter-Kohn method
- All of these use the Kalman Filter to develop conceptually different tools

Follow Kim and Nelson (1999); also see Harvey (1989), Hamilton (1994), Durbin and Koopman (2001)

5.2 Maximum likelihood

5.2.1 Classical Maximum Likelihood Estimation

The principle of maximum likelihood is that the parameters should be chosen so that the probability of observing a given sample is maximized.

For time series models the joint density of $\psi_T = \{y_T, y_{T-1}, \dots, y_1\}$ and parameters θ in conditional form is

$$p(\theta|\psi_T) = \prod_{t=1}^{T} p(y_t|\psi_{t-1}, \theta) \tag{5.1}$$

emphasizing the serial dependence of observations.

Interpret this as the likelihood for a particular sample. Assuming (conditional) normality, the likelihood of any particular n-vector of observations is

$$p(y_t) = (2\pi)^{-\frac{n}{2}} |var(y_t)|^{-\frac{1}{2}} e^{\{-\frac{1}{2}(y_t-\mu)' var(y_t)^{-1}(y_t-\mu)\}} \tag{5.2}$$

Notice this depends on the observed data *and* the values of the parameters. It can be multivariate and for any underlying density. A maximum likelihood (ML) estimate of θ maximizes the likelihood of the parameter given an observed sample.

5.2.2 Poisson example

Th Poisson distribution is a nice one to consider as the maximum likelihood estimate can be calculated easily – essentially in your head.[1]

The Poisson distribution is

$$p(y_i,\ \theta) = \frac{e^{-\theta}\theta^{y_i}}{y_i!} \tag{5.4}$$

for $y > 0$, zero otherwise with the property $E[Y] = var(Y) = \theta$.

Count variables often modelled as a random Poisson process: numbers of road traffic accidents, sales, telephone calls, electron emissions. Greene's example is to find the most likely value of θ given observations.

$$5,\ 0,\ 1,\ 1,\ 0,\ 3,\ 2,\ 3,\ 4,\ 1 \tag{5.5}$$

For independent observations the joint density is

$$p(y,\ \theta) = \prod_{i=1}^{10} p(y_i,\ \theta) = \frac{e^{-10\theta}\theta^{\sum_i y_i}}{\prod_i (y_i!)} = \frac{e^{-10\theta}\theta^{20}}{207,360} \tag{5.6}$$

We can plot this function to see if it has a maximum

We can also find this by calculus. As ever, because the log function is monotonic it is convenient to take logs

$$\ln L(\theta) = -10\theta + 20\ln\theta - \ln(207,360) \tag{5.7}$$

First order conditions are

$$\frac{\partial \ln L(\theta)}{\partial \theta} = -10 + \frac{20}{\theta} \Rightarrow \theta = \frac{20}{10} = 2 \tag{5.8}$$

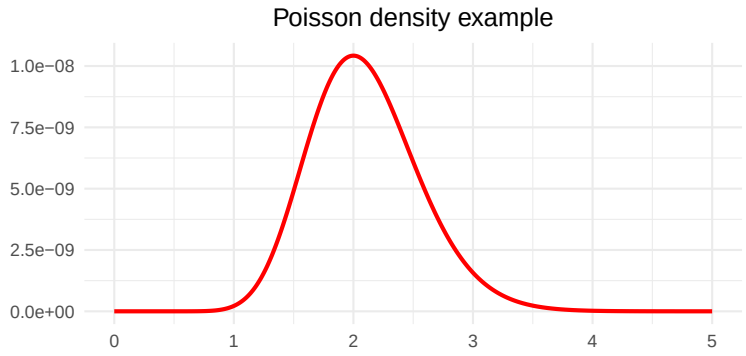
Check for maximum

$$\frac{\partial^2 \ln L(\theta)}{\partial \theta^2} = -\frac{20}{\theta^2} < 0 \tag{5.9}$$

[1] For example Greene (1997) constructs a Poisson distribution example where the chosen observations yield an exact estimate of the underlying parameter of the distribution. The example is to find the most likely value of θ given observations

$$5,\ 0,\ 1,\ 1,\ 0,\ 3,\ 2,\ 3,\ 4,\ 1 \tag{5.3}$$

ten observations which sum to 20.



5.3 Poisson example

The Poisson density for each observation is

$$p(y_i, \theta) = \frac{e^{-\theta}\theta^{y_i}}{y_i!} \tag{5.10}$$

for $y > 0$, zero otherwise, with $E[Y] = var(Y) = \theta$. For n independent observations, joint density is

$$P(y, \theta) = \prod_{i=1}^{n} p(y_i, \theta) = \frac{e^{-n\theta}\theta^{\sum_i y_i}}{\prod_i (y_i!)} \tag{5.11}$$

Interpret this as a likelihood function, i.e. a probability measure for θ given some observed y

$$L(\theta|y) = P(y, \theta) \tag{5.12}$$

As log function is monotonic

$$\ln L(\theta|y) = -n\theta + \textstyle\sum_i y_i \ln\theta - \ln\left(\prod_i y_i!\right) \tag{5.13}$$

First order conditions are

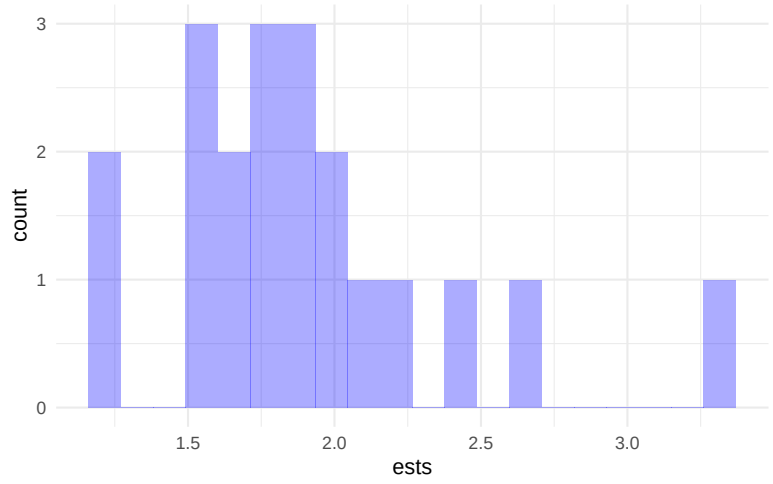
$$\frac{\partial \ln L(\theta|y)}{\partial\theta} = -n + \frac{\sum_i y_i}{\theta} \Rightarrow \theta = \frac{\sum_i y_i}{n} \tag{5.14}$$

Finally, check for maximum

$$\frac{\partial^2 \ln L(\theta)}{(\partial\theta)^2} = -\frac{\sum_i y_i}{\theta^2} < 0 \tag{5.15}$$

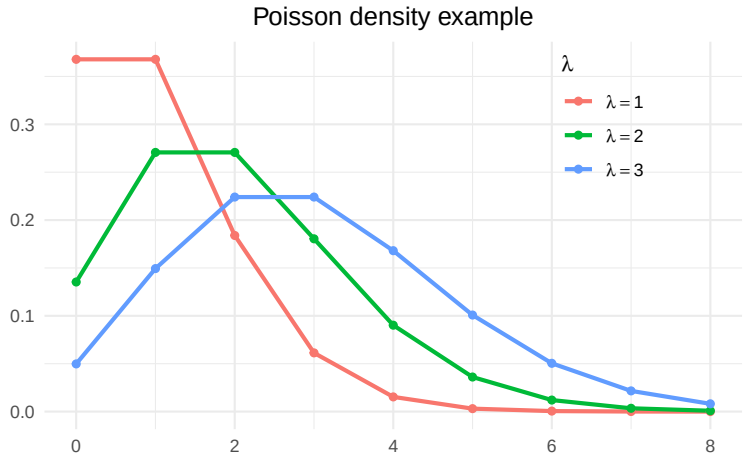
```
library(tidyverse)
library(scales)

reps   <- 20
lambda <- 2
n      <- 10

a <- tibble(x=seq(0, 3*lambda),
            p=(exp(-lambda*x)*(lambda^x))/prod(factorial(lambda)))

d <- matrix(rpois(n*reps,lambda), n, reps)

t <- as_tibble(factorial(d), .name_repair = ~paste0("Rep",1:reps)) %>%
  pivot_longer(cols = everything()) %>%
  group_by(name) %>%
  summarise(dd = prod(value)) %>%
  mutate(sum  = colSums(d),
         ests = sum/n)
```

```
ggplot(t) +
  geom_histogram(aes(x=ests), fill="blue", alpha=.33, color=NA, bins=20) +
  theme_minimal()
```



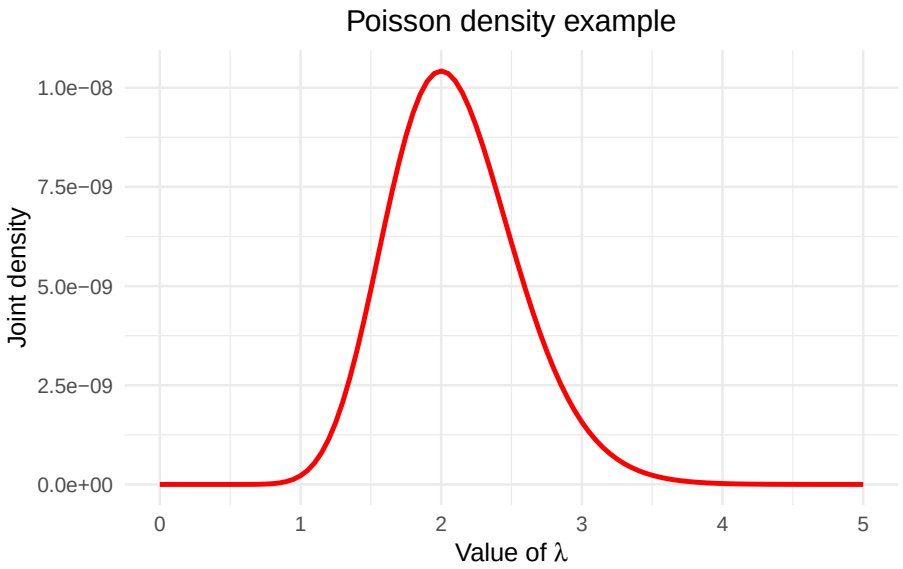

```
theta <- seq(0, 8, 1)
y     <- (exp(-lambda)*lambda^(theta))/(factorial(theta))
df    <- tibble(theta = theta) %>%
```

```
  mutate(`lambda == 1` = (exp(-1)*1^(theta))/(factorial(theta)),
         `lambda == 2` = (exp(-2)*2^(theta))/(factorial(theta)),
         `lambda == 3` = (exp(-3)*3^(theta))/(factorial(theta)))

df %>%
  pivot_longer(cols = -theta, names_to = "lambda") %>%
  ggplot() +
  geom_line(aes(x=theta, y=value, color=lambda), size=1) +
  geom_point(aes(x=theta, y=value, color=lambda)) +
  theme_minimal() +
  labs(title="Poisson density example", x="", y="",
       color=expression(lambda)) +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = c(.8,.8)) +
  scale_colour_discrete(labels = parse_format())
```




```
e <- c(5,0,1,1,0,3,2,3,4,1)
l <- seq(0,5,0.05)
y <- (exp(-n*l)*l^(sum(e)))/(prod(factorial(e)))

tibble(l=l, y=y) %>%
  ggplot() +
  geom_line(aes(x=l, y=y), color="red", linewidth=1) +
  theme_minimal() +
  labs(title="Poisson density example",
```

