**qgam: computer lab exercises**

Please make sure that the following packages are installed:

1. `qgam` (`install.packages("qgam")`);
2. `devtools` (`install.packages("devtools")`);
3. `mgcViz` (`library(devtools); install_github("mfasiolo/mgcViz")`);
4. `gamair` (`install.packages("gamair")`).

We have two main exercises:

1. $CO_2$ modelling;
2. Electricity load forecasting;

but feel free to try `qgam` on your own data.

# 1 $CO_2$ modelling

This question[1] is about modelling data with seasonality, and the need to be very careful if trying to extrapolate with QGAMs (or any statistical model). The data frame `co2s` contains monthly measurements of $CO_2$ at the south pole from January 1957 onwards. The columns are `co2`, the month of the year, `month`, and the cumulative number of months since January 1957, `c.month`. There are missing `co2` observations in some months.
Questions:

1. Load `qgam`, `mgcViz` and the data with `library(gamair); data(co2s)`
2. Plot the $CO_2$ observations against cumulative months.
3. Fit an additive model for the median, $\mu_{0.5}(\texttt{c.month}_i) = f(\texttt{c.month}_i)$ where $f$ is a smooth function, using the `qgam` function. Use the `cr` basis, and a basis dimension of 100. Set `err` $= 0.1$ in `qgam` to avoid warnings and speed up computation.
4. Obtain the predicted $CO_2$ for each month of the data, plus 36 months after the end of the data, as well as associated standard errors. Produce a plot of the predictions with twice standard error bands. Are the predictions in the last 36 months credible? NB: to produce the plot you have to write your own code, `mgcViz` does not produce such plots.
5. Fit the model $\mu_{0.5}(\texttt{c.month}_i) = f_1(\texttt{c.month}_i) + f_2(\texttt{month}_i)$ where $f_1$ and $f_2$ are smooth functions. Use a basis of dimension 50 for $f_1$ and a cyclic basis for $f_2$. In the `qgam` call, you will need to set argument `argGam` to `list(knots=list(month=c(1,13)))` to make so that that the effect of January is the same as January, not that December and January are the same!
6. Repeat the prediction and plotting in question 4 for the new model. Are the predictions more credible now? Explain the differences between the new results and those from question 4.
7. Transform the fitted qgam model to a `gamViz` object, by calling `getViz` with argument `nsim = 0` (we need to set `nsim` to 0 because, at the present moment, qgam does not allow to simulate data from the model). Plot the smooth effects and experiment with model checks (e.g. QQ-plots). Can you expect the residuals to be normally distributed in a quantile regression model? We talk more about this in the exercise on electricity load forecasting.

[1] Shamelessly adapted from Simon Wood's notes

## 2 Electricity load forecasting

Here we consider a UK electricity demand dataset, taken from the national grid. The dataset covers the period January 2011 to June 2016 and it contains the following variables:

- `NetDemand` net electricity demand between 11:30am and 12am.
- `wM` instantaneous temperature, averaged over several English cities.
- `wM_s95` exponential smooth of `wM`, that is `wM_s95[i] = a*wM[i] + (1-a)*wM_s95[i]` with `a=0.95`.
- `Posan` periodic index in `[0, 1]` indicating the position along the year.
- `Dow` factor variable indicating the day of the week.
- `Trend` progressive counter, useful for defining the long term trend.
- `NetDemand.48` lagged version of `NetDemand`, that is `NetDemand.48[i] = NetDemand[i-2]`.
- `Holy` binary variable indicating holidays.
- `Year` and `Date` should obvious, and partially redundant.

Questions:

1. Load `qgam`, `mgcViz` and the data (`data("UKload")`). Then create a model formula (e.g. `y~s(x)`) containing: smooth effects for `wM`, `wM_s95`, `Posan` and `Trend` with 20, 20, 4 and 50 knots and cubic regression splines bases (`bs='cr'`); parametric effects for `Dow`, `NetDemand.48` and `Holy`.
2. Use the `tuneLearn` function to evaluate the calibration loss for the median, on a grid of values for $\log(\sigma)$ (e.g. `lsig=seq(4,8,le=16)`). In the `tuneLearn` call, set `err=0.1` to avoid numerical problems, speed up computation by using only 20 bootstrap samples (`control=list("K"=20)`) and by setting `multicore=T`.
3. Let `closs` be the output of `tuneLearn`. Use the function `check` on `closs`. It is important that the loss looks smooth, especially around its minimum. How do the effective degrees of freedom vary with $\log(\sigma)$?
4. Call `qgam` to fit this model for the median (use again `err=0.1`), and set argument `lsig` to `closs$lsig`, the latter being the value of $\log(\sigma)$ achieving the lowest loss on the grid. Convert to the output of `qgam` using `getViz`, with `nsim=0` and plot the fitted smooth effects. Do you notice anything problematic about the effect of `Posan`? How many degrees of freedom are we using for this smooth effect (use `summary`)?
5. Modify the effect of `Posan` to use an adaptive (`bs='ad'`) spline basis. Then refit the model with `qgam` (use again `lsig=closs$lsig` and `err=0.1`) and plot the smooth effects. Has the effect of `Posan` changed? How many degrees of freedom are we using now for `Posan`? Explain what happened.
6. Use `mqgam` to fit this model to 10 quantiles between 0.05 and 0.95, using `err=0.15` and `lsig=closs$lsig`. Use `qdo` and `pen.edf` to extract the effective degrees of freedom corresponding to each quantile and plot them against the quantiles. Why would we expect the degrees of freedom to decrease, as we move in the far tails of the response?

