Using the package `hypergsplines`: some examples.

Daniel Sabanés Bové

21st November 2013

This short vignette shall introduce into the usage of the package `hypergsplines`. For more information on the methodology, see the technical report (Sabanés Bové, Held, and Kauermann, 2011).

If you have any questions or critique concerning the package, write an email to me: `daniel.sabanesbove@ifspm.uzh.ch`. Many thanks in advance!

0.1 Pima Indians diabetes data

First, we will have a look at the Pima Indians diabetes data, which is available in the package `MASS`:

```
> library(MASS)
> pima <- rbind(Pima.tr, Pima.te)
> pima$hasDiabetes <- as.numeric(pima$type == "Yes")
> pima.nObs <- nrow(pima)
```

Setup For $n = 532$ women of Pima Indian heritage, seven possible predictors for the presence of diabetes are recorded. We would like to investigate with an additive logistic regression, which of them are relevant, and what form the statistical association has – is it a linear effect, or rather a nonlinear effect? We will use a fully and automatic Bayesian approach for this, see the technical report (Sabanés Bové et al., 2011) for more details.

The first step is to define a `modelData` object, where we input the response vector `y`, the matrix with the covariates, the spline type (here we use `cubic` O'Sullivan splines with 4 inner knots) and the exponential family (here a canonical `binomial` model, i.e. we want logistic regression):

```
> library(hypergsplines)
> modelData.pima <- with(pima,
+                        glmModelData(y=hasDiabetes,
+                                     X=
+                                     cbind(npreg,
+                                           glu,
+                                           bp,
+                                           skin,
+                                           bmi,
+                                           ped,
+                                           age),
+                                     splineType="cubic",
+                                     nKnots=4L,
+                                     family=binomial))
```

Stochastic model search

```
> chainlength.pima <- 100
> computation.pima <- getComputation(useOpenMP=FALSE,
+                                    higherOrderCorrection=FALSE)
```

Next, we will do a stochastic search on the (very large) model space to find "good" models. Here we have to decide on the model prior, and in this example we use the `dependent` type which corrects for the implicit multiplicity of testing. We use a `chainlength` of 100, which is very small but enough for illustration purposes (usually one should use at least 100 000), and save all models (in general `nModels` is the number of models which are saved from all visited models). Finally, we decide that we do not want to use OpenMP acceleration and no higher order correction for the Laplace approximations. In order to be able to reproduce the analysis, it is advisable to set a seed for the random number generator before starting the stochastic search.

```
> set.seed(93)
> time.pima <-
+     system.time(models.pima <-
+                 stochSearch(modelData=modelData.pima,
+                             modelPrior="dependent",
+                             chainlength=chainlength.pima,
+                             nModels=chainlength.pima,
+                             computation=computation.pima))

----25---50---75---100
----|----|----|----|
====================

Number of non-identifiable model proposals:     0
Number of total cached models:                  77
Number of returned models:                      77
```

Wee see that the search took 11 seconds, and 77 models were found. The "models" list element of `models.pima` gives the table of the found models, with their degrees of freedom for every covariate, the log marginal likelihood, the log prior probability, the posterior probability and the number of times that the sampler encountered that model:

```
> head(models.pima$models)

  npreg glu bp skin bmi ped age logMargLik  logPrior       post hits
1     3   1  0    0   4   2   4  -242.2493 -14.08276 0.28527824    2
2     3   1  0    0   4   2   3  -242.4728 -14.08276 0.22813320    2
3     3   1  0    0   3   2   3  -242.5401 -14.08276 0.21329269    2
4     1   1  0    0   2   2   2  -244.2944 -14.08276 0.03690369    5
5     3   1  1    0   4   2   4  -243.6871 -14.77591 0.03386998    3
6     0   1  0    3   4   2   3  -244.6538 -14.08276 0.02576411    0
```

```
> map.pima <- models.pima$models[1, 1:7]
```

We have saved the degrees of freedom vector of the estimated MAP model in `map.pima`.

Inclusion probabilities The estimated marginal inclusion probabilities (probabilities for exclusion, linear inclusion and nonlinear inclusion) for all covariates are also saved:

```
> round(models.pima$inclusionProbs,2)
```

```
             npreg  glu   bp skin  bmi  ped age
not included  0.03 0.00 0.91 0.86 0.00 0.00   0
linear        0.07 0.96 0.09 0.09 0.04 0.01   0
non-linear    0.90 0.04 0.00 0.05 0.96 0.99   1
```

