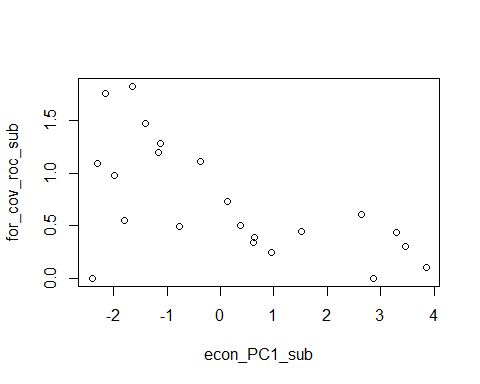
R Markdown test - non-linear models

Matt Nuttall

28 September 2018

# FlexParamCurve

I began by trying to use the FlexParamCurve package. I began with forest cover rate of change and the first principal component of the economic data. The plot below shows the data.



First I ran the modpar() function to estimate initial parameter estimates and parameter bounds

modpar(econ\_PC1\_sub, for\_cov\_roc\_sub, grp = NA, pn.options = "myoptions1")

## [1] modpar will attempt to parameterize your data using the following sequential procedures:  
## [1] (1) Extract parameter estimates for 8-parameter double-Richards curve in nls  
## [1] (2) Use getInitial to retrieve parameter estimates for 8-parameter double-Richards curve  
## [1] (3) Extract parameter estimates for 4-parameter Richards curve in   
## [1] (4) Use getInitial to retrieve parameter estimates for 4-parameter Richards curve  
## [1] if any approaches are successful, modpar will return these and terminate at that stage  
## [1]   
## [1] "Warning: positive optimization failed,using estimated parameters"  
## [1] (1) Status of 8-parameter double-Richards curve fit in nls:  
## [1] ....8 parameter nls fit failed  
## [1] (2) Status of 8-parameter double-Richards getInitial call  
## [1] "Warning: positive optimization failed,using estimated parameters"  
## [1] ....8-parameter getInitial successful

## $Asym  
## [1] 1.184295  
##   
## $K  
## [1] -3.002118  
##   
## $Infl  
## [1] -0.2150298  
##   
## $M  
## [1] 3.993167  
##   
## $RAsym  
## [1] -0.5881517  
##   
## $Rk  
## [1] 2.872548  
##   
## $Ri  
## [1] 3.45588  
##   
## $RM  
## [1] 0.8933009

This produced estimates for 8 parameters. Next I ran the two model selection tools - mod.compare and mod.step

The function struggled to fit any models, except model 1 which has 8 parameters (which seems like too many to me?). The next step is to use that base model to fit another model with the parameter estimates that modpar() produced:

FCroc.ecp1.mod1 <- nls(for\_cov\_roc\_sub ~ SSposnegRichards(econ\_PC1\_sub,   
 Asym = Asym, K = K, Infl = Infl, M = M,   
 RAsym = RAsym, Rk = Rk, Ri = Ri, RM = RM, modno = 1,   
 pn.options = "myoptions1"),  
 control = nls.control(maxiter = 1000))

But this throws up the error

Error in nls(for\_cov\_roc\_sub ~ SSposnegRichards(econ\_PC1\_sub, Asym = Asym, : singular gradient

Which I have read is more than likely an issue with the model itself, which is a bit odd seeing as it’s an in-built model within FlexParamCurve. I am wondering whether there are too few data points for FlexParamCurve?

# Custom models

## Forest cover rate of change ~ econ\_PC1

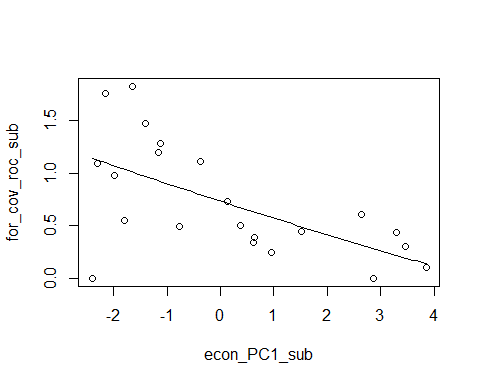
As I couldn’t get FlexParamCurve to work (and there is a sparcity of online help for trouble shooting with it), I decdided to try and build my own models. I did quite a lot of reading, particulalry the Crawley book, and based on the shape of my data I had a go with 2 parameter and 3 parameter asymptotic exponential models and a biexponential model.

