over, when the response data is very noisy (i.e., low signal–to–noise ratio), `tgp` can be expected to partition heavily under the `bprior="bflat"` prior. In such cases, one of the other proper priors like the full hierarchical `bprior="b0"` or the independent `bprior="b0tau"` might be preferred.

3.5 Friedman data

This Friedman data set is the first one of a suite that was used to illustrate MARS (Multivariate Adaptive Regression Splines) [10]. There are 10 covariates in the data ($\mathbf{x} = \{x_1, x_2, \ldots, x_{10}\}$). The function that describes the responses (Z), observed with standard Normal noise, has mean

$$E(Z|\mathbf{x}) = \mu = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5, \qquad (18)$$

but depends only on $\{x_1, \ldots, x_5\}$, thus combining nonlinear, linear, and irrelevant effects. Comparisons are made on this data to results provided for several other models in recent literature. Chipman et al. [5] used this data to compare their treed LM algorithm to four other methods of varying parameterization: linear regression, greedy tree, MARS, and neural networks. The statistic they use for comparison is root mean-square error (RMSE)

$$\text{MSE} = \textstyle\sum_{i=1}^{n}(\mu_i - \hat{z}_i)^2/n \qquad\qquad \text{RMSE} = \sqrt{\text{MSE}}$$

where $\hat{z}_i$ is the model–predicted response for input $\mathbf{x}_i$. The $\mathbf{x}$'s are randomly distributed on the unit interval.

Input data, responses, and predictive locations of size $N = 200$ and $N' = 1000$, respectively, can be obtained by a function included in the `tgp` package.

```
> f <- friedman.1.data(200)
> ff <- friedman.1.data(1000)
> X <- f[, 1:10]
> Z <- f$Y
> XX <- ff[, 1:10]
```

This example compares Bayesian treed LMs with Bayesian GP LLM (not treed), following the RMSE experiments of Chipman et al. It helps to scale the responses so that they have a mean of zero and a range of one. First, fit the Bayesian treed LM, and obtain the RMSE.

```
> fr.btlm <- btlm(X = X, Z = Z, XX = XX, tree = c(0.95,
+     2), pred.n = FALSE, m0r1 = TRUE, verb = 0)
> fr.btlm.mse <- sqrt(mean((fr.btlm$ZZ.mean - ff$Ytrue)^2))
> fr.btlm.mse

[1] 1.939446
```

Next, fit the GP LLM, and obtain its RMSE.

```
> fr.bgpllm <- bgpllm(X = X, Z = Z, XX = XX, pred.n = FALSE,
+     m0r1 = TRUE, verb = 0)
> fr.bgpllm.mse <- sqrt(mean((fr.bgpllm$ZZ.mean - ff$Ytrue)^2))
> fr.bgpllm.mse

[1] 0.4241515
```

So, the GP LLM is 4.573 times better than Bayesian treed LM on this data, in terms of RMSE (in terms of MSE the GP LLM is 2.138 times better).

Parameter traces need to be gathered in order to judge the ability of the GP LLM model to identify linear and irrelevant effects.

```
> XX1 <- matrix(rep(0, 10), nrow = 1)
> fr.bgpllm.tr <- bgpllm(X = X, Z = Z, XX = XX1, pred.n = FALSE,
+     trace = TRUE, verb = 0)
```

Notice that the `m0r1=TRUE` has been omitted so that the $\boldsymbol{\beta}$ estimates provided below will be on the original scale. A summary of the parameter traces show that the Markov chain had the following (average) configuration for the booleans.

```
> apply(fr.bgpllm.tr$trace$XX[[1]][, 27:36], 2, mean)

 b1  b2  b3  b4  b5  b6  b7  b8  b9 b10
  1   1   1   0   0   0   0   0   0   0
```

Therefore the GP LLM model correctly identified that only the first three input variables interact only linearly with the response. This agrees with dimension–wise estimate of the total area of the input domain under the LLM (out of a total of 10 input variables).

```
> mean(fr.bgpllm.tr$trace$linarea$ba)

[1] 7
```

A similar summary of the parameter traces for $\boldsymbol{\beta}$ shows that the GP LLM correctly identified the linear regression coefficients associated with the fourth and fifth input covariates (from (18))

```
> summary(fr.bgpllm.tr$trace$XX[[1]][, 9:10])

     beta4            beta5
 Min.   : 8.623   Min.   :4.309
 1st Qu.: 9.370   1st Qu.:5.176
 Median : 9.564   Median :5.376
 Mean   : 9.550   Mean   :5.375
 3rd Qu.: 9.735   3rd Qu.:5.582
 Max.   :10.431   Max.   :6.313
```

and that the rest are much closer to zero.

```
> apply(fr.bgpllm.tr$trace$XX[[1]][, 11:15], 2, mean)

      beta6       beta7       beta8       beta9      beta10
-0.23968561  0.37046946  0.13081722 -0.07842566  0.11911203
```

3.6 Adaptive Sampling

In this section, sequential design of experiments, a.k.a. *adaptive sampling*, is demonstrated on the exponential data of Section 3.3. Gathering, again, the data:

```
> exp2d.data <- exp2d.rand(lh = 0, dopt = 10)
> X <- exp2d.data$X
> Z <- exp2d.data$Z
> Xcand <- lhs(1000, rbind(c(-2, 6), c(-2, 6)))
```

In contrast with the data from Section 3.3, which was based on a grid, the above code generates a randomly subsampled D–optimal design $\mathbf{X}$ from LH candidates, and random responses $\mathbf{Z}$. As before, design configurations are more densely packed in the interesting region. Candidates $\tilde{\mathbf{X}}$ are from a large LH–sample.

Given some data $\{\mathbf{X}, \mathbf{Z}\}$, the first step in sequential design using `tgp` is to fit a treed GP LLM model to the data, without prediction, in order to infer the MAP tree $\hat{\mathcal{T}}$.

```
> exp1 <- btgpllm(X = X, Z = Z, pred.n = FALSE, corr = "exp",
+     verb = 0)
```

The trees are shown in Figure 15. Then, use the `tgp.design` function to create D–optimal candidate designs in each region of $\hat{\mathcal{T}}$. For the purposes of illustrating the `improv` statistic, I have manually added the known (from calculus) global minimum to `XX`.

```
> XX <- tgp.design(200, Xcand, exp1)

sequential treed D-Optimal design in 3 partitions
dopt.gp (1) choosing 55 new inputs from 272 candidates
dopt.gp (2) choosing 53 new inputs from 263 candidates
dopt.gp (3) choosing 93 new inputs from 465 candidates

> XX <- rbind(XX, c(-sqrt(1/2), 0))
```

Figure 16 shows the sampled `XX` locations (circles) amongst the input locations `X` (dots) and MAP partition ($\hat{\mathcal{T}}$). Notice how the candidates `XX` are spaced out relative to themselves, and relative to the inputs `X`, unless they are near partition boundaries. The placing of configurations near region boundaries is a symptom particular to D–optimal designs. This is desirable for experiments with `tgp` models, as model uncertainty is usually high there [3].

Now, the idea is to fit the treed GP LLM model, again, in order to assess uncertainty in the predictive surface at those new candidate design points. The following code gathers all three adaptive sampling statistics: ALM, ALC, & EI.

```
> exp.as <- btgpllm(X = X, Z = Z, XX = XX, corr = "exp",
+     improv = TRUE, Ds2x = TRUE, verb = 0)
```