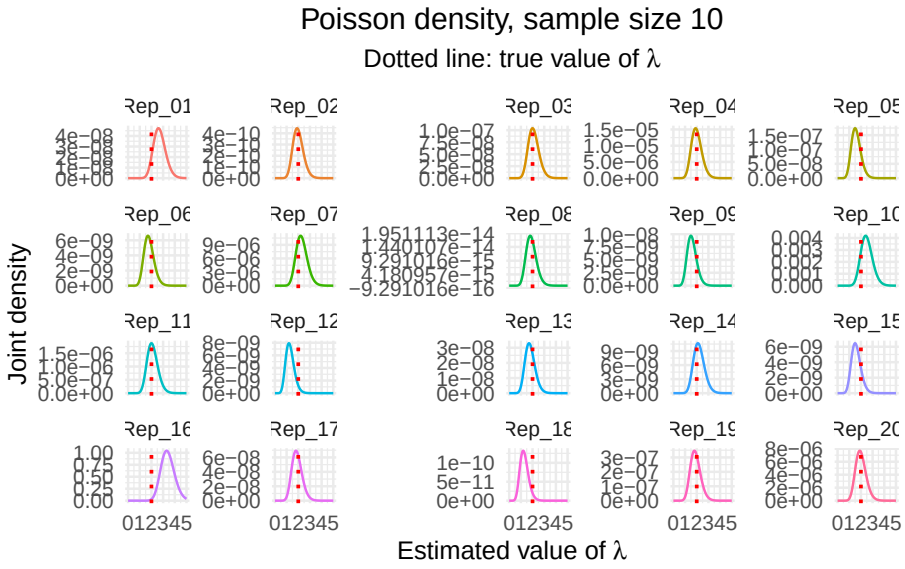
```
       x=expression(paste("Value of ", lambda)),
       y="Joint density") +
  theme(plot.title = element_text(hjust = 0.5))
```



Maintain the value of λ of 2. Now generate $reps = 20$ replications of $n = 10$ observations and plot the empirical density of each and the maximum likelihood estimate for each replication.

```
t2 <- t %>%
  group_by(name) %>%
  mutate(r = list((exp(-n*l)*l^sum)/dd))

nms <- paste0("Rep_", str_pad(1:reps, 2, pad="0"))

t2$r %>%
  bind_cols(.name_repair = ~ nms) %>%
  mutate(l = l) %>%
  pivot_longer(cols = -l) %>%
  ggplot() +
  geom_line(aes(x=l, y=value, color=name), show.legend = FALSE) +
  geom_vline(aes(xintercept=lambda), color="red", linetype=3, linewidth=0.75) +
  facet_wrap(~ name, scales = "free_y") +
  theme_minimal() +
  labs(title    = paste("Poisson density, sample size", n),
```

```
      subtitle = expression(paste("Dotted line: true value of ", lambda)),
      x        = expression(paste("Estimated value of ", lambda)),
      y        = "Joint density") +
theme(plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5))
```



5.3.1 ML and regression

Linear regression problem is

$$L(\beta|y,X) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\frac{1}{2\sigma^2}(y-X\beta)'(y-X\beta)\right] \tag{5.16}$$

The log-likelihood is

$$\ln L = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}(y-X\beta)'(y-X\beta) \tag{5.17}$$

As before, find the extremum by calculus; yields *likelihood equations*

$$\frac{\partial \ln L}{\partial \beta} = -\frac{2}{2\sigma^2}(X'y - X'X\beta) = 0 \tag{5.18}$$

$$\Rightarrow \hat{\beta}_{ml} = (X'X)^{-1}X'y \tag{5.19}$$

and

$$\frac{\partial \ln L}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}(y - X\beta)'(y - X\beta) = 0 \tag{5.20}$$

$$\Rightarrow -n + \sigma^{-2}(\epsilon'\epsilon) = 0 \tag{5.21}$$

$$\Rightarrow \hat{\sigma}^2_{ml} = \frac{\hat{\epsilon}'\hat{\epsilon}}{n} \tag{5.22}$$

ML estimate of σ^2 divided by n (not $n-k$) so biased in small samples but not asymptotically

5.4 Kalman filter tricks

For some initial condition – say $\beta_0 \sim N(\mu_0, P_0)$) – the conditional log-likelihood for sample 1 to T

$$\log L(\psi_t|\theta) = \sum_{t=1}^{T} \log p(y_t|\psi_{t-1}, \theta) \tag{5.23}$$

$$\propto -\sum_{t=1}^{T} \left(\log \left|f_{t|t-1}\right| + \eta'_{t|t-1} f^{-1}_{t|t-1} \eta_{t|t-1}| \ \theta\right) \tag{5.24}$$

Note we could obtain $\eta_{t|t-1}$ and $f_{t|t-1}$ from the Kalman filter, i.e.

$$f_{t|t-1} = (H_t P_{t|t-1} H'_t + Q) = \Sigma_{\eta\eta} \tag{5.25}$$

This is the *prediction error decomposition* of the log-likelihood. For a classical approach we estimate θ by numerically maximizing $\log L(\psi_T|\theta)$.This gives a point estimate for the value of θ and we typically apply classical inference using the estimated standard errors. Note to do this we need to evaluate the best estimate of the state as well as maximize the likelihood: the Kalman Filter is a key ingredient in both.

> **Maximisation**
>
> A suitable numerical maximization routine will (in principle) maximize the likelihood straightforwardly. Often use Chris Sims' **csminwel** in Matlab or R as well-suited to this type of problem.

Can show (via Cramer-Rao) that

$$\hat{\theta} \sim N\left(\theta, -\frac{\partial^2 \log L(\psi_T|\theta)}{\partial\theta\partial\theta'}\right) \tag{5.26}$$

5.5 Full sample estimates of β_t

5.5.1 The regression lemma again

You may recall that for any

$$\begin{bmatrix} z \\ y \\ \varepsilon \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_z \\ \mu_y \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{zz} & \Sigma_{zy} & \Sigma_{z\varepsilon} \\ \Sigma_{yz} & \Sigma_{yy} & 0 \\ \Sigma_{\varepsilon z} & 0 & \Sigma_{\varepsilon\varepsilon} \end{bmatrix} \right) \tag{5.27}$$

then it must be that

$$E[z|y,\varepsilon] = \mu_z + \Sigma_{zy}\Sigma_{yy}^{-1}(y-\mu_y) + \Sigma_{z\varepsilon}\Sigma_{\varepsilon\varepsilon}^{-1}\varepsilon \tag{5.28}$$

$$= E[z|y] + \Sigma_{z\varepsilon}\Sigma_{\varepsilon\varepsilon}^{-1}\varepsilon \tag{5.29}$$

We use this to derive a recursive update to *smooth* our estimates. We will also derive an appropriate conditional expectation which we can use in Gibbs sampling.

5.6 Smoothing

The Kalman filter estimates β_t recursively: it only uses information available up until time t. This means that the estimate of $\beta_{T|T}$ uses all available information, but any previous estimate doesn't. Indeed there must be some values of $\eta_{i|t-1}$

$$\beta_{t|T} = E(\beta_t|\psi_{t-1}, \eta_{t|t-1}, \eta_{t+1|t-1}, ..., \eta_{T|t-1}) \tag{5.30}$$

where the 'news' is relative to period t.

We could update $\beta_{t|t-1}$ using the (uncorrelated) future innovations

$$\beta_{t|T} = \beta_{t|t} + \sum_{j=t}^{T} \Sigma_{\beta_t\eta_j}\Sigma_{\eta_j\eta_j}^{-1}\eta_{j|t-1} \tag{5.31}$$

and recalling $\beta_{t|t} = E(\beta_t|\psi_{t-1}, \eta_{t|t-1})$.

This is a *fixed interval smoother*; often used for full sample estimates of β_t. Remember we already have an estimate of $\beta_{T|T}$ from the Kalman filter so *smoothers work backwards*; we sketch a derivation here.

In the last but one period we have a different prediction error

$$\varsigma_{T|T-1} = \beta_{T|T} - F\beta_{T-1|T-1} - \mu \tag{5.32}$$

which is the error in predicting β_T using ψ_{T-1}.

An 'update' has to be of the form

$$\beta_{T-1|T} = \beta_{T-1|T-1} + \Sigma_{\beta\varsigma}\Sigma_{\varsigma\varsigma}^{-1}\varsigma_{T|T-1} \tag{5.33}$$

where $\Sigma_{\varsigma\varsigma} = var\left[\varsigma_T|\psi_{T-1}\right]$ and $\Sigma_{\beta\varsigma} = cov\left[\beta_{T-1}, \varsigma_T|\psi_{T-1}\right]$.

These are

$$\begin{aligned}
\Sigma_{\varsigma\varsigma} &= var(\beta_T - F\beta_{T-1|T-1} - \mu) && (5.34)\\
&= var\left(F(\beta_{T-1} - \beta_{T-1|T-1}) + e_t\right) && (5.35)\\
&= FP_{T-1|T-1}F' + Q && (5.36)
\end{aligned}$$

and

$$\begin{aligned}
\Sigma_{\beta\varsigma} &= E\left[(\beta_{T-1} - \beta_{T-1|T-1})\left(\beta_T - F\beta_{T-1|T-1} - \mu\right)'\right] && (5.37)\\
&= E\left[(\beta_{T-1} - \beta_{T-1|T-1})(\beta_T - \beta_{T-1|T-1})'\right]F' && (5.38)\\
&= P_{T-1|T-1}F' && (5.39)
\end{aligned}$$

Plugging these definitions in gives us

$$\beta_{T-1|T} = \beta_{T-1|T-1} + P_{T-1|T-1}F'P_{T|T-1}^{-1}(\beta_{T|T} - F\beta_{T-1|T-1} - \mu) \tag{5.40}$$

Applying the argument backward in time gives the recursion

$$\begin{aligned}
\beta_{t|T} &= \beta_{t|t} + P_{t|t}F'P_{t+1|t}^{-1}(\beta_{t+1|T} - F\beta_{t|t} - \mu) && (5.41)\\
&= \beta_{t|t} - K_{t|T}(\beta_{t+1|T} - F\beta_{t|t} - \mu) && \text{(smooth)}
\end{aligned}$$

All these quantities are outputs of the Kalman filter so smoothing is easy to implement.

