Smoothing-splines Mixed-effects Models in R using the `sme` Package: a Tutorial

Maurice Berk
Imperial College London
maurice.berk01@imperial.ac.uk

April 3, 2013

Abstract

In this vignette, the user is guided through some basic analyses using the `sme` R package for fitting and visualising smoothing-splines mixed-effects (SME) models. SME models are an extension of the standard linear mixed-effects model that can account for a wide range of non-linear behaviours. They are robust to small sample sizes, noisy observations and missing data and hence a common application area is genomics time series data analysis, from which the example data sets distributed with the package and described in this tutorial originate.

1 Introducing the data

The `sme` package contains two data sets. The first of these is a small subset of data from an experiment conducted on blood samples from six healthy human volunteers to investigate the genetic response to *M.Tuberculosis* infection. This will be referred to as the 'MTB' data set and is a `data.frame` with variables:

```
> library(sme)
> data(MTB)
> names(MTB)

[1] "y"        "tme"      "ind"      "variable"
```

y contains the observed gene expression values, tme contains the corresponding time points (in hours) at which the measurements in y were taken, ind is a factor identifying which subject is associated with the measurements in y, and variable is a factor indicating which gene transcript the measurements are associated with.

The typical approach with replicated genomics data sets such as these, which contain repeated measurements on more than one biological unit, is to model each transcript independently using a functional mixed-effects model [Storey et al., 2005, Liu and Yang, 2009, Berk et al., 2010]. To begin, considering only the first transcript then, with identifier 6031, the raw data can be visualised as a trellis plot using the lattice package:

```
> library(lattice)
> print(xyplot(y ~ tme | ind,data=MTB[MTB$variable==6031,],
+   xlab="Hour",ylab="Gene Expression"))
```

with the ouput given in Figure 1. The following salient features of the data can now be noted: (1) there are very few subjects and time points, (2) the response is highly non-linear, with a distinctive spike in gene expression levels at 24 hours, and (3) some data is missing, specifically the final observation for subject 6.

2 Introducing the model

Users already familiar with the theoretical details of functional mixed-effects models in general and SME models in particular may wish to skip straight to Section 3 where the illustration of the use of the sme package is resumed.

Given the few time points and aperiodicity, traditional time series analysis models are unlikely to yield good results on the type of data described above. Functional mixed-effects models have proven to be a popular alternative, capable of dealing with all of the associated issues. In a functional mixed-effects model, the observations on subject i are assumed to have come from an underlying smooth function of time, $y_i(t)$, which is decomposed into the following components:

$$y_i(t) = \mu(t) + v_i(t) + \epsilon_i(t) \tag{1}$$

where $\mu(t)$ is the mean function across all subjects, $v_i(t)$ is subject i's deviation from that mean function, also assumed to be a smooth function of time

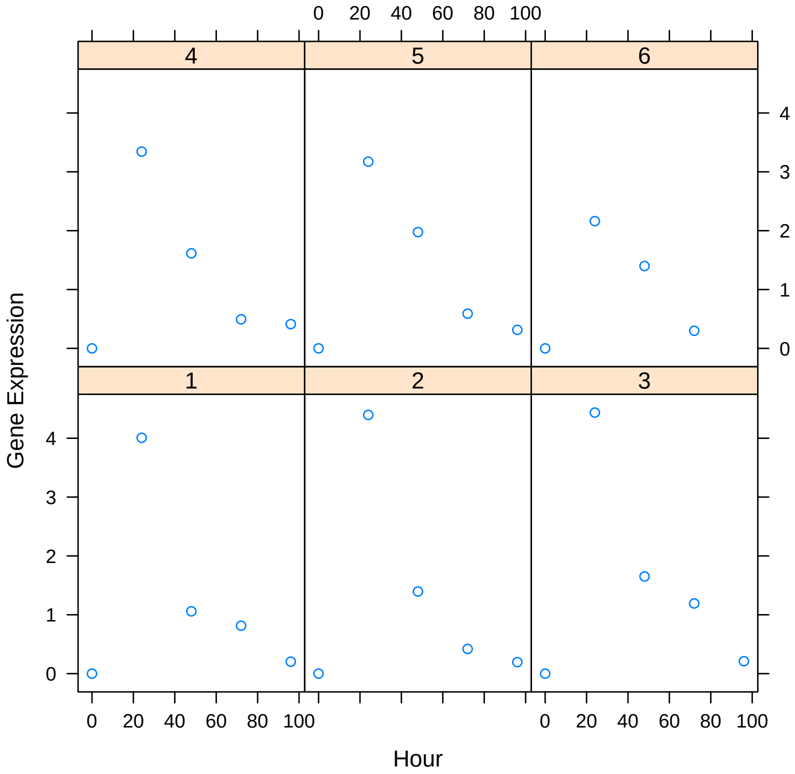

Figure 1: Raw data for gene transcript `6031` from the MTB data set. Note the small number of subjects and time points, the highly non-linear responses and the missing data (the final observation for subject 6)

