Lab No. 1 — State space models for SGPE

Advanced Time Series Econometrics Labs

Ping Wu (ping.wu@strath.ac.uk (mailto:ping.wu@strath.ac.uk))

Outline of today's lab

- The unmoored model (The local level model)
- The local linear trend model

Packages we will use in this lab

- dlm (https://cran.r-project.org/web/packages/dlm/dlm.pdf): Provides routines for Maximum likelihood, Kalman filtering and smoothing, and Bayesian analysis of Normal linear State Space models, also known as Dynamic Linear Models.
- tsm (https://github.com/KevinKotze/tsm/tree/master): Includes methods and tools specifically for state space models.

```
# Important packages described above
install.packages("dlm", repos = "https://cran.rstudio.com/",
    dependencies = TRUE)
devtools::install_github("KevinKotze/tsm")
```

The next step is to make sure that you can access the routines in this package by making use of the *library* command, which would need to be run regardless of the machine that you are using.

```
# Important packages described above
library(dlm)
library(tsm)
```

Part 0: Import data into R

In this lab, we will use the quarterly data set for the US CPI inflation from 1947Q2 to 2016Q1.

The first thing that we do is clear all variables from the current environment and close all the plots. This is performed with the following commands:

```
rm(list = ls())
graphics.off()
options(scipen = 9) # Avoid scientific notation
```

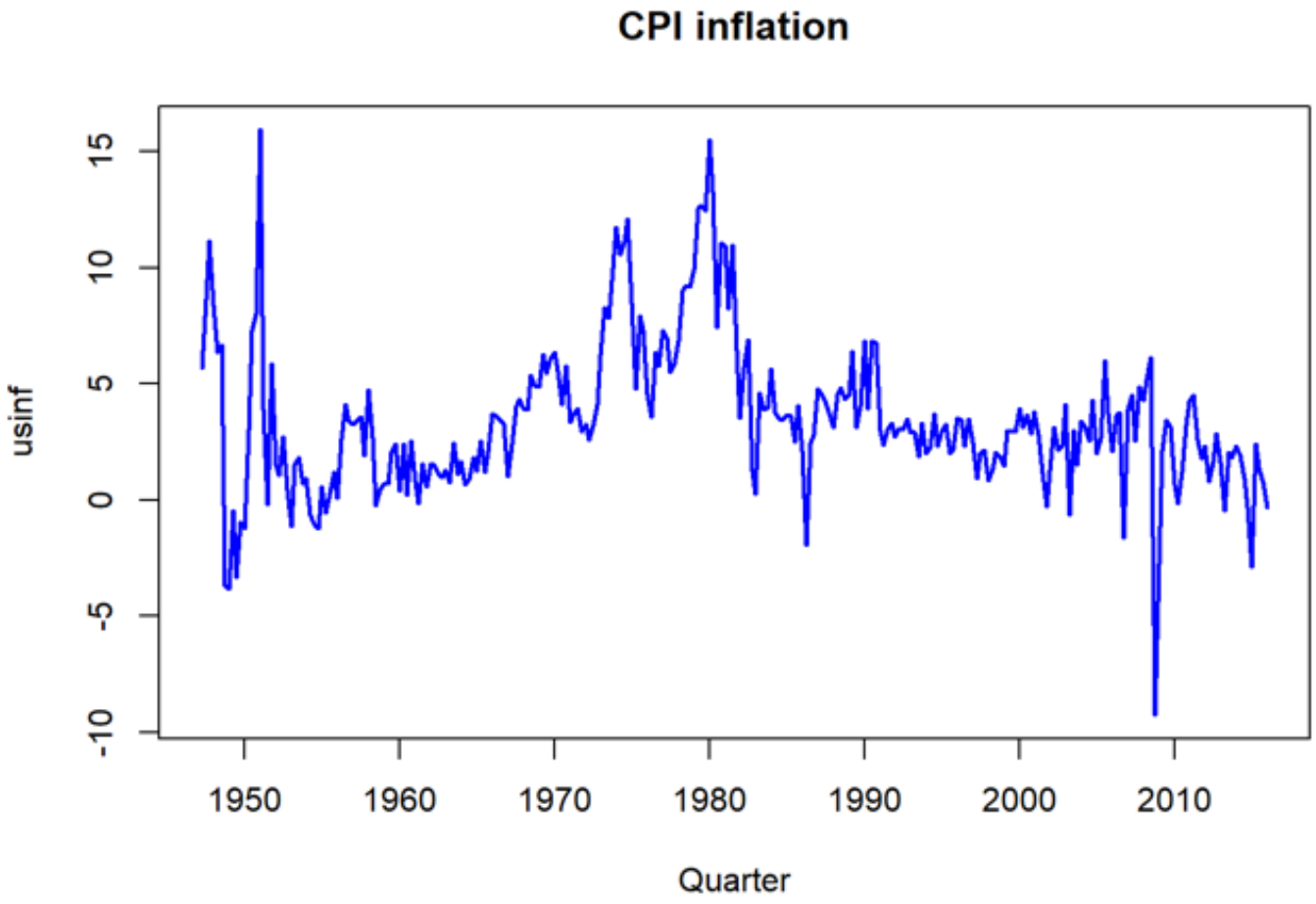
To import your downloaded data into R, you can:

```
# Allocate the macroeconomic variables to the object "usmacro"
if("inflation_data.csv" %in% list.files()){
  usinf <- read.table("inflation_data.csv", sep=",",header=T)
}else{ # file.choose() allows to choose the file interactively
  usinf <- read.table(file.choose(), sep=",",header=T)
}
# Specify "usinf" as time series object
usinf <- ts(usinf[,"CPIINFL"], end = c(2016, 1), frequency = 4)
```