The smoothed variance of $\beta_{t|T}$ found by multiplying out (smooth). To do this use $\beta_{t+1|t} = \mu + F\beta_{t|t}$ so rearranging gives

$$\tilde{\beta}_{t|T} + K_{t|T}\beta_{t+1|T} = \tilde{\beta}_{t|t} + K_{t|T}\beta_{t+1|t} \tag{5.42}$$

where $\tilde{\beta}_{t|t} = \beta_t - \beta_{t|t}$. Now square both sides and take expectations

$$P_{t|T} + K_{t|T}E\left[\beta_{t+1|T}\beta_{t+1|T}'\right]K_{t|T}' = P_{t|t} + K_{t|T}E\left[\beta_{t+1|t}\beta_{t+1|t}'\right]K_{t|T}' \tag{5.43}$$

Adding and subtracting $E[\beta_{t+1}\beta_{t+1}']$ we can show that

$$-E\left[\beta_{t+1|T}\beta_{t+1|T}'\right] + E\left[\beta_{t+1|t}\beta_{t+1|t}'\right] = P_{t+1|T} - P_{t+1|t} \tag{5.44}$$

to obtain

$$P_{t|T} = P_{t|t} + K_{t|T}(P_{t+1|T} - P_{t+1|t})K_{t|T}'. \tag{5.45}$$

5.7 Kalman filter in econometrics

5.7.1 Classical approach

The typical procedure is some variation on:

- Formulate state-space model
- Estimate the model by maximum likelihood
- Condition on the parameters to retrieve the (usually smoothed) state estimates and standard errors
- Use Cramer-Rao to calculate the standard errors of any other parameter estimates

For this the Kalman filter is a useful tool, as it allows a great deal of flexibility in the estimation of a variety of models, as is is an appropriate tool for models with unobserved components. However, it must be used with care: it is easy to try to estimate models that are essentially unidentified.

Further useful tools

- The *Extended Kalman filter* linearises the filter at every step and can be used for nonlinear models (such as ones where you need to estimate B_T and θ simultaneously)
- Increasingly non-Gaussian non-linear models are estimated using the *particle filter*

5.7.2 Bayesian approach

Bayesian approach is to generate the entire distribution of the model parameters.

- Now no longer just look for the point estimate obtained by maximum likelihood
- Use Gibbs sampling or some other appropriate method applied to the state space model
- In particular we treat the states and the parameters as jointly determined by the data
- As the state is estimated we need a way to draw the states conditional on the other estimates to do Gibbs sampling
- Seek a conditional updating algorithm that replicates the Gibbs sampling approach we have used before

We require a procedure such that:

- **Step 1** Conditional on θ and the data, generate the sequence $B_T = (\beta_1, \beta_2, ..., \beta_T)$
- **Step 2** Conditional on B_T and the data, generate values of θ
- **Step 3** Iterate previous two steps until convergence

In this way the joint distribution of the two can be obtained from the resulting simulation.

5.8 Carter-Kohn algorithm

- **Step 2** above relatively easy but how do we generate a sequence of states?
  - The state estimates depend on the parameter value through the Kalman filter
- Appropriate algorithm designed by Carter and Kohn (1994)
- Takes the form of a modified Kalman smoother
- Known as *multimove Gibbs sampling*
- Similar to above, define

$$B_t = [\beta_1 \quad \beta_2 \quad ... \quad \beta_t] \tag{5.46}$$

so in particular

$$B_{T-1} = [\beta_1 \quad \beta_2 \quad ... \quad \beta_{T-1}] \tag{5.47}$$

consistent with out earlier definition of ψ_t.

- Multimove Gibbs sampling generates the whole vector of states (B_T) at once
- We therefore need to generate a realization of B_T given the probability distribution $p(B_T|\psi_T)$
- We want to generate an appropriate conditional probability distribution $p(\beta_t|B_{j\neq t}, \psi_T)$ to sample from for our Gibbs sampler
- Just as for the Kalman smoother we use the outputs of the Kalman filter and a separate backward recursion to obtain the conditional distribution

5.9 Joint distribution

Deriving the appropriate distributions is easy if we know what to condition on. The joint probability density function can be split into a sequence of conditional distributions: $p(B_T|\psi_T)$ can be written recursively

$$p(B_T|\psi_T) = p(\beta_T|\psi_T) \times p(B_{T-1}|\beta_T, \psi_T) \tag{5.48}$$
$$= p(\beta_T|\psi_T) \times p(\beta_{T-1}|\beta_T, \psi_T) \times p(B_{T-2}|\beta_{T-1}, \beta_T, \psi_T) \tag{5.49}$$
$$= p(\beta_T|\psi_T) \times p(\beta_{T-1}|\beta_T, \psi_T) \times p(B_{T-2}|\beta_{T-1}, \psi_T) \tag{5.50}$$

Final simplification follows as the state vector is a Markov chain so there is no information in β_T not contained in β_{T-1} and ψ_T. Further as soon as we know β_{T-1} there is no information contained in ψ_T so we can drop that, so

$$p(B_T|\psi_T) = p(\beta_T|\psi_T) \times p(\beta_{T-1}|\beta_T, \psi_T) \times p(B_{T-2}|\beta_{T-1}, \psi_T) \tag{5.51}$$
$$= p(\beta_T|\psi_T) \times p(\beta_{T-1}|\beta_T, \psi_{T-1}) \times p(B_{T-2}|\beta_{T-1}, \psi_{T-2}) \tag{5.52}$$
$$= p(\beta_T|\psi_T) \times \prod_{t=1}^{T-1} p(\beta_t|\beta_{t+1}, \psi_t) \tag{5.53}$$

5.10 The Carter-Kohn equations

The estimated β variables are distributed

$$\beta_{T|\psi_T} \sim N(\beta_{T|T}, P_{T|T}) \tag{5.54}$$
$$\beta_{t|\psi_t,\beta_{t+1}} \sim N(\beta_{t|t,\beta_{t+1}}, P_{t|t,\beta_{t+1}}) \tag{5.55}$$

where

$$\beta_{t|t,\beta_{t+1}} = E\left[\beta_t|\psi_t, \beta_{t+1}\right] = E\left[\beta_t|\beta_{t|t}, \beta_{t+1}\right] \tag{5.56}$$
$$P_{t|t,\beta_{t+1}} = cov\left[\beta_t|\psi_t, \beta_{t+1}\right] = cov\left[\beta_t|\beta_{t|t}, \beta_{t+1}\right] \tag{5.57}$$

Carter-Kohn derive appropriate recursions so that, for example, we update the state estimate conditioning on some known value of β_{t+1}

$$\beta_{t|t,\beta_{t+1}} = \beta_{t|t} - K_{t|t+1}(\beta_{t+1} - F\beta_{t|t} - \mu) \tag{5.58}$$

Define

$$\varsigma_{t+1|t} = \beta_{t+1} - F\beta_{t|t} - \mu \tag{5.59}$$

as the 'innovation' in predicted $\beta_{t+1|t}$ where we have some realized β_{t+1} drawn from its probability distribution. The Carter-Kohn smoother comprises updates

to the conditional expectations that use this news.

$$E[\beta_t|\psi_t, \beta_{t+1}] = E[\beta_t|\psi_t] + \Sigma_{\beta\varsigma}\Sigma_{\varsigma\varsigma}^{-1}\varsigma_{t+1|t} \tag{5.60}$$
$$= \beta_{t|t} + \Sigma_{\beta\varsigma}\Sigma_{\varsigma\varsigma}^{-1}\varsigma_{t+1|t} \tag{5.61}$$
$$var[\beta_t|\psi_t, \beta_{t+1}] = var[\beta_t|\psi_t] - \Sigma_{\beta\varsigma}\Sigma_{\varsigma\varsigma}^{-1}\Sigma_{\varsigma\beta} \tag{5.62}$$
$$= P_{t|t} - \Sigma_{\beta\varsigma}\Sigma_{\varsigma\varsigma}^{-1}\Sigma_{\varsigma\beta} \tag{5.63}$$
$$= P_{t|t,\beta_{t+1}} \tag{5.64}$$

Both $\beta_{t|t}$ and $P_{t|t}$ are outputs of the Kalman filter.

5.10.1 Deriving $\Sigma_{\beta\varsigma}$ and $\Sigma_{\varsigma\varsigma}$

As before we just plug in the definitions so

$$\Sigma_{\varsigma\varsigma} = var[\beta_{t+1} - F\beta_{t|t} - \mu] \tag{5.65}$$
$$= var\left[F\beta_t + \mu + v_{t+1} - F\beta_{t|t} - \mu\right] \tag{5.66}$$
$$= var\left[F(\beta_t - \beta_{t|t}) + v_{t+1}\right] \tag{5.67}$$
$$= FP_{t|t}F' + Q \tag{5.68}$$

and

$$\Sigma_{\beta\varsigma} = E\left[(\beta_t - \beta_{t|t})(\beta_{t+1} - F\beta_{t|t} - \mu)'\right] \tag{5.69}$$
$$= E\left[(\beta_t - \beta_{t|t})\left(F(\beta_t - \beta_{t|t}) + v_{t+1}\right)'\right] \tag{5.70}$$
$$= P_{t|t}F' \tag{5.71}$$

5.11 'Kalman gain' again

So using the definitions of the covariances and the regression lemma we get

$$\beta_{t|t,\beta_{t+1}} = \beta_{t|t} + \Sigma_{s\eta}\Sigma_{\eta\eta}^{-1}\varsigma_t \tag{5.72}$$
$$= \beta_{t|t} + P_{t|t}F'\left(FP_{t|t}F' + Q\right)^{-1}(\beta_{t+1} - F\beta_{t|t} - \mu) \tag{5.73}$$
$$= \beta_{t|t} - K_{t|t}(\beta_{t+1} - F\beta_{t|t} - \mu) \tag{5.74}$$

where

$$K_{t|t+1} = -P_{t|t}F'(FP_{t|t}F' + Q)^{-1} \tag{5.75}$$

Like the Kalman smoother, this uses the filter's estimate of $P_{t|t}$ and updates β_t using the error in predicting β_{t+1} not y_t.

5.11.1 Conditional mean and variance of the state

Updating equations for the state and variance obtained directly from the regression lemma

$$\beta_{t|t,\beta_{t+1}} = \beta_{t|t} - K_{t|t+1}\varsigma_{t+1|t} \tag{5.76}$$

$$P_{t|t,\beta_{t+1}} = P_{t|t} - P_{t|t}F'(FP_{t|t}F' + Q)^{-1}FP_{t|t} \tag{5.77}$$

CK equations recursively evaluate these quantities *backwards* beginning from s_T, P_T obtained from the Kalman filter.