and $\epsilon_i(t)$ is an error process. Analogous to the linear mixed-effects model for vectorial data [Harville, 1977], $\mu(t)$ is treated as a fixed, but unknown, population parameter while the $v_i(t)$ functions are assumed to be randomly sampled from the population as a whole.

In practice, to estimate the function $\mu(t)$ and the distribution of the functions $v_i(t)$, they must be parameterised in some way. Typically this is done using *splines* — piecewise polynomials — although wavelets or Fourier bases are amongst the other options. Splines themselves come in different flavours, and the monograph of Wu and Zhang [2006] is an excellent introduction to the various representations in a functional mixed-effects model context.

Essentially the different spline representations vary in the way in which they control the *smoothness*, or analogously the non-linearity, of the functions. Achieving the right level of smoothing is a critical aspect of the modelling process — too much smoothing and the underlying temporal dynamics will be lost; too little smoothing and spurious conclusions are likely to be made.

B-splines [de Boor, 1978] are a traditional choice of spline representation which control the smoothness of the spline by varying the number and location of *knots* that define the break points between the piecewise polynomials. This means they suffer from the drawback of only providing coarse control over the smoothness as there can only be an integer number of knots, with the upper limit dependent on the number of time points. This problem is exacerbated in the replicated genomics time series context (and other biological domains such as metabolomics and proteomics) where the very small number of time points severely restricts the range of non-linear behaviours that can be considred.

Smoothing-splines deal with this issue by using every distinct time point as a knot and avoiding the overfitting this would normally incur by introducing a penalty parameter for the lack of smoothness or *roughness* of the function. This penalty parameter can take any non-negative real value and hence fine control over the smoothness is achieved.

Parameterising the functions in (1) as smoothing splines allows it to be rewritten in matrix-vector format as

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\mu} + \boldsymbol{X}_i\boldsymbol{v}_i + \boldsymbol{\epsilon}_i \tag{2}$$

where $\boldsymbol{y}_i$ is a vector of all observations on subject i, $\boldsymbol{X}_i$ is an incidence matrix mapping the distinct design time points onto the time points at which subject i was actually observed, $\boldsymbol{\mu}$ is a vector of fitted values for the mean function at the design time points, $\boldsymbol{v}_i$ is a vector of fitted values for the subject-specific function at the design time points and $\boldsymbol{\epsilon}_i$ is the vector of error terms.

Standard practice is to assume that the $\boldsymbol{v}_i$ and $\boldsymbol{\epsilon}_i$ terms are multivariate normally distributed with zero mean vectors and covariance matrices $\boldsymbol{D}$ and $\sigma^2\boldsymbol{I}$ respectively. Under these assumptions, $\boldsymbol{y}_i$ is multivariate normally distributed, and the model parameters $\boldsymbol{\mu}$, $\boldsymbol{D}$ and σ^2 which maximise the likelihood can be found by treating the $\boldsymbol{v}_i$ as missing values and employing the Expectation-Maximisation (EM) algorithm. The penalty parameters for the roughness of the functions $\mu(t)$ and $v_i(t)$, λ_μ and λ_v, are incorporated by instead finding the values of $\boldsymbol{\mu}$, $\boldsymbol{D}$ and σ^2 which maximise the *penalised* likelihood.

3 Fitting the model

SME models control the degree of non-linearity through two smoothing parameters, one for the mean function and one for the subject-specific functions, denoted λ_μ and λ_v respectively. The smoothing parameters are non-negative real values; when small there is little smoothing and the functions can interpolate the data points. When they tend to infinity then the amount of smoothing is maximised and the functions tend to straight lines.