### 3 parameter asymptotic exponential

First I used R’s self starting model SSasymp, which has the equation

mod.assym <- nls(for\_cov\_roc\_sub ~ SSasymp(econ\_PC1\_sub,Asym,RO,lrc))

##   
## Formula: for\_cov\_roc\_sub ~ SSasymp(econ\_PC1\_sub, Asym, RO, lrc)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Asym -4.5909 60.8193 -0.075 0.941   
## RO 0.7340 0.1473 4.982 8.27e-05 \*\*\*  
## lrc -3.4809 11.6256 -0.299 0.768   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4502 on 19 degrees of freedom  
##   
## Number of iterations to convergence: 0   
## Achieved convergence tolerance: 2.433e-06

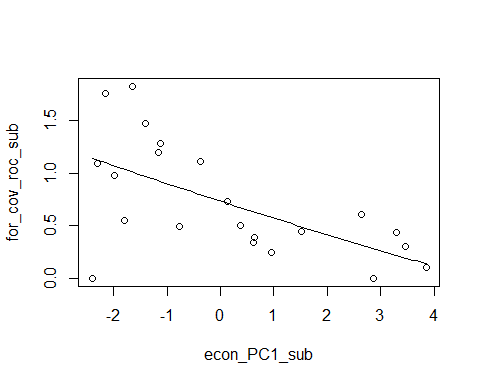


The shape wasn’t what I expected, and so I tried the slightly different equation from Crawley,

but parameterized the model using the parameter estimates from the above self-starter.

Asym <- -4.6  
RO <- 0.73  
b <- Asym-RO  
lrc <- (log((Asym - 1.25)/ b))/-2  
  
mod.assym2 <- nlsLM(for\_cov\_roc\_sub ~ Asym-b\*exp(-lrc\*econ\_PC1\_sub),   
 start = list(Asym=Asym, b=b, lrc=lrc),  
 control = nls.control(maxiter = 1000))

##   
## Formula: for\_cov\_roc\_sub ~ Asym - b \* exp(-lrc \* econ\_PC1\_sub)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)  
## Asym -4.59953 61.02397 -0.075 0.941  
## b -5.33355 61.13564 -0.087 0.931  
## lrc 0.03073 0.35782 0.086 0.932  
##   
## Residual standard error: 0.4502 on 19 degrees of freedom  
##   
## Number of iterations to convergence: 17   
## Achieved convergence tolerance: 1.49e-08



This didn’t seem to make any difference at all to the fit of the model on the plot, even though the value of lrc (which is the parameter representing the natural log of the rate parameter) changes from -3.48 in the self starting model to 0.031 in the above model. I would assume that if the rate parameter changed that much the slope of the model would change. But perhaps I’m misinterpreting that.

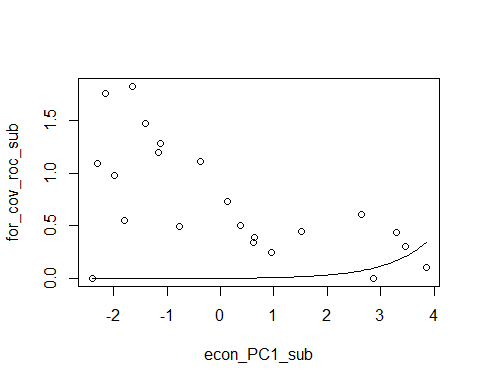
### 2 parameter asymptotic exponential

Next I tried a simpler asymptotic exponential, taken from Crawley. The models equation is

Again I estimated the starting value for asymptote to be ~0, and I varied the starting value of b from -2 to 8.

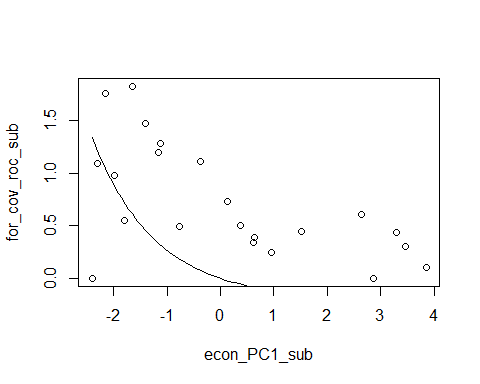
Asym <- 0  
b <- -2  
mod.assym3 <- nlsLM(for\_cov\_roc\_sub ~ Asym\*(1-exp(-b\*econ\_PC1\_sub)),   
 start = list(Asym=Asym, b=b))

##   
## Formula: for\_cov\_roc\_sub ~ Asym \* (1 - exp(-b \* econ\_PC1\_sub))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)  
## Asym -0.002771 0.065636 -0.042 0.967  
## b -1.250302 6.460834 -0.194 0.849  
##   
## Residual standard error: 0.931 on 20 degrees of freedom  
##   
## Number of iterations to convergence: 12   
## Achieved convergence tolerance: 1.49e-08



Asym <- 0  
b <- 2  
mod.assym3 <- nlsLM(for\_cov\_roc\_sub ~ Asym\*(1-exp(-b\*econ\_PC1\_sub)),   
 start = list(Asym=Asym, b=b))