Part I: Some descriptives

To make sure that the data has been imported correctly, we inspect a plot of the data:

```
ts.plot(usinf,
        main = "CPI inflation",    # Title of plot
        xlab = "Quarter",          # Label of x-axis
        col  = c("blue"),          # Define colours
        lty  = "solid",            # Define line types
        lwd  = 2)                  # Define line width)
```



Part II: The unmoored model (The local level model)

The first model that we are going to build is a local level model, which can be written as

$$y_t = \alpha_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V),$$
$$\alpha_t = \alpha_{t-1} + e_t, \quad e_t \sim \mathcal{N}(0, W).$$

The *dlm* package has a relatively convenient way to construct these models according to the order of the polynomial, where a first order polynomial is a local level model. We also need to allow for one parameter for the error in the measurement equation and a second parameter for the error in the state equation. These parameters are labelled *parm[1]* and *parm[2]*, respectively:

```
fn <- function(parm) {
    dlmModPoly(order = 1, dV = exp(parm[1]), dW = exp(parm[2]))
}
```

Thereafter, we are going to assume that the starting values for the parameters are small (i.e. zero) and proceed to fit the model to the data with the aid of maximum-likelihood methods:

```
fit <- dlmMLE(usinf, rep(0, 2), build = fn, hessian = TRUE)
fit$convergence # To see whether it has converged (0 means converged)
```

```
## [1] 0
```

The statistics for the variance-covariance matrix could be extracted from the model object with the use of the commands

```
mod <- fn(fit$par)
obs.error.var <- V(mod)                    # error variance in the measurement equat
ion
state.error.var <- W(mod)                  # error variance in the state equation
cat("V.variance", obs.error.var)           # To print
```

```
## V.variance 2.051627
```

```
cat("W.variance", state.error.var)
```

```
## W.variance 1.766865
```

We can then construct the likelihood function and generate statistics for the information criteria:

```
loglik <- dlmLL(usinf, dlmModPoly(1))
n.para <- 2                                  # Number of parameters
r.aic <- (2 * (loglik)) + 2 * (sum(n.para))  # dlmLL caculates the neg. LL
r.bic <- (2 * (loglik)) + (log(length(usinf))) * (n.para)
cat("AIC", r.aic)   # To print AIC
```

```
## AIC 814.0476
```

```
cat("BIC", r.bic)   # To print BIC
```

```
## BIC 821.2884
```

To extract information relating to the Kalman filter and smoother, we use the instructions:

```
filtered <- dlmFilter(usinf, mod = mod)
smoothed <- dlmSmooth(filtered)
```

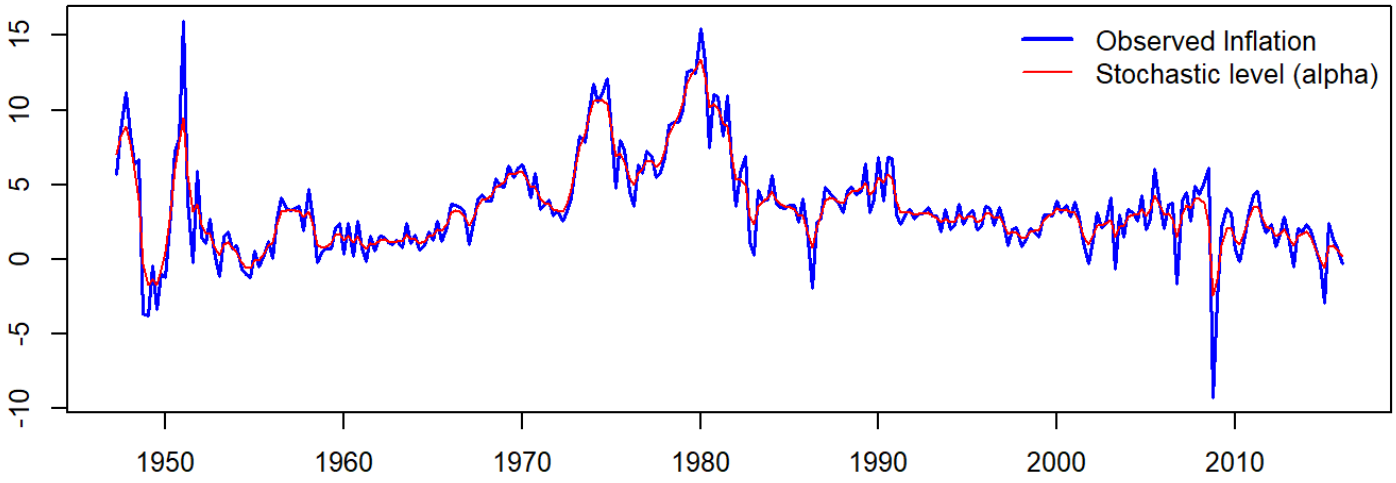
Thereafter, we can look at the results from the Kalman filter to derive the measurement errors. These have been stored in the *resids* object. We could also plot the smoothed values of the stochastic trend, which has been labeled *mu*

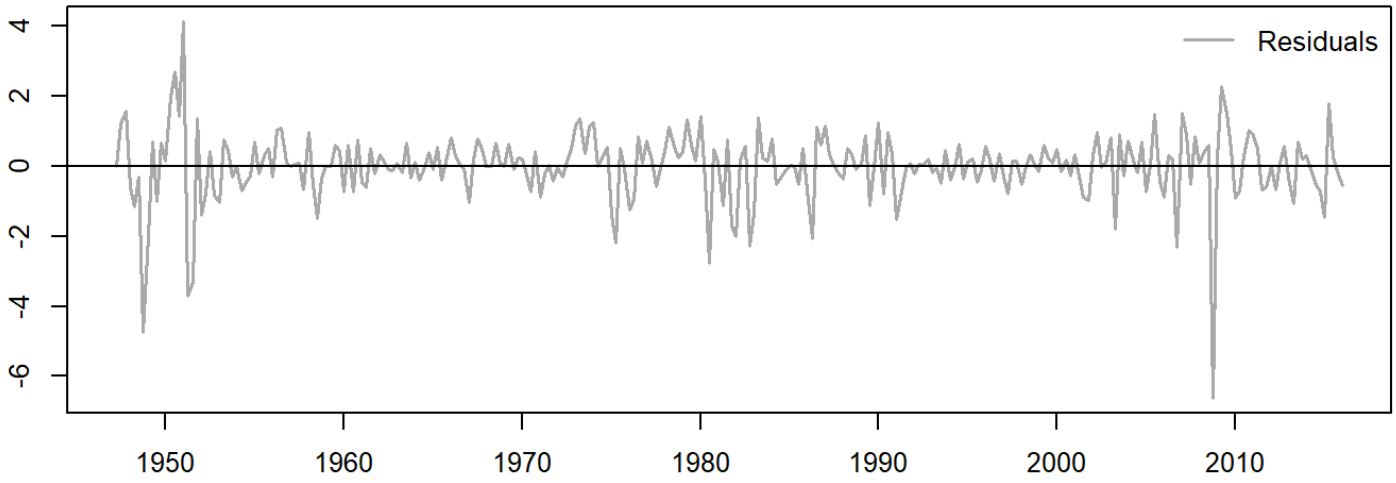
```
resids <- residuals(filtered, sd = FALSE)
mu <- dropFirst(smoothed$s)
mu.1 <- mu[1]
mu.end <- mu[length(mu)]
```