To illustrate this, first the transcript will be fit with $\lambda_\mu = \lambda_v = 0$. The model parameters are estimated using the EM algorithm by executing the following code:

```
> fit <- sme(MTB[MTB$variable==6031,c("y","tme","ind")],
+   lambda.mu=0,lambda.v=0)
> plot(fit,type="model",xlab="Hour",ylab="Gene Expression")
```

with the resulting model fit visualised in Figure 2. The single time point observations for all subjects are shown as circles. The thick red line is the fitted mean function, and the dashed black lines are the predicted subject specific functions. As expected, the subject specific functions interpolate the data points, which seems implausible as there is likely to be at least some degree of measurement error. Furthermore, checking the success flag for the EM algorithm:

```
> fit$info
```

```
[1] -1
```

indicates that the algorithm failed as the likelihood did not increase during one of the iterations (zero indicates success). This is likely due to numerical instability introduced by attempting to estimate more parameters than there are data points. With the smoothing parameters set to zero, there are effectively 31 model parameters: 5 parameters for the smoothing-spline representing the mean function (one for each time point); 25 parameters for the between-subject covariance matrix (5 time points × 5 time points) and 1 for the error variance. Taking into account the missing observation for subject 6 there are only 29 data points.

At the other extreme, the transcript can be refit with $\lambda_\mu = \lambda_v = 10^7$:

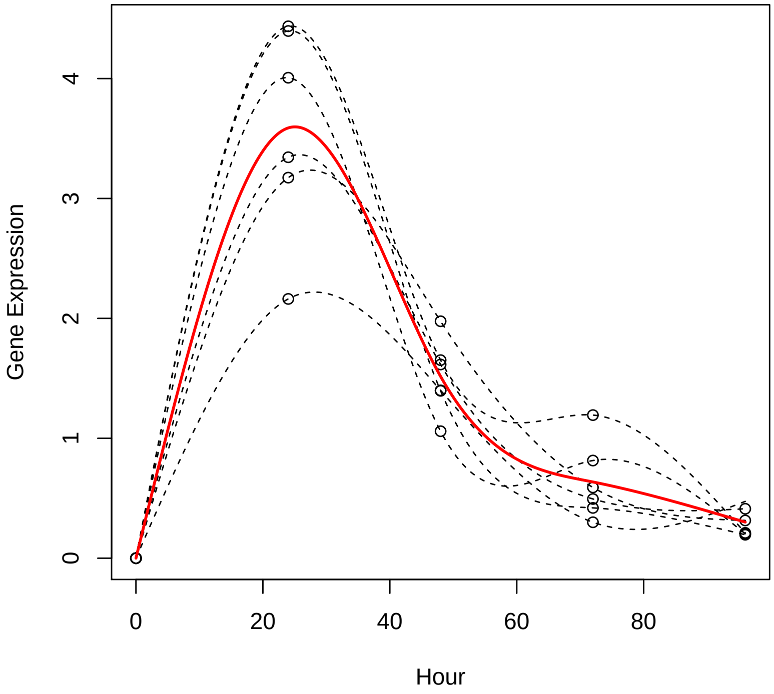

Figure 2: Fitting gene transcript 6031 from the MTB data set with $\lambda_\mu = 0$ and $\lambda_v = 0$. Note how this combination of smoothing parameters leads to the functions interpolating the data points. The idea of zero measurement error seems implausible

```
> fit <- sme(MTB[MTB$variable==6031,c("y","tme","ind")],
+   lambda.mu=1e7,lambda.v=1e7)
> plot(fit,type="model",xlab="Hour",ylab="Gene Expression")
```

Referring to Figure 3, this time the mean function is a straight line. Furthermore the subject specific functions coincide with the mean, indicating that with this level of smoothing all of the variance in the observations is attributed to measurement error and none to subject heterogeneity. Double checking the success flag:

```
> fit$info
```