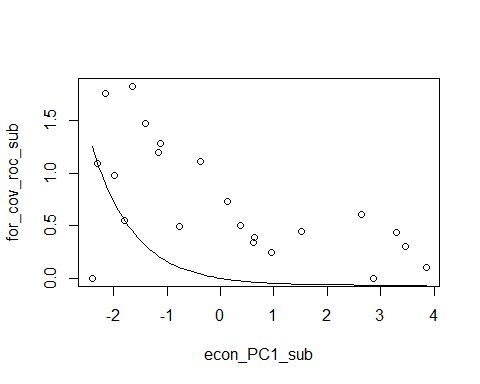
##   
## Formula: for\_cov\_roc\_sub ~ Asym \* (1 - exp(-b \* econ\_PC1\_sub))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)  
## Asym -0.2006 0.3056 -0.657 0.519  
## b 0.8470 0.6053 1.399 0.177  
##   
## Residual standard error: 0.7347 on 20 degrees of freedom  
##   
## Number of iterations to convergence: 13   
## Achieved convergence tolerance: 1.49e-08



Asym <- 0  
b <- 8  
mod.assym3 <- nlsLM(for\_cov\_roc\_sub ~ Asym\*(1-exp(-b\*econ\_PC1\_sub)),   
 start = list(Asym=Asym, b=b))

## Warning in nls.lm(par = start, fn = FCT, jac = jac, control = control, lower = lower, : lmdif: info = -1. Number of iterations has reached `maxiter' == 50.

##   
## Formula: for\_cov\_roc\_sub ~ Asym \* (1 - exp(-b \* econ\_PC1\_sub))  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)  
## Asym -0.06557 0.18012 -0.364 0.72  
## b 1.24744 1.17265 1.064 0.30  
##   
## Residual standard error: 0.7542 on 20 degrees of freedom  
##   
## Number of iterations till stop: 50   
## Achieved convergence tolerance: 1.49e-08  
## Reason stopped: Number of iterations has reached `maxiter' == 50.



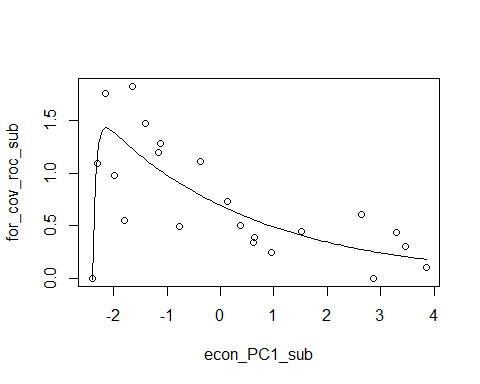
The above model is getting close to the shape of the data (if the first data point wasn’t there), but I tried increasing b even further and it did not improve the model fit.

### Biexponential

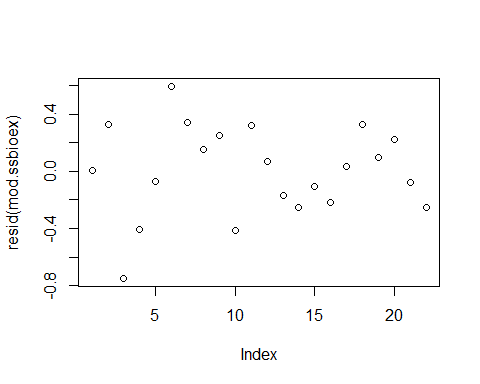
None of the above asymptotic exponentials fit the data well. The next model I tried was a biexponential model. I used the self-starting model which has the equation:

mod.ssbioex <- nls(for\_cov\_roc\_sub ~ SSbiexp(econ\_PC1\_sub,A1,lrc1,A2,lrc2))

##   
## Formula: for\_cov\_roc\_sub ~ SSbiexp(econ\_PC1\_sub, A1, lrc1, A2, lrc2)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## A1 -5.238e-16 1.337e-14 -0.039 0.969181   
## lrc1 2.696e+00 7.148e-01 3.772 0.001397 \*\*   
## A2 6.925e-01 9.470e-02 7.312 8.61e-07 \*\*\*  
## lrc2 -1.053e+00 2.546e-01 -4.137 0.000619 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3388 on 18 degrees of freedom  
##   
## Number of iterations to convergence: 0   
## Achieved convergence tolerance: 4.369e-06



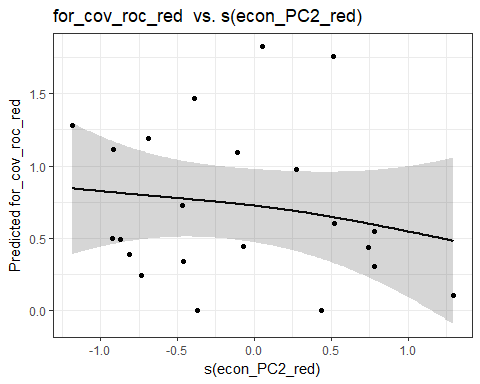
The residual plot of the model:



This model appears to be by far the best fit for the data, and the residuals look good. It captures the shape of the data at the lower values of econ\_PC1. However I can’t help but wonder about the first data point (x=-2.5, y=0), because if that point wasn’t there the data would take on a different shape all together. And that data point seems to be dictating the shape of the curve a lot. That data point alone is very important - if it is correct it suggests that the rate of forest cover loss increases very rapidly as the economy grows at the start, and then this rate of change reaches a threshold after which it starts to decline. If that data point is an outlier (and not correct), then the shape would tell a different story - that the rate of forest cover loss doesn’t increase at all within the bounds of the economic factors in this time period, but in fact starts high but decreases at a decreasing rate as the economy grows. The trouble is both of those shapes are plausible in reality.

## Forest cover rate of change ~ econ\_PC2

As can be seen in the below plot, there doesn’t really appear to be much of a relationship between forest cover RoC and econ\_PC2. I fitted a GAM to see what it would come up with:

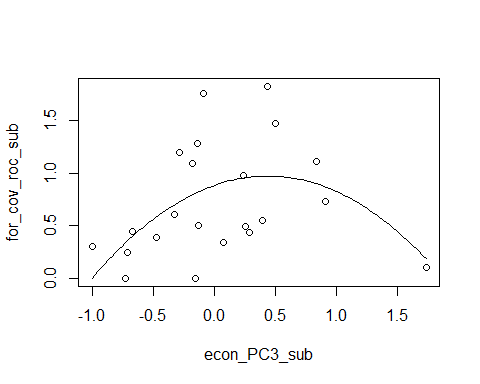


There clearly isn’t much of a relationship between the variables. I also fitted a linear model, as I couldn’t see much else to do, and the model fit was very poor (r2 = 0.03, p = 0.44).

## Forest cover rate of change ~ econ\_PC3

I fitted a GAM to the data to assess the shape, and based on the that I fitted a polynomial model with the equation (from Crawley):

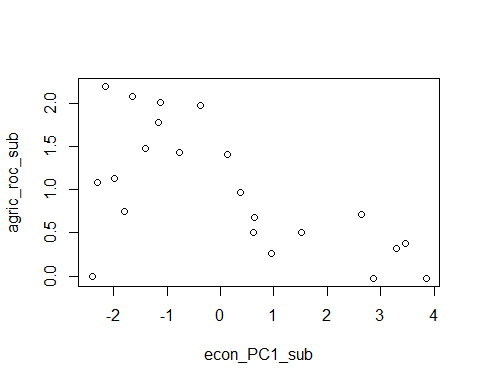
mod.poly <- nls(for\_cov\_roc\_sub ~ a+b\*econ\_PC3\_sub-c\*econ\_PC3\_sub^2,   
 start = list(a=2, b=5, c=0.2))



Again, we see that one data point (x=1.9, y=0.1) is having a huge effect on the shape of the curve. If that data point wasn’t there we would probably be looking at an exponential function.

# Agriculture cover rate of change ~ econ\_PC1

Moving on to agricultural cover rate of change, and we see that the relationship is similar to that of forest cover rate of change:

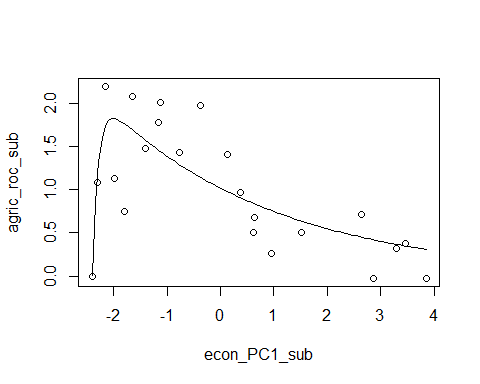


For that reason I started off with the biexponential model.

### Biexponenial model

For some reason the self starting model struggled to find starting values, but the model ran when I used the self starting model formula, but used custom starting values which I took from the for\_cov ~ econ\_PC1 biexponential model, as the data are quite similar.

mod.bioex2.agr <- nls(  
 agric\_roc\_sub ~ a\*exp(-exp(b)\*econ\_PC1\_sub)+c\*exp(-exp(d)\*econ\_PC1\_sub),  
 start = list(a=-5.238e-16, b=2.696, c=0.6925, d=-1.053),  
 control = nls.control(maxiter = 1000))



# Agriculture cover rate of change ~ econ\_PC2

As with forest cover RoC ~ econ\_PC2, the relationship between agriculture RoC and econ\_PC2 is fairly weak, however it is stronger than forest cover, as shown by the GAM below:

