Varying Coefficient Model

David Flatow

1. You have a set of p variables in the matrix X which represent some economic indicators measured over a number of years, and a response y also measured for the same years. You fit a linear model, but are criticized because you are told that the regime changes in time, and so should your model. You decide to let your regression coefficients (and intercept) change smoothly with time, a so-called varying coefficient model.

   (a) Since we want β_i to be a smooth function of time we express it in the basis space of a natural cubic spline of t:

   $$\beta_i(t) = \sum_{j=1}^{m} h_j(t)\theta_{ij} = h(t)^T\theta_i$$

   Then our time series model is,

   $$\begin{aligned} y(t) &= \beta_0(t) + \beta_1(t)x_1(t) + \beta_2(t)x_2(t) \\ &= h(t)^T\theta_0 + x_1(t)h(t)^T\theta_1 + x_2(t)h(t)^T\theta_2 \\ &= H(t)\theta \end{aligned}$$

   Where $H(t) = \begin{bmatrix} -h(t)^T- & -x_1(t)h(t)^T- & -x_2(t)h(t)^T- \end{bmatrix}$. And so,

   $$H = \begin{bmatrix} | & | & | \\ -h^T- & -x_1h^T- & -x_2h^T- \\ | & | & | \end{bmatrix}$$

   But this is a linear model in the design matrix H. And so we have,

   $$y = H\theta$$

   And we can estimate θ by OLS. Once we have our estimates $\hat{\theta}$ we can back out our time varying estimates for $\hat{\beta}_j(t)$. Specifically,

   $$\hat{\beta}_j = h^T\hat{\theta}_j$$

(b) We read in the data in vcdata.csv. There is a two-column x and a time variable t. We fit the varying coefficient model, and plot the three coefficient functions versus time. We compute the point-wise standard errors for each of these, and include a standard-error band (upper and lower) for each function.

We can compute the variance-covariance matrix of $\hat{\beta}_j$ as follows:

$$\Sigma_j = Var(\hat{\beta}_j) = Var(h^T \hat{\theta}_j) = h^T Var(\hat{\theta}_j) h$$

Note that $Var(\hat{\theta}_j)$ is sub-block of the usual variance-covariance matrix of $\hat{\theta}$ in the OLS estimation of $\hat{\theta}$ in the model $y = H\theta$. Specifically, it is the block $\Sigma_\theta[j : j+df, j : j+df]$. Then the point-wise standard errors of $\hat{\beta}_j$ are given by the square root of the diagonal of Σ_j and confidence bands are given by $\beta_j \pm 2\sqrt{diag(\Sigma_j)}$.

We can implement this in R as follows:

```
require(stats); require(graphics)
library(splines)

# helper function to repeat row
rep.row<-function(x,n){matrix(rep(x,each=n),nrow=n)}

# helper function to add confidence bands to plots
add_confidence_band = function(index, beta, cov, x, color){
  scaled_cov = x %*% cov %*% t(x)
  upper = beta + 2 * sqrt(diag(scaled_cov))
  lower = beta - 2 * sqrt(diag(scaled_cov))
  polygon( c(index, rev(index)) , c(upper , rev(lower)),
           col=color, border=NA)
}

# degrees of freedom for natural cubic spline bases
df = 7
n = 500

data = read.table('../data/vcdata.csv', sep=',', header=TRUE)

" We estimate a linear model in the natural cubic spline basis
  of time multiplied with each of our variables. Since we have
  3 variables (intercept, x1, x2) we will have (df * 3) basis
  functions where df is the degrees of freedom of each NCS basis.

  For example the second term in the linear model below is the
  NCS basis (df functions of t) of time each multiplied by x1.
  This term (expanded) will result in df terms in the linear
  model and we will have to estimate df coefficients for each of
```

```
  these bases."

t_basis = ns(data$t, df=df, intercept = T)
m = lm(y ~ 0 + t_basis + t_basis:x1 + t_basis:x2, data=data)
results = summary(m)

# coefficients for the intercept bases
int = results$coefficients[1:df, 1]

# coefficients for the beta_1 bases
b1 = results$coefficients[(df+1):(2*df), 1]

# coefficients for the beta_2 bases
b2 = results$coefficients[(2*df + 1):(3*df), 1]

# back out betas as a function of time
intercept = t_basis %*% int
beta_1 = t_basis %*% b1
beta_2 = t_basis %*% b2

# plot coefficient time series estimates
matplot(1:n, intercept, type="l",col="red",
       ylim=range(-1.5, 3.5),
       main="Time Varying Coefficients",
       xlab="t", ylab="coefficient value")

matplot(1:n, beta_1, type="l",col="blue", add=T)
matplot(1:n, beta_2, type="l",col="green", add=T)
legend("bottomleft", inset=.05, legend=c("intercept", "b1", "b2"),
      pch="-", col=c("red", "blue", "green"), horiz=TRUE)


# add confidence band to intercept coefficient time series
int_cov = vcov(results)[1:df, 1:df]
add_confidence_band(1:n, intercept, int_cov,
                  t_basis, rgb(1, 0, 0, 0.3))

# add confidence band to x1 coefficient time series
beta_1_cov = vcov(results)[(df + 1):(2*df),
                              (df + 1):(2*df)]
add_confidence_band(1:n, beta_1, int_cov,
                  t_basis, rgb(0, 0, 1, 0.3))

# add confidence band to x2 coefficient time series
beta_2_cov = vcov(results)[(2*df + 1):(3*df),
                              (2*df + 1):(3*df)]
```

```
add_confidence_band(1:n, beta_2, beta_2_cov,
                    t_basis, rgb(0, 1, 0, 0.3))
```