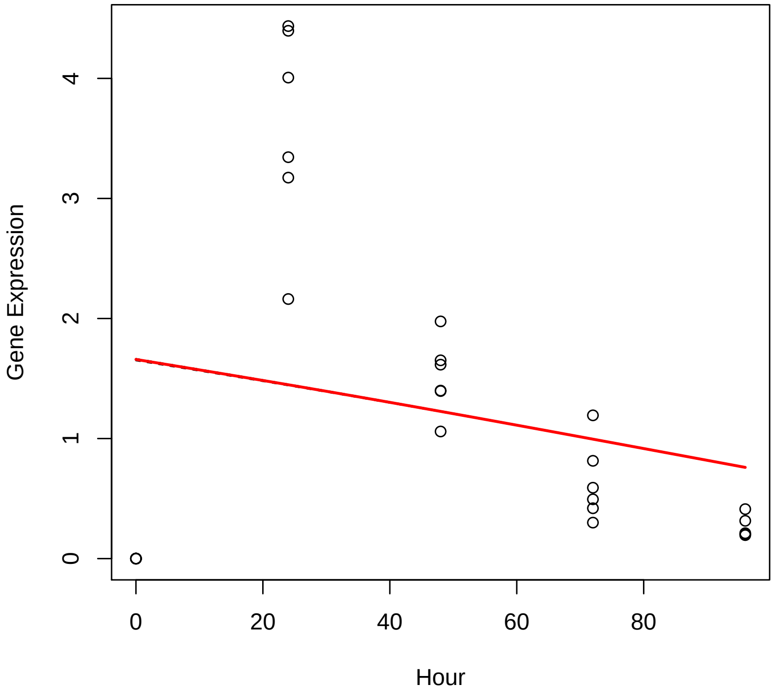

Figure 3: Fitting gene transcript 6031 from the MTB data set with $\lambda_\mu = 10^7$ and $\lambda_v = 10^7$. Note how this combination of smoothing parameters leads to a model which has dramatically oversmoothed the data, completely ignoring the underlying temporal behaviour indicated by the observations

```
[1] 0
```

shows that this time the EM algorithm successfully converged.

Neither of these two extremes has produced a satisfactory model fit. Alternatively, the *optimal* smoothing parameters according to some model selection criteria can be found using Nelder-Mead simplex search [Nelder and Mead, 1965]. This is the default behaviour for the `sme` function if `lambda.mu` or `lambda.v` is not set but, as shown above, users can not only get a feel for how changing the smoothing parameters impacts the model fit but also write their own search routines by calling the function with the parameters set explicitly.

One of the most popular model selection criteria is Akaike's Information Criterion (AIC) [Akaike, 1974] which scores the model as the log-likelihood with a penalty for the number of fitted parameters. Finding the optimal model under this criterion is achieved by executing:

```
> fit <- sme(MTB[MTB$variable==6031,c("y","tme","ind")],
+   criteria="AIC")
> plot(fit,type="model",xlab="Hour",ylab="Gene Expression")
```

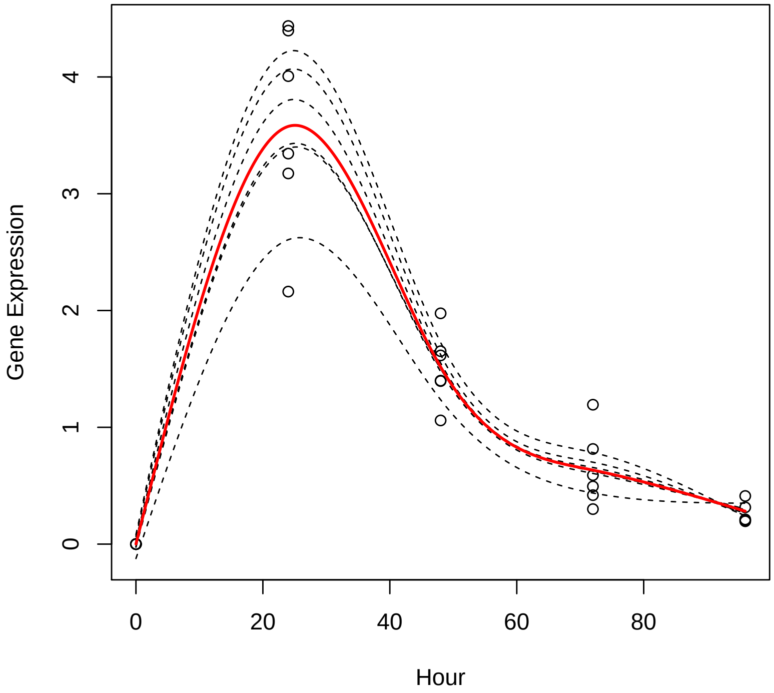

Figure 4: The optimal SME model for gene transcript `6031` from the MTB data set according to the AIC. Compared to Figures 2 and 3, this model yields fitted functions that strike a good balance between smoothness and adequately modelling the underlying temporal behaviour suggested by the observations

By default, however, the sme function uses the *corrected* AIC of Hurvich and Tsai [1989] which includes a correction for small sample sizes. In addition to these two forms of AIC, the user also has the option of the Bayesian Information Criterion [Schwarz, 1978] where the penalty can either depend on the total number of observations across all subjects (criteria="BIC") or the total number of subjects (criteria="BICn"). The 'correct' criteria to use will depend on how smooth the underlying functions are likely to be given any prior knowledge that may be available, and on how large the sample size is. The corrected AIC gives good results in most instances, hence it is the default option.