Generate appropriate conditional samples using

$$\beta_{t|t,\beta_{t+1}} \sim N\left(\beta_{t|t,\beta_{t+1}}, P_{t|t,\beta_{t+1}}\right) \tag{5.78}$$

to give $B_T|\psi_T$. This is a conditional sample that depends on a given parameter vector to use in a Gibbs sampling scheme that draws those parameters in turn from distributions conditioned on the states.

5.12 State-space Gibbs sampling in practice

Above approach cannot be used explicitly if $\Sigma_{\varsigma\varsigma}$ is singular, for example if we have more states than shocks (which is not uncommon). A simple modification given in KN can deal with this; we treat only those states that are shocked as observed.

In general we need conditional distributions for all the other parameters to be estimated. Need to store the complete sequence of states and covariances to implement the Gibbs sampler. We will investigate the exact implementation of Gibbs sampling for state-space models in the exercises.

5.13 Comparing the filters and smoothers

Filter	Innovation	Gain and state covariance
KF	$\eta_t = y_t - H_t\beta_{t\|t-1}$	$K_{t\|t} = -P_{t\|t-1}H_t'(H_tP_{t\|t-1}H_t' + R)^{-1}$
		$P_{t\|t} = P_{t\|t-1} - P_{t\|t-1}H_t'(H_tP_{t\|t-1}H_t' + R)^{-1}H_tP_{t\|t-1}$
KS	$\varsigma_t = \beta_{t+1\|T} - F\beta_{t\|t} - \mu$	$K_{t\|T} = -P_{t\|t}F'P_{t+1\|t}^{-1}$
		$P_{t\|T} = P_{t\|t} + K_{t\|T}(P_{t+1\|T} - P_{t+1\|t})K_{t\|T}'$
CK	$\varsigma_t = \beta_{t+1} - F\beta_{t\|t} - \mu$	$K_{t\|t,\beta_{t+1}} = -P_{t\|t}F'P_{t+1\|t}^{-1}$
		$P_{t\|t,\beta_{t+1}} = P_{t\|t} - P_{t\|t}F'(FP_{t\|t}F' + Q)^{-1}FP_{t\|t}$

Part III

Networks

Chapter 6

Causal Inference

Outline how to solve the Pearl, Glymour, and Jewell (2016) exercises in R.

6.1 Study question 1.3.2

Data:

```
library(tidyverse)
ed <- tibble(Gender = c("M","M","M","M","F","F","F","F"),
             eLevel = c("U","H","C","G","U","H","C","G"),
             num    = c(112,231,595,242,136,189,763,172)) %>%
  mutate(total = sum(num))
```

which we tabulate as

```
ed %>%
  kable()
```

Gender	eLevel	num	total
M	U	112	2440
M	H	231	2440
M	C	595	2440
M	G	242	2440
F	U	136	2440
F	H	189	2440
F	C	763	2440
F	G	172	2440

6.2 Exercises and answers

6.2.1 Find $P(eLevel = H)$

```
ed %>%
  filter(eLevel == "H") %>%
  mutate(p_H = sum(num)/total) %>%
  kable()
```

	Gender	eLevel	num	total	p_H
M		H	231	2440	0.1721311
F		H	189	2440	0.1721311

6.2.2 Find $P(eLevel = H \ \vee \ Gender = F)$

```
ed %>%
  filter(Gender == "F" | eLevel == "H") %>%
  mutate(p_HorF = sum(num)/total) %>%
  kable()
```

Gender	eLevel	num	total	p_HorF
M	H	231	2440	0.6110656
F	U	136	2440	0.6110656
F	H	189	2440	0.6110656
F	C	763	2440	0.6110656
F	G	172	2440	0.6110656

6.2.3 Find $P(eLevel = H \mid Gender = F)$

```
ed %>%
  filter(Gender == "F") %>%
  mutate(tcond = sum(num)) %>%
  filter(eLevel == "H") %>%
  mutate(p_HgivenF = sum(num)/tcond) %>%
  kable()
```

Gender	eLevel	num	total	tcond	p_HgivenF
F	H	189	2440	1260	0.15

6.2.4 Find $P(Gender = F \mid eLevel = H)$

```
ed %>%
  filter(eLevel == "H") %>%
  mutate(tcond = sum(num)) %>%
  filter(Gender == "F") %>%
  mutate(p_FgivenH = sum(num)/tcond) %>%
  kable()
```

Gender	eLevel	num	total	tcond	p_FgivenH
F	H	189	2440	420	0.45

Chapter 7

Mapping regional house price inflation

7.1 How heterogenous is UK house price inflation?

A simple enough question, and one that Bahaj, Foulis, and Pinter (2020) thought was best answered with a map – actually a referee asked for one. As I know how to draw a map in R they asked me if I could do it. Well yes, but there are some particular difficulties.

- The UK (actually Great Britain) is an awkward (but not too awkward) shape.
- Population in the UK is heavily concentrated in a small number of centres, such as London or Manchester.
- There are three different periods to compare.
- It has to be in grayscale.

Before all of this we need some data, with boundaries that correspond to areas that we have data for. The regional inflation data is available at the level of the Land Registry, which almost by local authority but amalgamates a number of the areas. So a map at Local Authority level would be fine as long as we can amalgamate some of the regions.

The map data used here is available from the UK's ONS geoportal, with a lot of administrative data available including local authority boundaries. The Local Authority data is specifically available from here, where I use the clipped full extent version. There are a number of possibilities, but in general high water mark, and enough but not too much detail is needed.

```
library(tidyverse)
library(readxl)
library(sf)
```

The information in the map file is comprehensive, and by Local Authority as of December 2015.

```
fle <- "LAD_Dec_2015_GCB_GB"
shape <- read_sf(dsn=".", layer=fle)
```

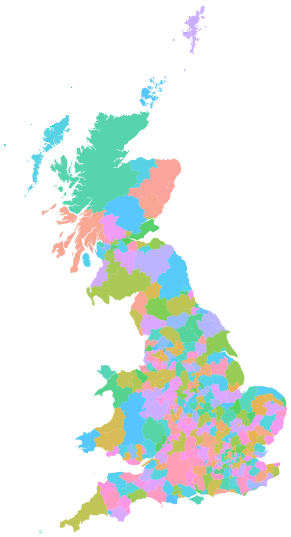
We can look at the attributes using `summary`.

```
summary(shape)
```

```
  lad15cd            lad15nm            lad15nmw           GlobalID
Length:380         Length:380         Length:380         Length:380
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
         geometry
MULTIPOLYGON :380
epsg:27700   :  0
+proj=tmer...:  0
```

This can be plotted straightforwardly using `ggplot`.

```
shape %>%
  ggplot() +
  geom_sf(aes(geometry=geometry, fill=lad15nm),
          color=NA, alpha=.66, show.legend=FALSE) +
  theme_void()
```



Looking at the read-out above, each of the 380 regions have some metadata associated, which are contained in each of the listed attributes. It should be obvious that `objectid` is just a sequence from 1 to 380. `lad15nm` turns out to be a list of names of the regions – I suspect `lad` for Local Authority District, `15` for 2015 and `nm` for name – and it is easy to specify this as the name to use for the region when using `tidy`.

Now this can be plotted using `ggplot`, using `geometry` for the x and y coordinates. The choice of fill colour is determined by `fill` and we can set the colour of the lines by `colour` (or `color`). The two extra arguments are for a suitable blank style and to impose an appropriate ratio of height to width.

Immediately, the awkward shape of the British Isles is apparent. (Note this is a plot of Great Britain, and there is no Northern Ireland.) The islands to the far north are somewhat unnecessary, although quite rightly the inhabitants get a bit tired of being left off maps! Nonetheless I'll do exactly the same by filtering out the polygons associated with `Orkney Islands` and `Shetland Islands`.

Fewer Scottish Islands makes the graphs a lot clearer with little loss of information, paticularly given the tiny number of transactions in the Orkneys and the Shetlands, very far to the north.

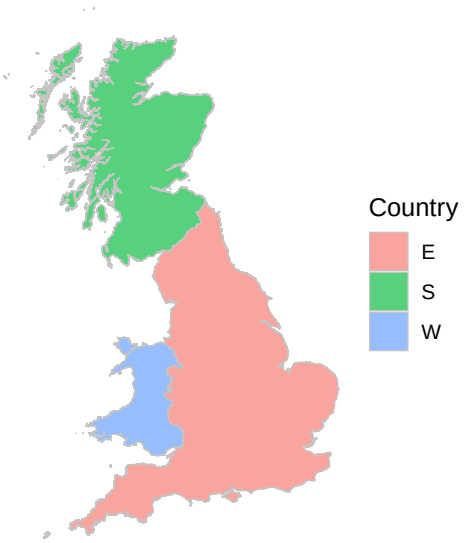
In what follows we filter out the islands using

```
shape <- read_sf(dsn=".", layer=fle) %>%
  filter(!lad15nm %in% c("Shetland Islands","Orkney Islands")) %>%
  mutate(Country=str_sub(lad15cd, 1, 1), .after=1)
```

where we also create an indicator of country using the first letter of the code string.

So the final country map is

```
shape %>%
  group_by(Country) %>%
  summarise() %>%
  ggplot() +
  geom_sf(aes(fill=Country), color="grey77", linewidth=.25, alpha=.66) +
  theme_void()
```



group and **summarise** can join geographical areas

Note the really nice feature – if we group by something, in this case country, we can summarise to amalgamate the geometries!

You may have noticed, one thing that that's missing on the LA graphs is the boundaries. They aren't, they're just invisible. That's because I set `colour = NA`, so I can fix that by choosing a colour and making the lines very thin so they don't swamp the map, as in the country one.

One further amendment, the `fill` is moved inside the `aes()` specification and made conditional. `R` now chooses unique colours for each of the regions.

Two things now need to be done to get the map colours right to illustrate re-

gional inflation rates. First we need to amalgamate some of the Local Authority boundaries to the Land Registry definitions, and second we need to assign the inflation rate to each area.

7.2 Inflation data and regions

We have a map, and we have that data in a form that is easy to understand. If we can suitably attach an inflation rate to each area then we can fill the individual areas with a colour unique to each individual inflation rates.

Recall that the Land Registry areas aren't quite what we have, and will need amalgamating. Bahaj, Foulis, and Pinter (2020) supplied me the areas that needed amalgamating (and the inflation rates) using the ONS codes. This is contained in the metadata `lad15cd` above.