Sampling model parameters If we now want to look at the estimated covariate effects in the estimated MAP model which has configuration (3,1,0,0,4,2,4), then we first need to generate parameter samples from that model:

```
> mcmc.pima <- getMcmc(samples=500L,
+                      burnin=100L,
+                      step=1L,
+                      nIwlsIterations=2L)
> set.seed(634)
> map.samples.pima <- glmGetSamples(config=map.pima,
+                                   modelData=modelData.pima,
+                                   mcmc=mcmc.pima,
+                                   computation=computation.pima)

----25---50---75---100
----|----|----|----|
====================
```

With the function `getMcmc`, we have defined a list of MCMC settings, comprising the number of samples we would like to have in the end, the length of the burn-in, the thinning step (here no thinning) and the number of IWLS iterations used (with 2 steps you get a higher acceptance rate than with 1 step, here the acceptance rate was 0.52). The result `map.samples.pima` has the following structure:

```
> str(map.samples.pima)
```

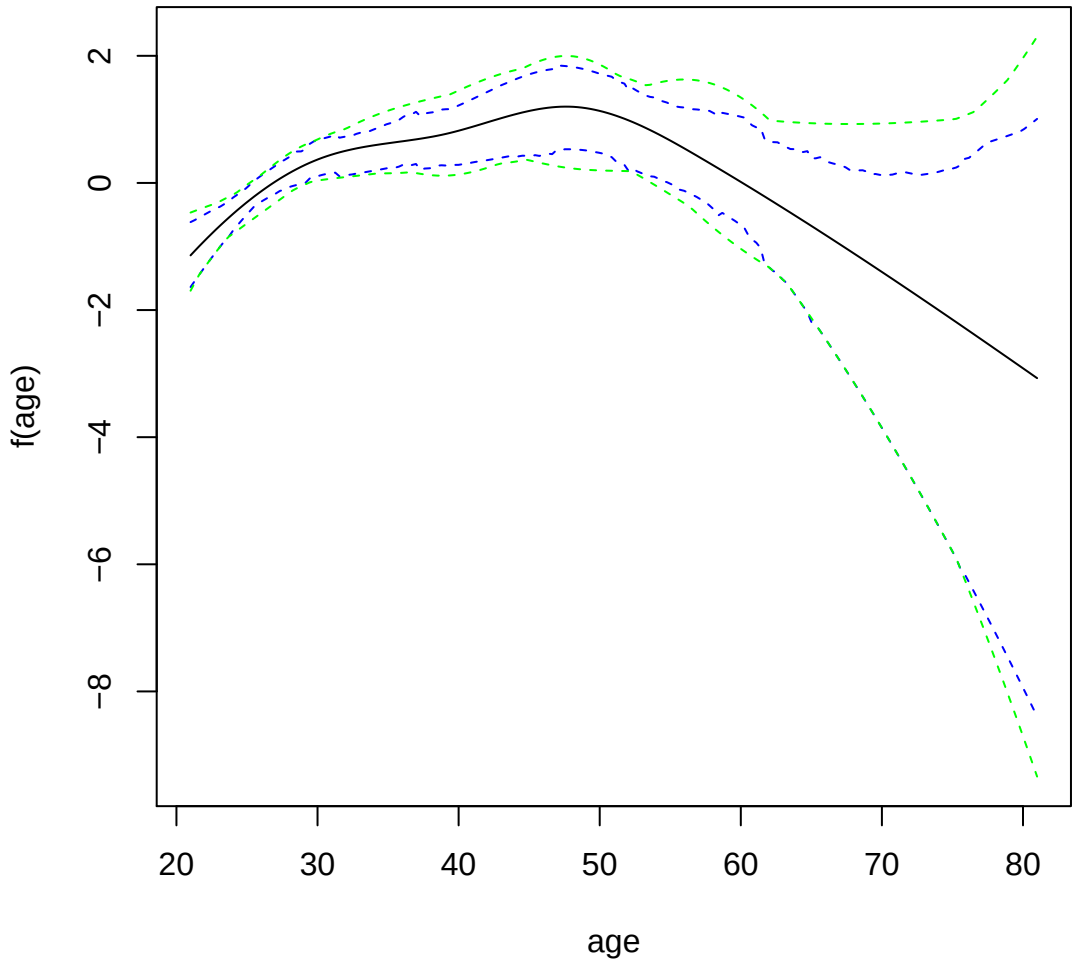
```
List of 3
 $ samples   :List of 5
  ..$ t          : num [1:500] 0.992 0.992 0.992 0.992 0.997 ...
  ..$ intercept  : num [1:500(1d)] -0.946 -0.946 -0.946 -0.946 -0.905 ...
  ..$ linearCoefs:List of 5
```

```
  .. ..$ npreg: num [1:500(1d)] -0.354 -0.354 -0.354 -0.354 1.223 ...
  .. ..$ glu  : num [1:500(1d)] 5.81 5.81 5.81 5.81 4.86 ...
  .. ..$ bmi  : num [1:500(1d)] 4.15 4.15 4.15 4.15 3.93 ...
  .. ..$ ped  : num [1:500(1d)] 4.09 4.09 4.09 4.09 2.46 ...
  .. ..$ age  : num [1:500(1d)] 3.67 3.67 3.67 3.67 2.01 ...
  ..$ splineCoefs:List of 4
  .. ..$ npreg: num [1:6, 1:500] 10.55 -8.13 11.9 -12.16 1.97 ...
  .. ..$ bmi  : num [1:6, 1:500] 12.13 -17 66.08 -3.97 -4.98 ...
  .. ..$ ped  : num [1:6, 1:500] -7.12 7.86 1.71 -6.35 -7.14 ...
  .. ..$ age  : num [1:6, 1:500] 46.16 26.28 4.27 6.25 -18.3 ...
  ..$ z          : num [1:500] 4.87 4.87 4.87 4.87 5.98 ...
 $ mcmc      :List of 2
  ..$ nAccepted      : num 309
  ..$ acceptanceRatio: num 0.515
 $ logMargLik:List of 4
  ..$ ilaEstimate    : num -242
  ..$ mcmcEstimate   : num -242
  ..$ mcmcSe         : num 0.114
  ..$ margApproxZdens:List of 2
  .. ..$ args: num [1:100] -52.0976 -29.1796 -0.8346 -0.0443 0.4362 ...
  .. ..$ dens: num [1:100] 0.00 1.41e-51 7.72e-30 3.36e-23 1.09e-18 ...
```