4 Additional visualisation options

While Figure 4 provides a useful overview of the model fit, it is impossible to determine which observations are associated with which subject. As a result, it may be useful to produce a trellis plot as with the raw data, except with the fitted subject-specific functions superimposed on top. This can be done by running:

```
> plot(fit,type="raw",showModelFits=TRUE,xlab="Hour",
+   ylab="Gene Expression")
```

with the result given in Figure 5.

It can be insightful to visualise the 95% confidence band for the estimated mean function in order to assess the model fit. This is done by passing showConfidenceBands=TRUE to the plot function:

```
> plot(fit,type="model",xlab="Hour",ylab="Gene Expression",
+   showConfidenceBands=TRUE)
```

with the result given in Figure 6.

The model fit can further be assessed using a diagnostic plot generated via:

```
> plot(fit,type="diagnostic")
```

which is shown in Figure 7. There are four panels to this plot, which has been heavily inspired by Wu and Zhang [2006]. Each panel visualises the standardised residuals in a different way in order to help determine whether

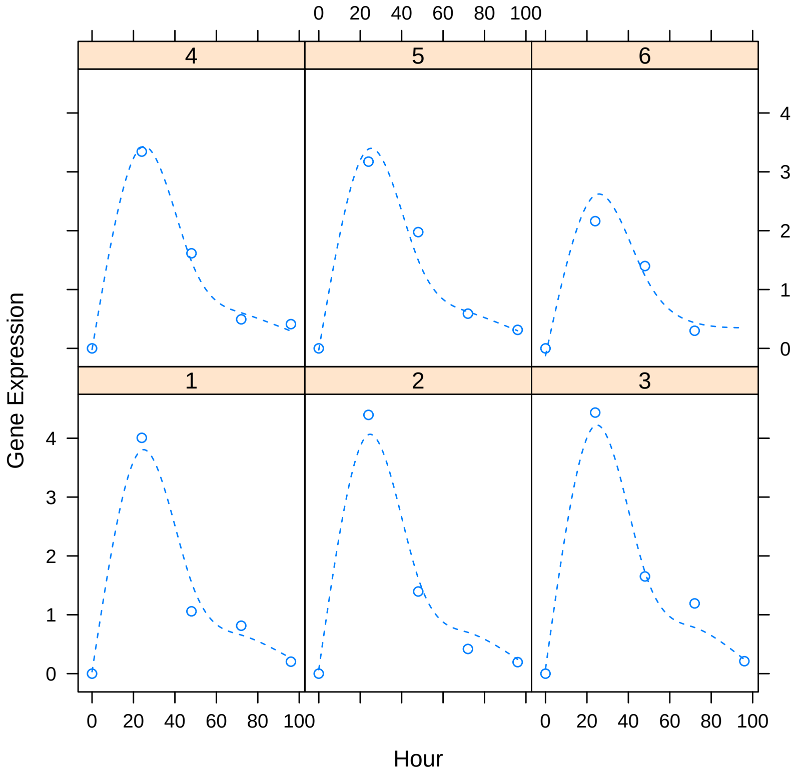

Figure 5: Raw data for gene transcript `6031` from the MTB data set with fitted subject-specific curves overlaid

the data has been adequately modelled. The top panels and the bottom left are all intended to detect whether there is any latent structure to the residuals. The Q-Q plot in the bottom right helps to determine whether the assumption for normality for the residuals is valid. In this particular instance, there does not appear to be any latent structure to the residuals. The Q-Q plot raises questions over the assumption of normality due to the extreme values but this is largely to be expected given the small number of observations.