To plot the results on two graphs, we use the *mfrow* command to create two rows and one column for the plot area. Thereafter, the first plot includes data for the actual variable and the stochastic trend. The graph below depicts the measurement errors.

```
par(mfrow = c(2, 1), mar = c(2.2, 2.2, 1, 1), cex = 0.8)
plot.ts(usinf, col = "blue", xlab = "", ylab = "", lwd = 1.5)
lines(mu, col = "red")
legend("topright", legend = c("Observed Inflation", "Stochastic level (alpha)"),
    lwd = c(2, 1), col = c("blue", "red"), bty = "n")

plot.ts(resids, ylab = "", xlab = "", col = "darkgrey",
    lwd = 1.5)
abline(h = 0)
legend("topright", legend = "Residuals", lwd = 1.5, col = "darkgrey",
    bty = "n")
```





Part III: The local linear trend model

The second model that we are going to build is a local linear level model, which can be written as

$$
\begin{aligned}
y_t &= \alpha_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, V), \\
\alpha_{t+1} &= \alpha_t + \beta_t + e_t, \quad e_t \sim \mathcal{N}(0, W), \\
\beta_{t+1} &= \beta_t + u_t, \quad u_t \sim \mathcal{N}(0, Q).
\end{aligned}
$$

To estimate values for the parameters and unobserved components for this model, we could continue using the data that we have imported *usinf*.

However, in this case we need to incorporate an additional state equation for the slope of the model. This is a second order polynomial model and as there are two independent stochastic errors we also need to include this feature in the model.

```
fn <- function(parm) {
    dlmModPoly(order = 2, dV = exp(parm[1]), dW = exp(parm[2:3]))
}
```

We can then calculate the information criteria in the same way that we did previously, using the following commands

```
fit <- dlmMLE(usinf, rep(0, 3), build = fn, hessian = TRUE)
conv <- fit$convergence  # zero for converged

loglik <- dlmLL(usinf, dlmModPoly(2))
n.para <- 3  # Now we have three parameters to estimate
r.aic <- (2 * (loglik)) + 2 * (sum(n.para))  # dlmLL caculates the neg. LL
r.bic <- (2 * (loglik)) + (log(length(usinf))) * (n.para)

mod <- fn(fit$par)
obs.error.var <- V(mod)
state.error.var <- diag(W(mod))

cat("AIC", r.aic)
```

```
## AIC 935.5955
```

```
cat("BIC", r.bic)
```

```
## BIC 946.4567
```

```
cat("V.variance", obs.error.var)
```

```
## V.variance 2.025992
```

```
cat("W.variance", state.error.var) # You will see two values in W. The first one is for alpha equation, while the second is for beta equation.
```

```
## W.variance 1.811584 0.000000007030971
```

The values for the Kalman filter and smoother could also be extracted as before, but note that in this case we are also interested in the values of the stochastic slope, which is allocated to the *upsilon* object

```
filtered <- dlmFilter(usinf, mod = mod)
smoothed <- dlmSmooth(filtered)
resids <- residuals(filtered, sd = FALSE)
mu <- dropFirst(smoothed$s[, 1])
upsilon <- dropFirst(smoothed$s[, 2])
mu.1 <- mu[1]
mu.end <- mu[length(mu)]
ups.1 <- upsilon[1]
ups.end <- upsilon[length(mu)]
```

We are now going to plot the results for the three graphs underneath one another, using the commands

```
par(mfrow = c(3, 1), mar = c(2.2, 2.2, 1, 1), cex = 0.8)
plot.ts(usinf, col = "blue", xlab = "", ylab = "", lwd = 2)
lines(mu, col = "red")
legend("topright", legend = c("Observed Inflation", "Stochastic level (alpha)"),
    lwd = c(2, 1), col = c("blue", "red"), bty = "n")

plot.ts(upsilon, col = "orange", xlab = "", ylab = "",
    lwd = 2)
legend("topright", legend = "Slope (beta)", lwd = 2, col = "orange",
    bty = "n")

plot.ts(resids, ylab = "", xlab = "", col = "darkgrey",
    lwd = 2)
abline(h = 0)
legend("topright", legend = "Residuals", lwd = 2, col = "darkgrey",
    bty = "n")
```