It is a list with the `samples`, two diagnostics for the `mcmc`, and estimates for the log marginal likelihood (`logMargLik`). The latter one contains the original ILA estimate, the MCMC estimate of the log marginal likelihood with its standard error, and the coordinates of the posterior density of $z = \log(g)$.

Curve estimates Now we can use the samples to plot the estimated effects of the MAP model covariates, with the `plotCurveEstimate` function. For example:

```
> plotCurveEstimate(covName="age",
+                   samples=map.samples.pima$samples,
+                   modelData=modelData.pima)
```



Post-processing If we want to have estimates of the degrees of freedom on a continuous scale instead of the fixed grid $(0, 1, 2, 3, 4)$, we can optimise the marginal likelihood with respect to the degrees of freedom of the MAP covariates:

```
> optim.map.pima <- postOptimize(modelData=modelData.pima,
+                                modelConfig=map.pima,
+                                computation=computation.pima)
> optim.map.pima

     npreg glu bp skin      bmi      ped      age
1 2.276746   1  0    0 3.555901 2.103911 3.653702
```

For that model, we could again produce samples and plot curve estimates.

Prediction samples If we would like to get prediction samples for new covariate values, this is also very easy via the `getFitSamples` function. Here we get posterior predictive samples because we input a covariate matrix which is part of the original covariate matrix used to fit the MAP model. Because the `getFitSamples` function produces samples on the linear predictor scale, we have to apply the appropriate response function (here the logistic distribution function `plogis`) to get samples on the observation scale.

```
> fit.samples.pima <- getFitSamples(X=modelData.pima$origX[1:10,],
+                                   samples=map.samples.pima$samples,
+                                   modelData=modelData.pima)
> obs.samples.pima <- plogis(fit.samples.pima)
```

The posterior predictive means are thus:

```
> rowMeans(obs.samples.pima)

 [1] 0.04672958 0.63051441 0.08926869 0.69338988 0.03227731 0.29412741
 [7] 0.06373851 0.58590940 0.28167756 0.68851588
```

and could be compared to the actual observations

```
> modelData.pima$Y[1:10]

 [1] 0 1 0 0 0 1 0 0 0 1
```

Model averaging Model averaging works in principle similar to sampling from a single model, but multiple model configurations are supplied and their respective log posterior probabilities. For example, if we wanted to average the top ten models found, we would do the following:

```
> average.samples.pima <-
+     with(models.pima,
+          getBmaSamples(config=models[1:10,],
+                        logPostProbs=models$logMargLik[1:10] +
+                        models$logPrior[1:10],
+                        nSamples=500L,
+                        modelData=modelData.pima,
+                        mcmc=mcmc.pima,
+                        computation=computation.pima))

----25---50---75---100
----|----|----|----|
====================
```

Then internally, first the models are sampled, and for each sampled model so many samples are drawn as determined by the model frequency in the model average sample. On this sample object, the above presented functions can again be applied (e.g. `plotCurveEstimate`).

To be continued ...

Bibliography

D. Sabanés Bové, L. Held, and G. Kauermann. Hyper-g priors for generalised additive model selection with penalised splines. Technical report, University of Zurich and University Bielefeld, 2011. URL `http://arxiv.org/abs/1108.3520`.