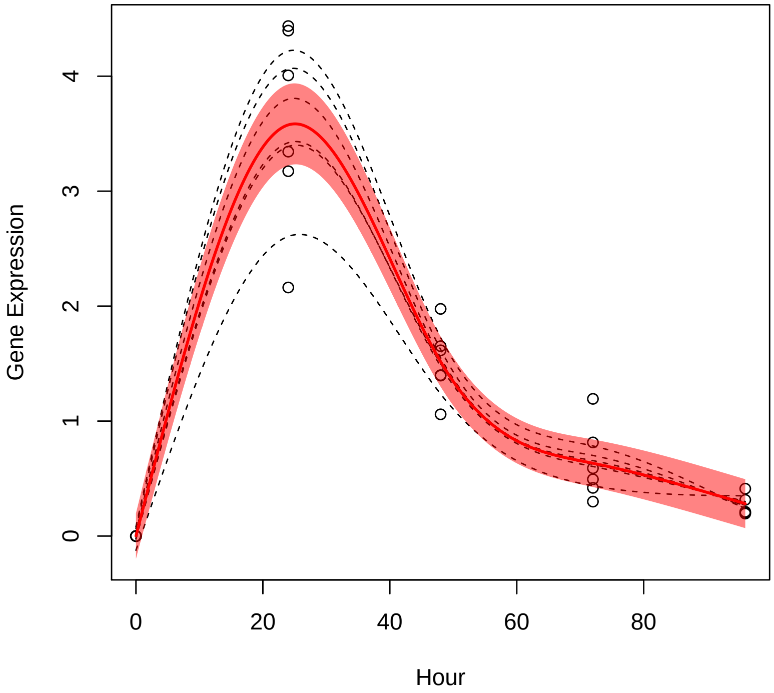

Figure 6: Visualisation of the SME model fit for gene transcript `6031` from the MTB data set, with the addition of the 95% confidence band for the estimate of the mean function

5 Advanced topics

5.1 Carrying out multiple model fits in parallel

Thus far fitting only a single gene transcript has been considered. In a typical genomics data set there will be tens of thousands of transcripts to be fit. In order to make this task as efficient as possible, the `sme` package supports OpenMP (`http://www.openmp.org/`) for using multiple threads to carry out multiple model fits in parallel (note that this feature is platform dependent and in particular is unavailable on OS X). The MTB data set contains ten gene transcripts in total in order to illustrate this process.

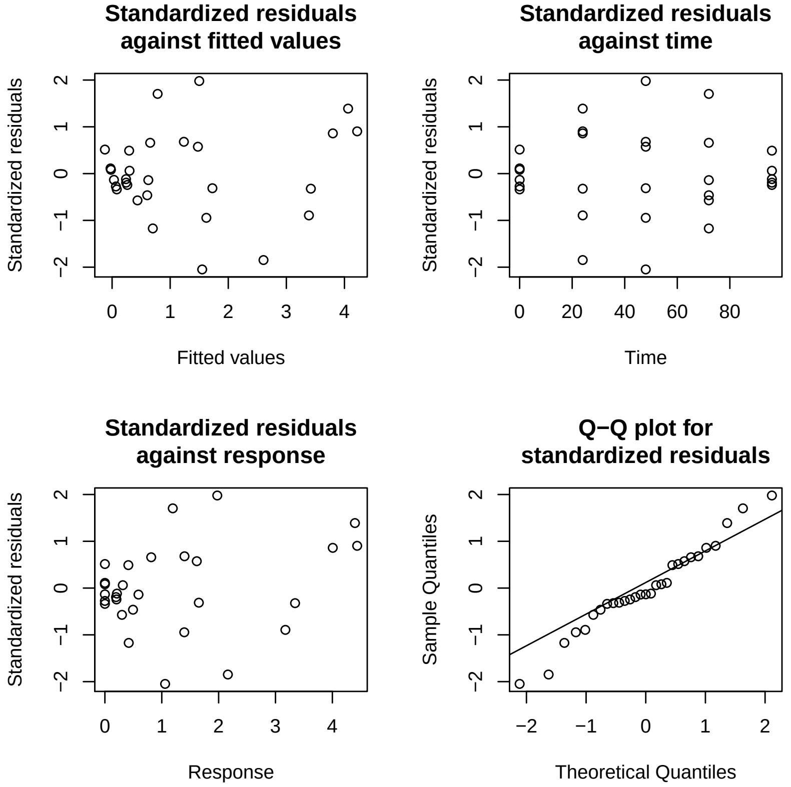

Figure 7: Diagnostic plot for assessing the SME model fit to gene transcript `6031` from the MTB data set

First, all ten transcripts can be fit one after the other as follows:

```
> system.time(fits <- lapply(unique(MTB$variable),
+   function(v) sme(MTB[MTB$variable==v,c("y","tme","ind")])))
```

```
   user  system elapsed
   1.75    0.00    1.76
```

Alternatively, the `sme` function can be called on the entire MTB data set. As this `data.frame` contains a `variable` factor, the `sme` function will automatically recognise that multiple fits should be carried out in parallel. By default, the OpenMP system will automatically select the number of threads to be used. There will be some inefficiency due to the overheads of thread creation (these overheads are greater on Windows than other operating systems, and are especially noticable with such a small example). Alternatively the number of threads can be specified explicitly through the `numberOfThreads` parameter:

```
> system.time(fits <- sme(MTB,numberOfThreads=3))

   user  system elapsed 
   1.85    0.00    1.00 
```