The data is structured in 'wide' format with one row for each Land Registry region. The details aren't very important for us now, but what it means is I can manipulate it to get

```
# Price data by Land Registry region, converted to long format
hp_data <- read_excel("house_price_data_figure_1.xls")  %>%
  select("land_reg_region", starts_with("e_"), starts_with("av_")) %>%
  pivot_longer(names_to  = "name",
               values_to = "lad15cd",
               cols      = c(-land_reg_region, -starts_with("av_"))) %>%
  drop_na() %>%
  select(land_reg_region, lad15cd, starts_with("av_"))

codes <- hp_data %>%
  select(lad15cd, land_reg_region)
```

The important thing that the `pivot_longer` achieves is that for every `land_reg_region` I get a list of all the ONS codes that makes up the Local Authority level. So if I look at `buckinghamshire` as an example there are four ONS codes now associated with it.

```
filter(codes, land_reg_region == "buckinghamshire")
```

```
# A tibble: 4 x 2
  lad15cd   land_reg_region
  <chr>     <chr>
1 E07000004 buckinghamshire
2 E07000005 buckinghamshire
3 E07000006 buckinghamshire
```

```
4 E07000007 buckinghamshire
```

Join these together

```
# Join polygons defined by Land Registry regions
gg <- shape %>%
  select(starts_with(c("lad","C"))) %>%
  left_join(codes, by="lad15cd") %>%
  group_by(land_reg_region) %>%
  summarise()
```

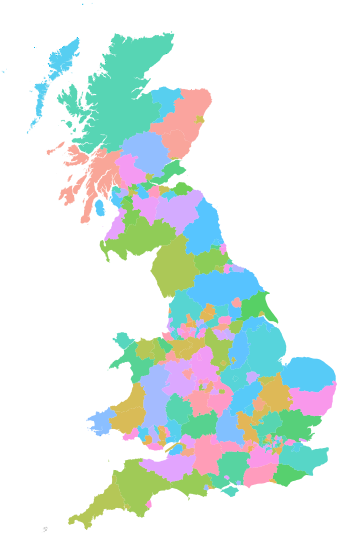
which produces a match between the Land Registry and the Local Authority areas, plus the inflation rates.

7.2.1 Inflation in grayscale

All the information required to plot the Land Registry-based regional inflation rates is now available. As you can see from the `buckinghamshire` data above, there are three average rates in three different periods, so I'll focus on one, 2002-2007 to begin with.

First, augment the geographic data with the inflation data, and call them something better.

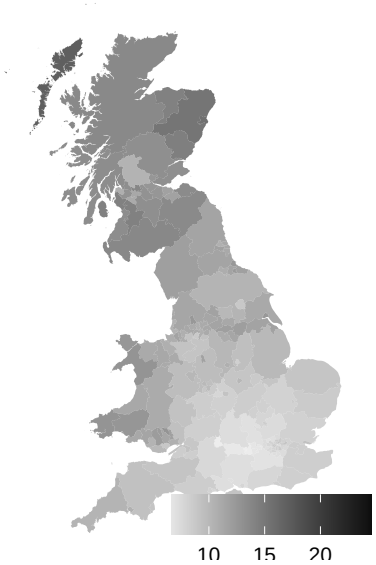
```
gg %>%
  ggplot() +
  geom_sf(aes(geometry=geometry, fill=land_reg_region),
          color=NA, alpha=.66, show.legend = FALSE) +
  theme_void()
```



Then specify gray and put the legend at the bottom.

```
nms <- gsub("av_hp_growth", "HPI", colnames(hp_data))

hp_data %>%
  rename_all( ~ nms) %>%
  select(land_reg_region, starts_with("HPI")) %>%
  distinct() %>%
  left_join(gg) %>%
  ggplot() +
  geom_sf(aes(geometry=geometry, fill=HPI_02_07),
          color=NA, alpha=.66, show.legend = TRUE) +
  theme_void() +
  scale_fill_gradient(low=grey(0.9), high=grey(0.05)) +
  theme(legend.direction = "horizontal",
        legend.position  = c(0.75,0.05),
        legend.title     = element_blank())
```

```
Joining with `by = join_by(land_reg_region)`
```


10
15
20

Part IV

Time

Chapter 8

Linear rational expectations models

8.1 Introduction

How do we solve rational expectations models? What does that even mean? Here I show how to implement versions of the Blanchard and Kahn (1980) and Klein (2000) solutions to linear rational expectations models in R. The implementation is fairly general, and copes with singular models. It is a very transparent implementation, with all the necessary code, and also shows how to calculate and plot impulse responses.

8.2 Model

We take a simple New Keynesian model

$$y_t = y^e_{t+1} - \frac{1}{\sigma}(i_t - \pi^e_{t+1}) + e^1_t \tag{8.1}$$

$$\pi_t = \beta\pi^e_{t+1} + \kappa y_t + e^2_t \tag{8.2}$$

$$i_t = \gamma i_{t-1} + (1-\gamma)\delta\pi_t + \varepsilon^3_t \tag{8.3}$$

$$e^1_t = \rho_1 e^1_{t-1} + \varepsilon^1_t \tag{8.4}$$

$$e^2_t = \rho_2 e^2_{t-1} + \varepsilon^2_t \tag{8.5}$$

The model comprises a dynamic IS curve, a Phillips Curve and a policy rule with smoothing. There are three shocks, two of which are persistent. This we need to write in the general algebraic linear state-space form:

$$E\begin{bmatrix} z_t \\ x^e_{t+1} \end{bmatrix} = A\begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + B\varepsilon_t$$

We map our variables to their algebraic equivalent as $(z_t,\, x_t) = ((e_t^1,\, e_t^2,\, i_t),\, (y_t,\, \pi_t))$. Then the model in state-space form but including the matrix E is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -\frac{1}{\sigma} & 1 & \frac{1}{\sigma} \\ 0 & 1 & 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} e_t^1 \\ e_t^2 \\ i_t \\ y_{t+1}^e \\ \pi_{t+1}^e \end{bmatrix} = \begin{bmatrix} \rho_1 & 0 & 0 & 0 & 0 \\ 0 & \rho_2 & 0 & 0 & 0 \\ 0 & 0 & \gamma & 0 & (1-\gamma)\delta \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\kappa & 1 \end{bmatrix} \begin{bmatrix} e_{t-1}^1 \\ e_{t-1}^2 \\ i_{t-1} \\ y_t \\ \pi_t \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon_t^1 \\ \varepsilon_t^2 \\ \varepsilon_t^3 \end{bmatrix}$$

Anyone wanting to code up solutions should familiarize themselves with this before continuing.

8.2.1 Coding the model

Before we begin coding this in R, load the `tidyverse` libraries so we can do impulse responses with our usual tool kit and then we can forget about it.

```
library(tidyverse)
```

Set the model parameters

```
nf    <- 2
ns    <- 5
ne    <- 3
np    <- ns-nf

beta  <- 0.99   # Discount factor
sigma <- 2.0    # Elas. substitution
kappa <- 0.075  # Slope PC
delta <- 1.5    # Inflation feedback
gamma <- 0.75   # Smoothing
rho_1 <- 0.9    # AR1
rho_2 <- 0.8    # AR1
Omega <- diag(c(0.33,0.33,0.33)) # SE of 3 shocks
```

Now define the model matrices 'long hand' and some variable names, which we put in `labels`.

```
labels <- c("e^1","e^2","i","y","pi")

E <- matrix(0,ns,ns)
A <- matrix(0,ns,ns)
B <- diag(1,ns,ne)

# Now put the equations in matrix form
```

```
diag(E[1:2,1:2]) <- 1
diag(A[1:2,1:2]) <- c(rho_1, rho_2)

E[3,3]             <- 1
E[4,c(1, 3, 4, 5)] <- c(1, -1/sigma, 1, 1/sigma)
E[5,c(2, 5)]       <- c(1, beta)

A[3,c(3, 5)]       <- c(gamma, (1-gamma)*delta)
A[4,4]             <- 1
A[5,c(4,5)]        <- c(-kappa, 1)
```

where for example, E and A are

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & -0.5 & 1 & 0.5 \\ 0 & 1 & 0 & 0 & 0.99 \end{bmatrix} \tag{8.6}$$

$$A = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0 & 0.375 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -0.075 & 1 \end{bmatrix} \tag{8.7}$$

Calculate the reduced form state-space model

$$\begin{bmatrix} z_t \\ x^e_{t+1} \end{bmatrix} = C \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + D\varepsilon_t \tag{8.8}$$

which is done in R very simply as

```
C <- solve(E,A)
D <- solve(E,B)
```

Why can't we solve this for impulse responses?

The following function simulates the impulse responses of a model in a loop within a loop[1] and returns the time series in a suitably organised data frame.

```
impulse_responses <- function(P, Q, Omega, labels, T) {
  s   <- matrix(0, ncol(Q), 1)
  z   <- matrix(0, nrow(Q), T)
  rownames(z) <- labels
  dza <- NULL
```

[1] Sometimes a loop is the right way to do something.

```
  for (j in 1:ncol(Q)) {
    s[j]  <- Omega[j,j]
    z[,1] <- Q %*% s
    for (i in 1:(T-1)) {
      z[,i+1] <- P %*% z[,i]
    }
    s[j] <- 0
    dz <- as_tibble(t(z)) %>%
      mutate(Period = 1:T, Shock = paste0("epsilon^",j))
    dza <- bind_rows(dza,dz)
  }
  return(dza)
}
```

A function to plot the impulses will be useful, so we create one.

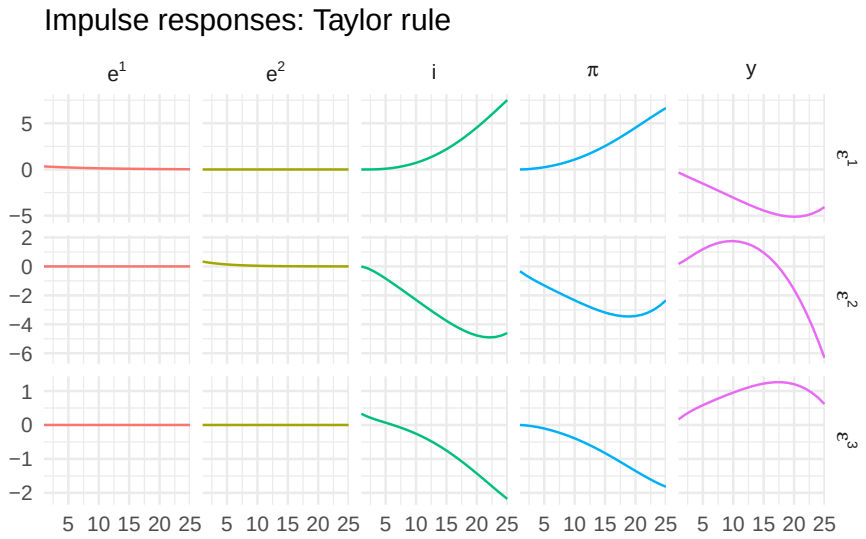
```
response_plot <- function(series, title) {
  return(pivot_longer(series, cols = -c(Period,Shock), names_to="Var", values_to = '
           ggplot() +
           geom_line(aes(x=Period, y=Val, group=Shock, colour=Var), show.legend=FALS
           facet_grid(Shock~Var, scales="free", labeller=label_parsed) +
           scale_x_continuous(expand=c(0,0)) +
           theme_minimal() +
           labs(title=title, x="",y=""))
}
```