7. Let `fitM` be the of the `mqgam` call in the previous question. Use the code in Appendix to plot the smooth effects corresponding to each quantile on a single plot. The effect corresponding to high quantiles (e.g. 0.95) are green, the orange curves correspond to low quantiles (e.g. 0.05). Think about the interpretation of these curves. NB: here we are plotting the smooth effects, not the predicted quantiles, hence the effects corresponding to, say, quantile 0.9 can fall below that of quantile 0.1.

8. Use `fit3 <- qdo(fitM, 0.45, getViz, nsim=0)` to extract the `qgam` model corresponding to quantile $\tau = 0.45$. Look at the distribution of the residuals along `wM`, `wM_s95` and `Posan` using `check1D` with the `l_densCheck` layer. You should see large departures from normality, but this is not a problem, because in quantile regression we don't have a probabilistic model of the response, hence we cannot expect the residuals to be normal. But we can expect that, if we are fitting quantile 0.45, around 45% of the residuals should be negative. Use `check1D` with the `l_gridCheck1D(gridFun = function(.x){sum(.x < 0)/length(.x)})` layer to check that the fraction of negative residuals does not depart too much from 0.45. HARD PART: it would be nice to add confidence bands to the plots produced by `check1D`. You can do this by adding further `l_gridCheck1D` layers to these plots, but using a `gridFun` which uses the number of samples $n$ in each bin to calculate the confidence bands (which are based simply on binomial distribution with parameters $p = 0.5$ and $n$). If you have no idea about how to do this last point, copy-paste the code in Appendix B.

# Appendices

## A Code for plotting smooth effects of all quantiles

Let `fitM` be the output of `mqgam` from question 6. This code should plot the smooth effects of all quantile together:

```
qus <- as.numeric( names(fitM$fit) )
nqus <- length( qus )
fits <- qdo(fitM, qus, getViz, nsim = 0)
pl <- list()
for( ii in 1:4 ){
 smo <- lapply(fits, sm, ii)
 pl[[ii]] <- lapply(smo, plot, n = 200)
}
EdfColors <- c("#FE5815", "#FFA02F", "#C4D600", "#509E2F", "#005BBB", "#001A70")
EdfColors <- colorRampPalette(EdfColors[1:4])( nqus )
basic <- list()
for( ii in 1:4 ){ # Loop over effects
  basic[[ii]] <- pl[[ii]][[1]]
  for( kk in 1:nqu ){ # Loop over nqu quantiles
    basic[[ii]] <- basic[[ii]] + geom_line(data = pl[[ii]][[kk]]$data$fit,
                                           colour = EdfColors[kk], na.rm = T)
  }
}
# Green high quantiles, orange low quantiles
```

```
library(gridExtra)
grid.arrange(grobs = lapply(basic, "[[", "ggObj"))
```

# B Confidence intervals for model checks

Let `fit3` be the `qgam` model corresponding to quantile 0.45. Then copy-paste this on the console:

```
# Function that returns the appropriate "gridFun" function, depending on quantile .qu,
# confidence level .lev and type of confidence line .type.
funCreator <- function(.qu, .lev, .type){

   ciFun <- function(.x){

     .n <- length(.x)

     if( .type == "lower" ){ return( qbinom(.lev/2, .n, .qu) / .n ) }

     if( .type == "upper" ){ return( qbinom(.lev/2, .n, .qu, lower.tail = FALSE) / .n ) }

   }

}
# Getting out functions
qus <- seq(0.05, 0.95, length.out =  10)
lowCI <- funCreator(.qu = qus[5], .lev = 0.05, .type = "lower")
upCI  <- funCreator(.qu = qus[5], .lev = 0.05, .type = "upper")
pl <- list()
pl[[1]] <- check1D(fit3, x = "wM_s95") +
  l_gridCheck1D(gridFun = function(.x){sum(.x < 0)/length(.x)},
                level = 0, shape = 3, size = 2) +
  l_gridCheck1D(gridFun = lowCI, level = 0, colour = 2) +
  l_gridCheck1D(gridFun = upCI, level = 0, colour = 3) +
  geom_hline(yintercept = qus[5], linetype = 2)
pl[[2]] <- check1D(fit3, x = "Posan") +
  l_gridCheck1D(gridFun = function(.x){sum(.x < 0)/length(.x)},
                 level = 0, shape = 3, size = 2) +
  l_gridCheck1D(gridFun = lowCI, level = 0, colour = 2) +
  l_gridCheck1D(gridFun = upCI, level = 0, colour = 3) +
  geom_hline(yintercept = qus[5], linetype = 2)
library(gridExtra)
grid.arrange(grobs = lapply(pl, "[[", 1), ncol = 2)
```