5.2 Using less knots

Although using every distinct time point as a knot works well for the MTB data set, and works well for the vast majority of other genomics time series experiments, in some instances it may be inappropriate. The `sme` package contains a second data set, which consists of a single gene transcript from an experiment investigating an inflammatory condition in children. This will be referred to as the 'inflammatory' data set. In this experiment the sampling times are highly irregular, as can be seen by visualising the data:

```
> data(inflammatory)
> plot(y~tme,data=inflammatory,xlab="Day",
+   ylab="Gene Expression")
> for(ind in inflammatory$ind) lines(y~tme,
+   data=inflammatory[inflammatory$ind==ind,],
+   lty="dashed")
```

shown in Figure 8. The single time point observations are shown as circles which have been joined by lines where they belong to the same subject. Using each distinct time point as a knot will still work but will be slow. Some speed can be gained by relaxing the convergence criteria for the Nelder-Mead search by setting the parameter `deltaNM=0.1` (the default is 0.001):

```
> system.time(fit <- sme(inflammatory,
+   deltaNM=0.1))

   user  system elapsed 
  52.69    0.70   53.70 
```

```
> plot(fit,xlab="Day",ylab="Gene Expression")
```

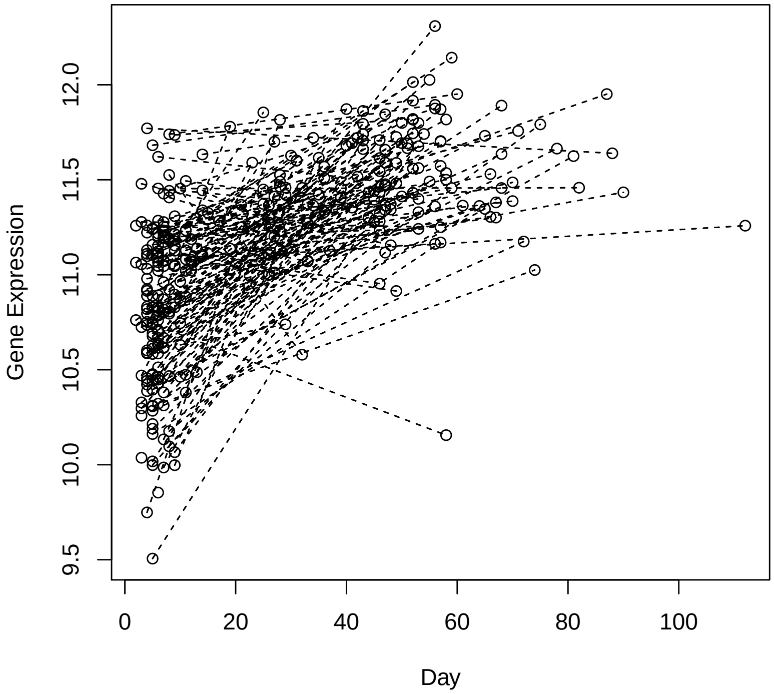

Figure 8: Raw data for the example gene transcript in the inflammatory data set. Note the irregular time points at which the observations were collected

with the model fit visualised in Figure 9. Alternatively, a vector of time points to use as knots can be specified as an argument to the `sme` function. Note that these should be *internal* knots, as the time points at the extremes of the time course will automatically be used. For example, using five equally spaced knots can be achieved by running:

```
> my.knots <- seq(min(inflammatory$tme),max(inflammatory$tme),
+   length.out=7)[-c(1,7)]
> my.knots
```

```
[1] 20.33333 38.66667 57.00000 75.33333 93.66667
```

```
> system.time(fit <- sme(inflammatory,knots=my.knots,
+   deltaNM=0.1))
```