Call the impulse response function using the model C and D.

```
T <- 25
z <- impulse_responses(C, D, Omega, labels, T)
```

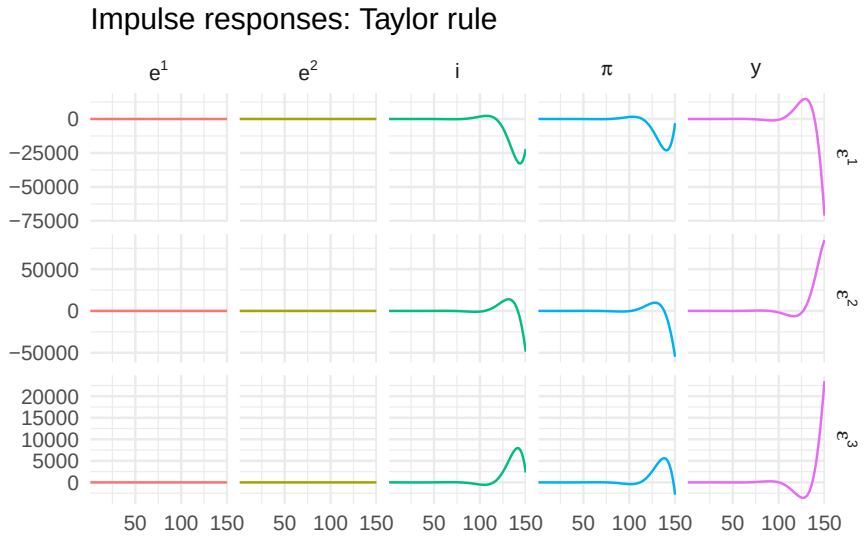
and plot

```
response_plot(z, "Impulse responses: Taylor rule")
```



Oh! That's not looking good. Let's try a few more periods.

```
T <- 150
z <- impulse_responses(C, D, Omega, labels, T)
response_plot(z, "Impulse responses: Taylor rule")
```



This is clearly exploding. But it's rational – we're solving forward so expectations are always fulfilled. This is a key the insight of the early rational expectations modellers – rational isn't enough, non-explosive is necessary too. Fortunately we know how to find this.

8.3 Blanchard and Kahn (1980)

To solve this model to give a unique *stable* rational expectations equilibrium, we appeal to the following. Consider the eigenvalue decomposition

$$MC = \Lambda M$$

where Λ is a diagonal matrix of *eigenvalues* in increasing absolute value and M is a non-singular matrix of *left eigenvectors*. Note that computer routines (including the one in R) usually calculate *right eigenvectors* such that $CV = V\Lambda$ and that $M = V^{-1}$, so be aware of this in what follows.

We can *diagonalise* C and write it as $C = M^{-1}\Lambda M$. So pre-multiplying the reduced form model by M gives

$$M \begin{bmatrix} z_t \\ x^e_{t+1} \end{bmatrix} = \Lambda M \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + MD\varepsilon_t$$

Blanchard and Kahn (1980) (following Vaughan (1970)) show uniqueness requires as many unstable eigenvalues as jump variables. To see this, define

$$\begin{bmatrix} \xi^s_{t-1} \\ \xi^u_t \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix}$$

Write the normalized model as

$$\begin{bmatrix} \xi^s_t \\ \xi^u_{t+1} \end{bmatrix} = \begin{bmatrix} \Lambda_s & 0 \\ 0 & \Lambda_u \end{bmatrix} \begin{bmatrix} \xi^s_{t-1} \\ \xi^u_t \end{bmatrix} + \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} D\varepsilon_t$$

where the eigenvalues are split into stable (Λ_s) and unstable (Λ_u). If we ignore the stochastic bit for a moment

$$\begin{bmatrix} \xi^s_t \\ \xi^u_{t+1} \end{bmatrix} = \begin{bmatrix} \Lambda_s & 0 \\ 0 & \Lambda_u \end{bmatrix} \begin{bmatrix} \xi^s_{t-1} \\ \xi^u_t \end{bmatrix}$$

We seek a non-explosive solution, and this turns out to be easy to find using the following

- The dynamics of ξ^u_t are determined by Λ_u and nothing else;
- If they don't start at 0 they must explode;
- This implies they must start at 0 and are always 0.

Thus the definition of the canonical variables necessarily implies

$$\begin{bmatrix} \xi^s_{t-1} \\ 0 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix}$$

From this it is clear that the jump variables themselves are only on the saddle path if

$$M_{21} z_{t-1} + M_{22} x_t = 0$$

The rational solution implies that the jump variables are linearly related to the predetermined ones through

$$x_t = -M_{22}^{-1} M_{21} z_{t-1} \tag{8.9}$$
$$= N z_{t-1} \tag{8.10}$$

We'll deal with the shocks in a moment.

How do we do this in R? First, find the eigenvalue decomposition of C using

```
m <- eigen(C, symmetric=FALSE)
```

which yields

```
eigen() decomposition
$values
[1] 1.0715518+0.092734i 1.0715518-0.092734i 0.9000000+0.000000i
[4] 0.8000000+0.000000i 0.6548762+0.000000i

$vectors
                       [,1]                  [,2]          [,3]           [,4]
[1,]  0.0000000+0.0000000i  0.0000000+0.0000000i 0.2854942+0i  0.00000000+0i
[2,]  0.0000000+0.0000000i  0.0000000+0.0000000i 0.0000000+0i  0.09783896+0i
[3,]  0.1599159-0.5089425i  0.1599159+0.5089425i 0.7830500+0i  0.49622464+0i
[4,] -0.6991064+0.0000000i -0.6991064+0.0000000i 0.4552131+0i -0.86012270+0i
[5,]  0.2629802-0.3968579i  0.2629802+0.3968579i 0.3132200+0i  0.06616328+0i
              [,5]
[1,]  0.0000000+0i
[2,]  0.0000000+0i
[3,]  0.6351203+0i
[4,] -0.7554249+0i
[5,] -0.1611069+0i
```

However this calculates *right* eigenvectors. We will need to invert it for left ones. Given the number of jump variables in the model satisfies the Blanchard-Kahn conditions of as many unstable roots (1.072+0.093i, 1.072-0.093i) as jump variables (2) we can calculate the reaction function from the eigenvectors

```
iz <- 1:np
ix <- (np+1):ns
M  <- solve(m$vectors[,ns:1])        # Invert & reverse order for increasing abs val
N  <- -Re(solve(M[ix,ix], M[ix,iz])) # Drop tiny complex bits (if any)
```

where `iz` are the indices of the first `np` variables and `ix` those of the remaining `nf` ones.

8.3.1 Stochastic part

What about the shocks? Assume the stochastic reaction function is

$$x_t = Nz_{t-1} + G\varepsilon_t$$

Following Andrew P. Blake (2004), note that $x^e_{t+1} = Nz_t$ as the expected value of $\varepsilon_{t+1} = 0$, meaning we can write

$$Nz_t = C_{21}z_{t-1} + C_{22}x_t + D_2\varepsilon_t$$

or

$$N\left(C_{11}z_{t-1} + C_{12}x_t + D_1\varepsilon_t\right) = C_{21}z_{t-1} + C_{22}x_t + D_2\varepsilon_t$$

Gathering terms we obtain

$$(C_{22} - NC_{12})x_t = (NC_{11} - C_{21})z_{t-1} + (ND_1 - D_2)\varepsilon_t$$

which implies

$$G = (C_{22} - NC_{12})^{-1}(ND_1 - D_2)$$

Notice it also implies $N = (C_{22} - NC_{12})^{-1}(NC_{11} - C_{21})$. It is this fixed point nature of the solution for N – which in turn implies the quadratic matrix equation $C_{21} = NC_{11} - C_{22}N + NC_{12}N$ – that means we need to use the Blanchard and Kahn (1980) method in the first place.

All of this means that

```
G <- solve((C[ix,ix] - N %*% C[iz,ix]), (N %*% D[iz,]- D[ix,]))
```

so for our model and parameters N and G are

$$N = \begin{bmatrix} 4.8568 & -2.7586 \\ -1.1894 & 1.7929 \\ 1.9628 & -0.2537 \end{bmatrix} \tag{8.11}$$

$$G = \begin{bmatrix} 5.3964 & -3.4483 \\ -1.5859 & 1.9921 \\ 2.4535 & -0.3382 \end{bmatrix} \tag{8.12}$$

The 'fixed point' check is that the following should be the same as N

```
solve((C[ix,ix] - N %*% C[iz,ix]), (N %*% C[iz,iz]- C[ix,iz]))
```

```
        [,1]      [,2]       [,3]
[1,] 4.85680 -2.758647 -1.1894200
[2,] 1.79286  1.962790 -0.2536635
```

which it is.

The solved model is finally

$$\begin{bmatrix} z_t \\ x_t \end{bmatrix} = \begin{bmatrix} C_{11} + C_{12}N & 0 \\ N & 0 \end{bmatrix} \begin{bmatrix} z_{t-1} \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} D_1 + C_{12}G \\ G \end{bmatrix} \varepsilon_t \tag{8.13}$$

$$= P \begin{bmatrix} z_{t-1} \\ x_{t-1} \end{bmatrix} + Q\varepsilon_t \tag{8.14}$$

which can be coded as

```
P  <- cbind(rbind((C[iz,iz] + C[iz,ix] %*% N), N), matrix(0,ns,nf))
Q  <- rbind(D[iz,] + C[iz,ix] %*% G, G)
```

8.3.2 Digression – right eigenvector version

It turns out that we could use the output from the standard eigenvalue/vector routine directly by exploiting the following. This time, let M be the matrix of *right eigenvectors* so

$$CM = M\Lambda \text{ or } C = M\Lambda M^{-1}$$

and

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \xi^s_{t-1} \\ \xi^u_t \end{bmatrix} = \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix}$$

Written this way around, if $\xi^u_t = 0 \ \forall \ t$ then (again ignoring stochastics)

$$M_{11}\xi^s_{t-1} = z_{t-1}, \ M_{21}\xi^s_t = x_t$$

$$\Rightarrow x_t = M_{21}M_{11}^{-1}z_t$$

so

```
M <- m$vectors[,ns:1]             # Don't invert as already right vectors, but reorder
Re(M[ix,iz] %*% solve(M[iz,iz])) # Again, drop tiny complex bits
```

```
        [,1]      [,2]       [,3]
[1,] 4.85680 -2.758647 -1.1894200
[2,] 1.79286  1.962790 -0.2536635
```

The result is identical. This method is particularly useful if there are fewer predetermined variables than jumps as the matrix we need to invert is of the same dimension as the predetermined variables this way round.

