Some GAM exercises

1. The help system. Start R. Type `help.start()`. Navigate to the `mgcv` package. Browse a little, starting with the page `mgcv-package`.

2. Basic modelling of UK load. In R type

   ```
   library(qgam);library(mgcv)
   data(UKload)
   head(UKload)
   ?UKload
   ```

   (a) First try a simple 1-D smooth of cumulative time.

   ```
   UKload$day <- 1:nrow(UKload)
   b <- gam(NetDemand~s(day,k=100),data=UKload,method="REML")
   plot(b,n=1000,rug=FALSE,scheme=1,residuals=TRUE)
   ```

   `?plot.gam` explains the plotting options.

   (b) Run `gam.check(b)` and plot the model residuals against the day of week variable `Dow`. Modify the model appropriately, and also double the basis dimension for time, since 100 maybe looks a little restrictive. Note the `all.terms=TRUE` option to `plot.gam`.

   (c) Run `gam.check` again — note the heavy tailed residuals. Look at the help file for the `scat` family.

   (d) Re-run the model using the `scat family`. To speed things up use `bam` in place of `gam`, and use the options `discrete=TRUE` and `nthreads=2`.

   (e) Try out `summary()` and `anova()` on your fitted model.

   (f) Add in a smooth effect of `wM` (a basis dimension of 20 should be adequate here).

   (g) An alternative model would use a repeating cyclic smooth of time of year, `Posan`, with a long term smooth of `day`. Try this out using the `"cc"` basis for the cyclic term and dropping the basis dimension for `day` to 50.

3. Extra follow up question, examining prediction. `predict.gam` takes your fitted GAM and a data frame containing predictor variable values at which predictions are required, and produces the required predictions.

   (a) Produce a data frame suitable for one month ahead prediction from the models in question 1. Just set `wM` to the last value observed.

   (b) Make 30 day predictions for the models from 2f and 2g, plotting these predictions on the same plot as the model fitted load over the whole dataset. Which prediction is more reasonable and why?

   (c) The `b.spline` helpfile has some further discussion and examples relating to extrapolation with smoothers, and the strong influence of the penalty on extrapolation behaviour. Try running the extrapolation example, in particular.

4. Auto-regressive load prediction modelling. Today's load is like yesterday's load, to the extent that it is worth using yesterday's load in a predictive model. This often works so well that the question arises of whether loads further back in time might also be useful predictors (for example heating and cooling demand are susceptible to thermal inertia effects). We might use a model like

$$f_1(\texttt{load}_{i-1}) + f_2(\texttt{load}_{i-2}) + \dots$$

but this is probably too flexible, as it allows completely different forms for the f_j. An alternative is to use a single tensor product smooth of lagged load and lag, so that the dependence of load on lagged load varies smoothly with the lag. The model term then becomes

$$f(\texttt{load}_{i-1}, 1) + f(\texttt{load}_{i-2}, 2) + \dots = \sum_j f(\texttt{load}_{j-1}, j).$$

`mgcv` has a mechanism to allow such summation terms. Suppose that we set up a matrix of lagged loads, `lalo`, each column of which represents a different lag. We also set up a corresponding matrix containing the corresponding lags themselves, `lag`. Then `te(lalo,lag)` implements exactly the required term. The `te` smoother is evaluated to give a matrix of evaluated values, and the matrix then has its rows summed (`by` variables can be used to weight the summation if needed).

(a) Create appropriate lag matrices with something like

```
lo <- UKload$NetDemand
n <- length(lo)
lag <- lalo <- matrix(0,n,7)
for (i in 1:ncol(lag)) {
  lag[,i] <- i
  lalo[,i] <- c(rep(lo[1],i),lo[1:(n-i)])
}
```

The substitution of not available lagged loads at the start is somewhat arbitrary, but will have little effect here.

(b) Now try adding a distributed lag term to the model from Q2. Note that there is one awkward detail. Imagine that you included an `s(lag)` term in the model. If you think about it, this is just like adding another constant term to the model: as he smooth evaluates to the same thing for every term. So such a term is not identifiable. Now the tensor product basis contains a basis for terms like `s(lag)`, so also has an identifiability problem. To fix it we need the basis for terms like `s(lag)` to be removed. If we remove the constant function from the span of the marginal basis for `lalo`, then this will happen: the `ti` tensor product constructor allows such constraints. The `ti` term we need is

```
ti(lola,lag,mc=c(1,0),k=c(20,5))
```

where `mc` specifies the constraints and `k` specifies the marginal basis dimensions.

(c) Examine the model fits, and decide whether the distributed lag term really seems worthwhile.

5. Extra question. The dependence of load on previous load is likely to depend on whether we are predicting a working day after a working day, a holiday after a working day, a holiday after a holiday or a working day after a holiday ('holiday' includes 'weekend'). Incorporate this fact into a model dependent on the previous day's load (but not higher lags).