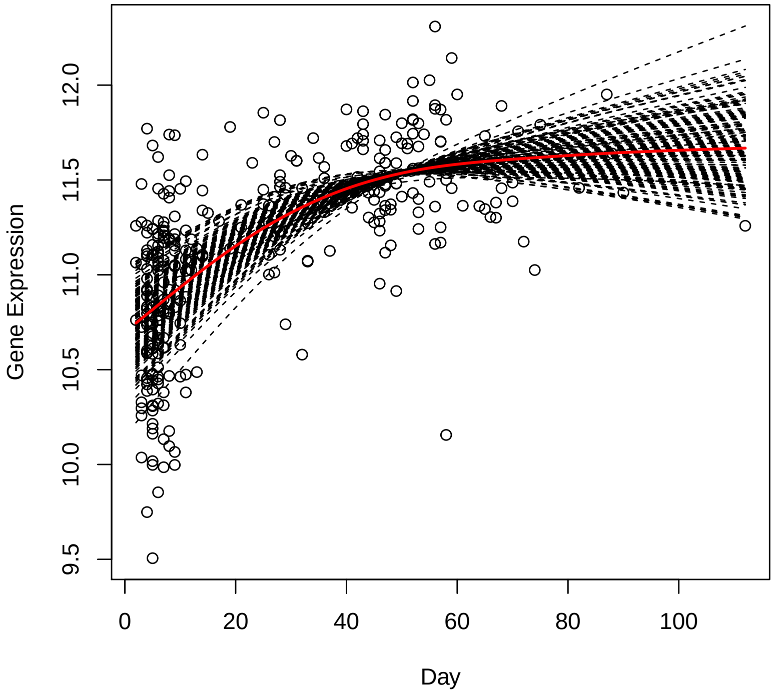

Figure 9: SME model fit to the example gene transcript in the inflammatory data set when using every distinct time point as a knot

```
   user  system elapsed 
   5.76    0.00    5.79 
```

```
> plot(fit,xlab="Day",ylab="Gene Expression")
```

which executes significantly faster. Comparing the visualisation of the model fit, given in Figure 10, with Figure 9, ultimately there is very little difference between the two. Note that in these situations it is not overly critical to pick a 'good' number of knots; provided 'enough' knots are used so that a sufficient range of non-linear behaviours can be considered. The smoothing parameters will take care of avoiding overfitting as usual.

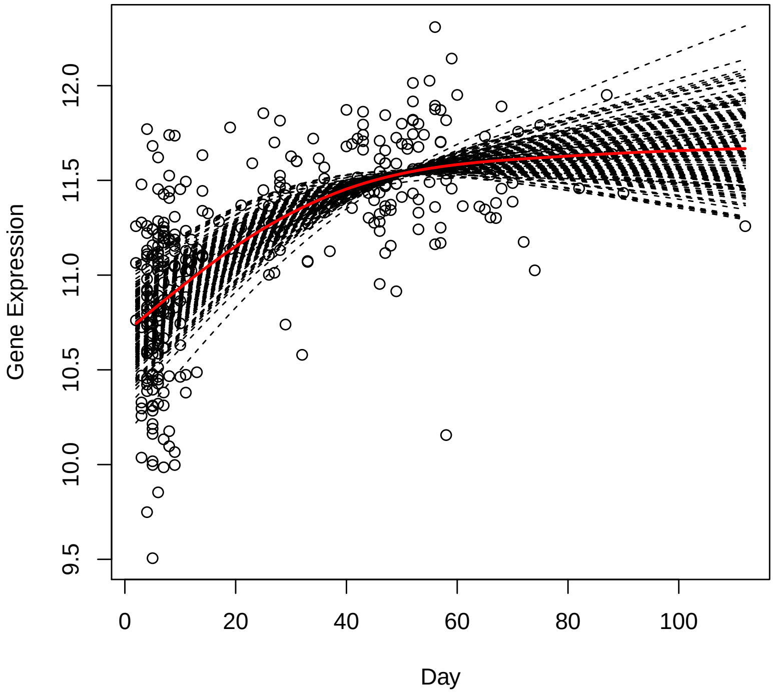

Figure 10: SME model fit to the example gene transcript in the inflammatory data set when using only five equally spaced internal knots

References

H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, Dec 1974. ISSN 0018-9286.

M. Berk, M. Levin, C. Hemingway, and G. Montana. Longitudinal analysis of gene expression profiles using functional mixed-effects models. In *Studies in Theoretical and Applied Statistics*, 2010.

Carl de Boor. *A Practical Guide to Splines*. Springer, New York, 1978.

David A. Harville. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, 72(358):320–388, 1977.

C. M. Hurvich and C.-L. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

Xueli Liu and Mark C. K. Yang. Identifying temporally differentially expressed genes through functional principal components analysis. *Biostatistics*, page kxp022, 2009.

J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6: 461–464, 1978.

John D Storey, Wenzhong Xiao, Jeffrey T Leek, Ronald G Tompkins, and Ronald W Davis. Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 102(36):12837–12842, Sep 2005.

Hulin Wu and Jin-Ting Zhang. *Nonparametric Regression Methods for Longitudinal Data Analysis*. Wiley, 2006.