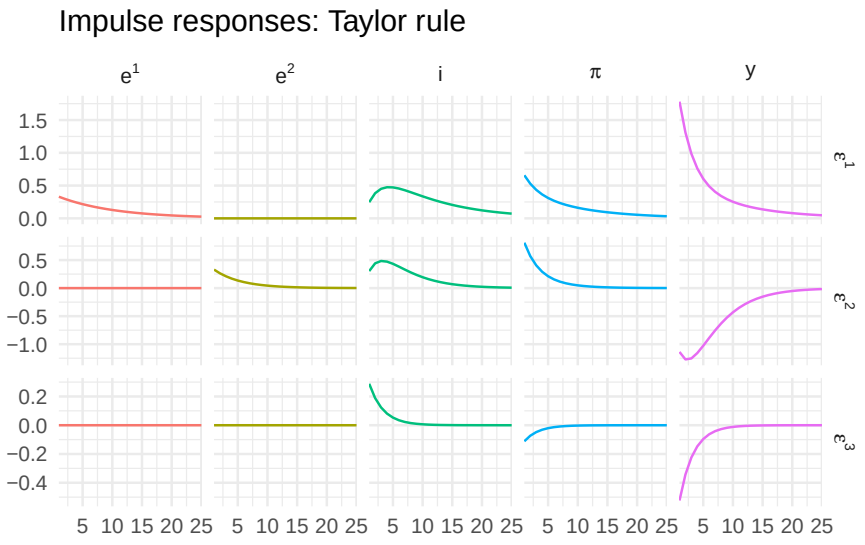
8.3.3 Impulse responses

We now call the impulse response function using the model solved for rational expectations.

```
T <- 25
z <- impulse_responses(P, Q, Omega, labels, T)
```

Now plot these responses

```
response_plot(z, "Impulse responses: Taylor rule")
```



Now, that looks better! It is no longer explosive. It also makes complete economic sense, which you can verify by going through the dynamics of the different demand, supply and monetary shocks.

8.4 Generalized solution

Sometimes for a model E is singular. A more general solution was proposed by Klein (2000), that doesn't require E to be non-singular. This uses a generalized Schur decomposition instead of an eigenvalue one and is applied to the structural model represented by the matrix pencil (A, E), and is considered much more numerically stable (see Pappas, Laub, and Sandell (1980)). The generalized Schur form of (A, E) is (QTZ', QSZ'), so we can write the model as

$$E \begin{bmatrix} z_t \\ x^e_{t+1} \end{bmatrix} \equiv QTZ' \begin{bmatrix} z_t \\ x^e_{t+1} \end{bmatrix} \equiv QT \begin{bmatrix} \xi^s_t \\ \xi^u_{t+1} \end{bmatrix}$$

and

$$A \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} \equiv QSZ' \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} \equiv QS \begin{bmatrix} \xi^s_{t-1} \\ \xi^u_t \end{bmatrix}$$

so the model pre-multiplied by Q' is

$$T \begin{bmatrix} \xi^s_{t+1} \\ \xi^u_{t+1} \end{bmatrix} = S \begin{bmatrix} \xi^s_t \\ \xi^u_t \end{bmatrix} + Q'B\varepsilon_t$$

We use the function `gqz` from the library `geigen` for this

```
d <- geigen::gqz(A, E, sort="S") # Option "S" puts the stable roots first
```

We can check that this is actually saddle path using `gevalues()` to get all the eigenvalues from the generalized Schur decomposition, and the unstable ones are

```
e <- geigen::gevalues(d)
e[abs(e) > 1]
```

```
[1] 1.071552+0.092734i 1.071552-0.092734i
```

The number of *stable* roots is returned in `d$sdim` which is 3.

We then modify our solution function to calculate `Ns` and `Gs` using the matrix `Z` and a generalized version of the formula for G and calculate the reduced form model `Ps` and 'Q which are

$$N_s = Z_{21}Z_{11}^{-1} \tag{8.15}$$
$$H = (E_{11} + E_{12}N_s)^{-1} \tag{8.16}$$
$$W = (E_{21} + E_{22}N_s)H \tag{8.17}$$
$$G_s = (A_{22} - WA_{12})^{-1}(WB_1 - B_2) \tag{8.18}$$
$$P_s = H(A_{11} + A_{12}N_s) \tag{8.19}$$
$$Q_s = H(B_1 + A_{12}G_s) \tag{8.20}$$

Verify this yourself with a bit of matrix algebra!

The R code for this is

```
solveGenBK <- function(E,A,B,n) {
  d  <- geigen::gqz(A, E, sort="S")
  np <- d$sdim
  ns <- nrow(E)
  print(paste("Number of unstable roots is", ns-np))
  if (n == np) {
    iz <- 1:n
    ix <- (n+1):ns
    Ns <- d$Z[ix,iz] %*% solve(d$Z[iz,iz])
    H  <- solve(E[iz,iz] + E[iz,ix] %*% Ns)
    W  <- (E[ix,iz] + E[ix,ix] %*% Ns) %*% H
    Gs <- solve((A[ix,ix] - W %*% A[iz,ix]), (W %*% B[iz,] - B[ix,]))
    As <- H %*% (A[iz,iz] + A[iz,ix] %*% Ns)
    Bs <- H %*% (B[iz,] + A[iz,ix] %*% Gs)
    return(list(P=cbind(rbind(As,Ns),matrix(0,ns,ns-n)), Q=rbind(Bs, Gs)))
    }
  else {
    return(-1)
    }
}
```

Using this on our original model gives

```
S  <- solveGenBK(E,A,B,np)
```

```
[1] "Number of unstable roots is 2"
```

```
Ps <- S$P
Qs <- S$Q
```

and comparing `Ps` and `Qs` with `P` and `Q` obtained using Blanchard-Kahn we find

```
round(max(abs(P-Ps), abs(Q-Qs)), 12)
```

```
[1] 0
```

They are, as expected, the same – at least up to 12 decimal places, which should be enough.

8.5 Singular models: optimal policy

However, this is an easy test. What we need is to use a model that can't be solved using the BK method. Under optimal policy, the interest rate instrument rule is replaced with a targeting rule, so that

$$\pi_t = -\mu \Delta y_t - \varepsilon_t^3$$

for some value of μ that reflects the optimal trade-off between output (gap) growth and inflation, and we've included a disturbance which we can loosely describe as a monetary policy shock. We modify the model above by dropping the Taylor rule in favour of the targeting rule. This requires a lagged value of y to be created. The following does the trick

```
nf <- 2
ne <- 3
ns <- 6      # One extra state
np <- ns-nf
mu <- 0.75   # Representative trade-off

labels <- c("e^1","e^2","ylag","i","y","pi") # New variable order

E <- matrix(0,ns,ns)
A <- E
B <- matrix(0,ns,ne)
B[1,1] <- 1
B[2,2] <- 1
B[4,3] <- -1

diag(E[1:3,1:3]) <- 1
diag(A[1:2,1:2]) <- c(rho_1, rho_2)
A[3,5]           <- 1

E[4,3]           <- 1
A[4,c(3, 6)]     <- c(1, -1/mu)

E[5,c(1, 4, 5, 6)] <- c(1, -1/sigma, 1, 1/sigma)
A[5,5]           <- 1

E[6,c(2, 6)]     <- c(1, beta)
A[6,c(5, 6)]     <- c(-kappa, 1)
```

The new E and A system matrices are then

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -0.5 & 1 & 0.5 \\ 0 & 1 & 0 & 0 & 0 & 0.99 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1.333 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -0.075 & 1 \end{bmatrix}$$

Now we have a singular model. The matrix E is clearly singular as rows 3 and 4 are identical. But we have a problem using the code above. To use it we need the matrices H and $(A_{22} - WA_{21})$ to be non-singular. What to do?

There are two ways out. Klein (2000) gives a solution that depends on the decomposed matrix pencil, which is what is typically implemented, but you don't actually need it although it is easiest. Instead, all you need to do is reorder the equations.

The real problem is that with a targeting rule that doesn't include the interest rate, and the interest rate is now only determined by the IS curve. But we can swap the location of any two rows of the model arbitrarily. If we swap the positions of the equations for the IS curve and the targeting rule (rows 4 and 5) using the following

```
E[4:5,] <- E[5:4,]
A[4:5,] <- A[5:4,]
B[4:5,] <- B[5:4,]
```

then the model is unchanged but now we have

$$E_{11} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -0.5 \end{bmatrix}$$

so $E_{11} + E_{12}N$ is likely non-singular (it is). Also, note after the re-ordering A_{22} is

$$A_{22} = \begin{bmatrix} 0 & -1.33 \\ -0.07 & 1 \end{bmatrix}$$

which is guaranteed non-singular for zero W. We can now proceed as before. First, check for saddle path stability

```
e <- geigen::gevalues(geigen::gqz(A, E, sort="S"))
e[abs(e) > 1]
```

```
[1] 1.378195      Inf
```

which confirms that it has a unique saddle path stable solution. This is

```
So <- solveGenBK(E,A,B,np)
```

```
[1] "Number of unstable roots is 2"
```

```
Po <- So$P
Qo <- So$Q
```

The solved model is then

```
Po
```

```
     [,1]      [,2]          [,3] [,4] [,5] [,6]
[1,]  0.9  0.000000  0.000000e+00    0    0    0
[2,]  0.0  0.800000  6.986592e-17    0    0    0
[3,]  0.0 -1.863455  7.329156e-01    0    0    0
[4,]  1.8 -1.241330 -2.446879e-01    0    0    0
[5,]  0.0 -1.863455  7.329156e-01    0    0    0
[6,]  0.0  1.397591  2.003133e-01    0    0    0
```

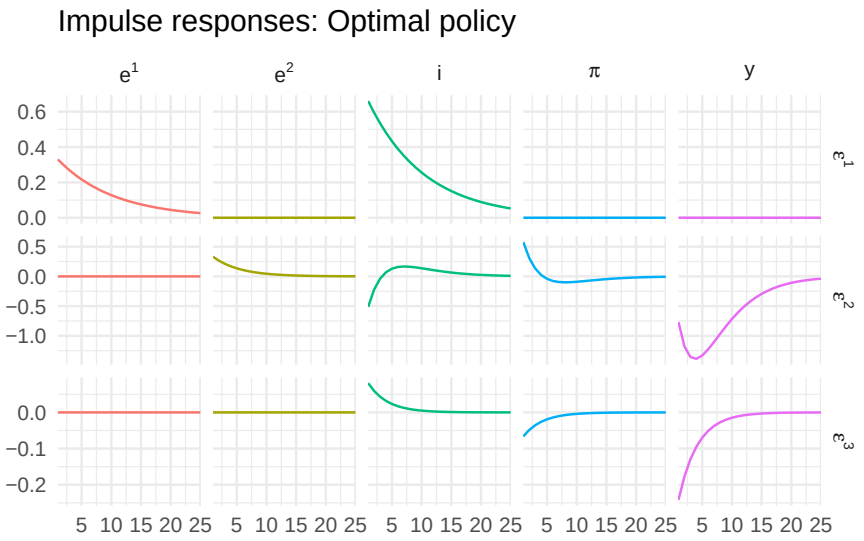
```
Qo
```

```
     [,1]      [,2]          [,3]
[1,]    1  0.000000  0.000000e+00
[2,]    0  1.000000 -6.986592e-17
[3,]    0 -2.329318 -7.329156e-01
[4,]    2 -1.551663  2.446879e-01
[5,]    0 -2.329318 -7.329156e-01
[6,]    0  1.746989 -2.003133e-01
```

8.5.1 Optimal impulse responses

We can now simulate the model under optimal policy and plot using

```
zo <- impulse_responses(Po, Qo, Omega, labels, T) %>%
  select(-ylag) # Drop duplicate series
response_plot(zo, "Impulse responses: Optimal policy")
```



8.6 Dummy jumps

But this isn't the only way to get this to work. Effectively what we just did was create an extra predetermined variable and reorder the system to give us non-singularity. What if instead of including an unused i_{t-1} on the right hand side, we instead include an unused i^e_{t+1} on the left hand side? So we swap to having one more jump variable, one less predetermined one?

Compare the following to the previous model. When we pick out the interest rate we do so on the right hand side of the matrix equation, not the left as before.

```
ns <- 6       # One extra state
nf <- 3       # And one extra jump
np <- ns-nf
labels <- c("e^1","e^2","ylag","i","y","pi") # New variable order

E <- matrix(0,ns,ns)
A <- E
```

```
B <- matrix(0,ns,ne)
B[1,1] <- 1
B[2,2] <- 1
B[4,3] <- -1

diag(E[1:3,1:3]) <- 1
diag(A[1:2,1:2]) <- c(rho_1, rho_2)
A[3,5]           <- 1

E[4,3]           <- 1
A[4,c(3, 6)]     <- c(1, -1/mu)

E[5,c(1, 5, 6)]  <- c(1, 1, 1/sigma) # One less coefficient
A[5,c(4, 5)]     <- c(1/sigma, 1)    # One more - nothing else changes

E[6,c(2, 6)]     <- c(1, beta)
A[6,c(5, 6)]     <- c(-kappa, 1)
```

This is still a singular model, as we can see from

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0.5 \\ 0 & 1 & 0 & 0 & 0 & 0.99 \end{bmatrix}$$

with column 4 all zeros. Is *this* model saddle path stable?

```
e <- geigen::gevalues(geigen::gqz(A, E, sort="S") )
e[abs(e) > 1]
```

```
[1]      -Inf 1.378195       Inf
```

Again, it is with an extra unstable root for the extra jump variable. We could simplify the solution. As that top left 3 by 3 block, E_{11}, is the identity matrix and E_{12} is all zeros this `Ei` is always an identity matrix. However, here we simply re-use `solveGenBG`

```
So2 <- solveGenBK(E,A,B,np)
```

```
[1] "Number of unstable roots is 3"
```

```
Po2 <- So2$P
Qo2 <- So2$Q
```

Now the solved model is

```
Po2
```

```
     [,1]      [,2]       [,3] [,4] [,5] [,6]
[1,]  0.9  0.000000  0.0000000    0    0    0
[2,]  0.0  0.800000  0.0000000    0    0    0
[3,]  0.0 -1.863455  0.7329156    0    0    0
[4,]  1.8 -1.241330 -0.2446879    0    0    0
[5,]  0.0 -1.863455  0.7329156    0    0    0
[6,]  0.0  1.397591  0.2003133    0    0    0
```

```
Qo2
```

```
     [,1]      [,2]       [,3]
[1,]    1  0.000000  0.0000000
[2,]    0  1.000000  0.0000000
[3,]    0 -2.329318 -0.7329156
[4,]    2 -1.551663  0.2446879
[5,]    0 -2.329318 -0.7329156
[6,]    0  1.746989 -0.2003133
```

which is actually identical to our previous solution. This is because I have preserved the order of the solved-out variables, and shows that the swap from a predetermined to a jump variable is completely arbitrary.

8.7 Substituting out

But even this doesn't exhaust the possible re-parametrisations of the model. We can reduce the number of jump variables to 1 and find the same solution. There exist formal methods for reducing models (see King and Watson (2002)) but there is an obvious way to proceed here. From the targeting rule, it must be that

$$y^e_{t+1} = y_t - \frac{1}{\mu}\pi^e_{t+1}$$

as the expected shock is zero. This means the IS curve can be rewritten

$$y_t = y_t - \frac{1}{\mu}\pi^e_{t+1} - \frac{1}{\sigma}\left(i_t - \pi^e_{t+1}\right) + e^1_t$$

implying

$$i_t = \left(1 - \frac{\sigma}{\mu}\right)\pi^e_{t+1} + \sigma e^1_t$$

This is the required interest rate consistent with the targeting rule holding. Now the only jump variable is the inflation rate as we have eliminated the expected output gap.

$$y_t = y_{t-1} - \frac{1}{\mu}\pi_t + \frac{1}{\mu}\varepsilon^3_t \tag{8.21}$$

$$\pi_t = \beta\pi^e_{t+1} + \kappa y_t + e^2_t \tag{8.22}$$

$$i_t = \left(1 - \frac{\sigma}{\mu}\right)\pi^e_{t+1} + \sigma e^1_t \tag{8.23}$$

$$e^1_t = \rho_1 e^1_{t-1} + \varepsilon^1_t \tag{8.24}$$

$$e^2_t = \rho_2 e^2_{t-1} + \varepsilon^2_t \tag{8.25}$$

We can code this

```
ns <- 5       # Back to 5 states
nf <- 1       # Now only one jump
np <- ns-nf

labels <- c("e^1","e^2","i","y","pi") # Lose a y

E <- matrix(0,ns,ns)
A <- E
B <- matrix(0,ns,ne)
B[1,1] <- 1
B[2,2] <- 1
B[4,3] <- -1

diag(E[1:4,1:4]) <- 1
diag(A[1:2,1:2]) <- c(rho_1, rho_2)

E[3,c(1, 3, 5)]  <- c(-sigma, 1, sigma/mu-1)

A[4,c(4,5)]      <- c(1, -1/mu)

E[5,c(2, 4, 5)]  <- c(1, kappa, beta)
A[5,5]           <- 1
```

and solve it using

```
Ss <- solveGenBK(E,A,B,np)
```

```
[1] "Number of unstable roots is 1"
```

```
Ps <- Ss$P
Qs <- Ss$Q
```

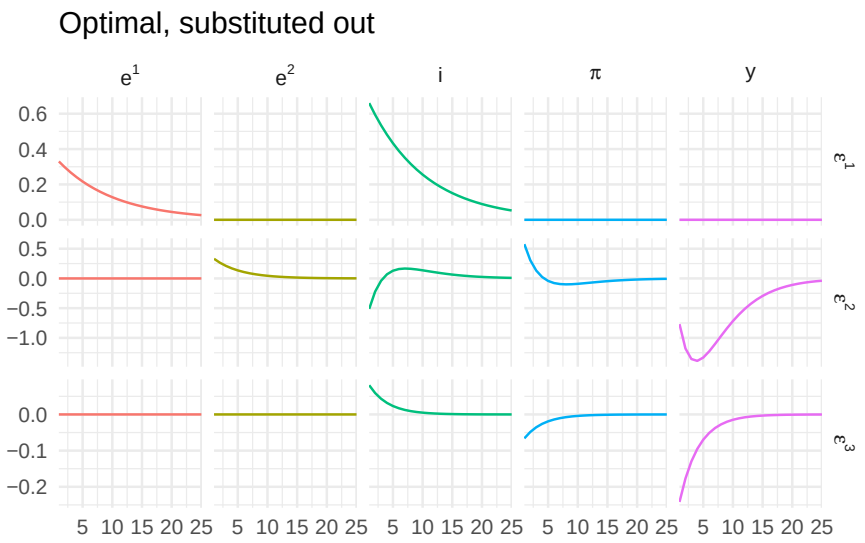
Compare the realized of `Ps`

```
Ps
```

```
     [,1]      [,2] [,3]       [,4] [,5]
[1,]  0.9  0.000000    0  0.0000000    0
[2,]  0.0  0.800000    0  0.0000000    0
[3,]  1.8 -1.241330    0 -0.2446879    0
[4,]  0.0 -1.863455    0  0.7329156    0
[5,]  0.0  1.397591    0  0.2003133    0
```

with `Po` above, say. This is the most 'efficient' way of programming the model, in that we have only five states, and indeed the repeated behavioural equations we had before have disappeared in the reduced form solution. Just to confirm this, simulating and plotting this version gives

```
response_plot(impulse_responses(Ps,Qs,Omega,labels,T), "Optimal, substituted out")
```



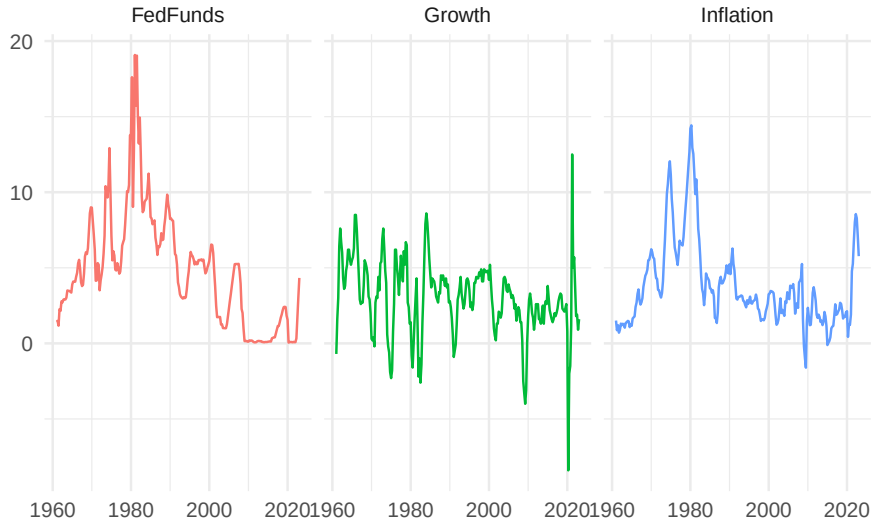
which are identical results to those above. But of course E is now invertible so we could solve this using the simplest Blanchard-Kahn variant. Try it!

Chapter 9

BVAR with dummies

9.1 Estimating BVARs using US data

We will the Fed Funds rate, annual GDP growth and annual CPI inflation data from FRED, retrieved 2023-05-27. These are:



We will build a variety and two and three variable BVARs. More details on the data are given below.