

The first part `(r,1)=(5000,104)` is indicating the MCMC round number `r=5000` and the number of leaves waiting to be "consumed" for prediction by the parallel prediction thread. When parallelization is turned off (default), the print will simply be `"r=5000"`.

The second part is a printing of the d -(range) parameter to a separable correlation function. For 2 partitions there are two sets of square brackets. Inside the square brackets is the m_X (2 in this case) range parameters for the separable correlation function. Whenever the LLM governs one of the input dimensions a zero will appear. I.e., the placement of `0/0.626` indicates the LLM is active in the 2nd dimension of the 2nd partition. `0.626` is the d -(range) parameter that would have been used if the LLM were inactive. Whenever all dimensions are under the LLM, the d -parameter print is simply `[0]`. This also happens when forcing the LLM (i.e., for `blm` and `btlm`), where `[0]` appears for each partition. These prints will look slightly different if the isotropic instead of separable correlation is used, since there are not as many range parameters.

B.3 Collaboration with `predict.tgp`

In this section I revisit the motorcycle accident data in order to demonstrate how the `predict.tgp` function can be helpful in collaborative uses of `tgp`. Consider a fit of the motorcycle data, and suppose that we do inference for the model parameters only (obtaining no samples from the posterior predictive distribution). The `"tgp"`-class output object can be saved to a file using the R-internal `save` function.

```
> library(MASS)
> out <- btgpllm(X = mcycle[, 1], Z = mcycle[, 2],
+   bprior = "b0", mOr1 = TRUE, pred.n = FALSE, verb = 0)
> save(out, file = "out.Rsave")
> out <- NULL
```

Note that there is nothing to plot here because there is no predictive data. (`out <- NULL` is set for illustrative purposes.)

Now imagine e-mailing the "out.Rsave" file to a collaborator who wishes to use your fitted `tgp` model. S/he could first load in the `"tgp"`-class object we just saved, design a new set of predictive locations `XX` and obtain kriging estimates from the MAP model.

```
> load("out.Rsave")
> XX <- seq(2.4, 56.7, length = 200)
> out.kp <- predict(out, XX = XX, pred.n = FALSE)
```

Another option would be to sample from the posterior predictive distribution of the MAP model.

```
> out.p <- predict(out, XX = XX, pred.n = FALSE, BTE = c(0,
+   1000, 1))
```

This holds the parameterization of the `tgpr` model *fixed* at the MAP, and samples from the GP or LM posterior predictive distributions at the leaves of the tree.

Finally, the MAP parameterization can be used as a jumping-off point for more sampling from the joint posterior and posterior predictive distribution.

```
> out2 <- predict(out, XX, pred.n = FALSE, BTE = c(0,
+          2000, 2), MAP = FALSE)
```

Since the return-value of a `predict.tgpr` call is also a `"tgpr"`-class object the process can be applied iteratively. That is, `out2` can also be passed to `predict.tgpr`.

Figure 22 plots the posterior predictive surfaces for each of the three calls to `predict.tgpr` above. The kriging surfaces are smooth within regions of the partition, but the process is discontinuous across partition boundaries. The middle surface is simply a Monte Carlo-sample summarization of the kriging one above it. The final surface summarizes samples from the posterior predictive distribution when obtained jointly with $\mathcal{T}|\theta$ and $\theta|\mathcal{T}$. Though these summaries are still “noisy” they depict a process with smoother transitions across partition boundaries than ones conditioned only on the MAP.

The `predict.tgpr` function can also sample from the ALC statistic and calculate expected improvements (EI) at the `XX` locations.

C Configuration and performance optimization

In what follows I describe customizations and enhancements that can be made to `tgpr` at compile time in order to take advantage of custom computing architectures. The compilation of `tgpr` with a linear algebra library different from the one used to compile R (e.g., `ATLAS`), and the configuration and compilation of `tgpr` with parallelization is described in detail.

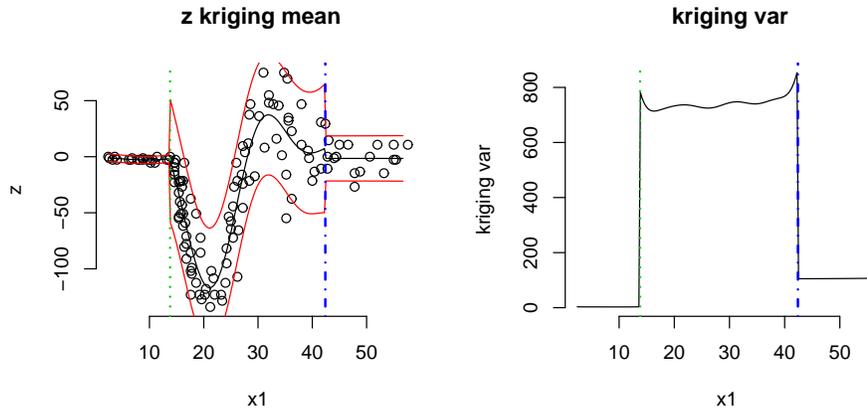
C.1 Linking to `ATLAS`

`ATLAS` [26] is supported as an alternative to standard BLAS and LAPACK for fast, automatically tuned, linear algebra routines. Compared to standard BLAS/Lapack, those automatically tuned by `ATLAS` are significantly faster. If you know that R has already been linked to tuned linear algebra libraries (e.g., on `OSX`), then compiling with `ATLAS` as described below, is unnecessary—just install `tgpr` as usual. As an alternative to linking `tgpr` to `ATLAS` directly, one could re-compile all of R linking it to `ATLAS`, or some other platform-specific BLAS/Lapack, i.e., Intel’s Math Kernel Library, or AMD’s Core Math Library, as described in:

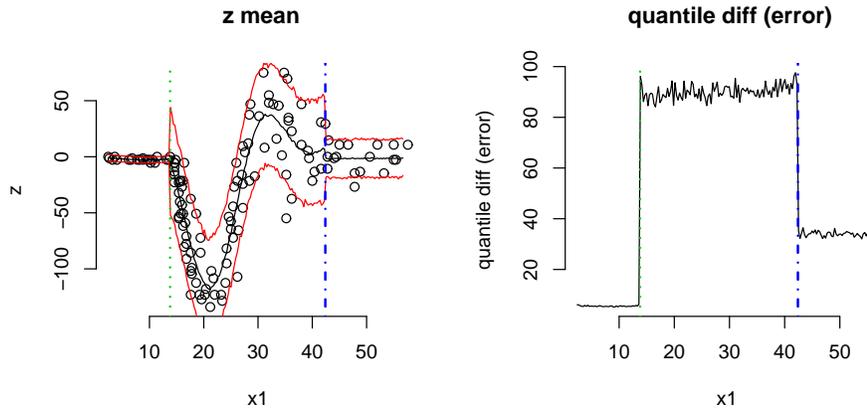
<http://cran.r-project.org/doc/manuals/R-admin.html>

Look for the section titled “Linear Algebra”. While this is arguably best solution since all of R benefits, the task can prove challenging to accomplish and may require administrator (root) privileges. Linking `tgpr` with `ATLAS` directly is described here.

```
> plot(out.kp, center = "km", as = "ks2")
```



```
> plot(out.p)
```



```
> plot(out2)
```

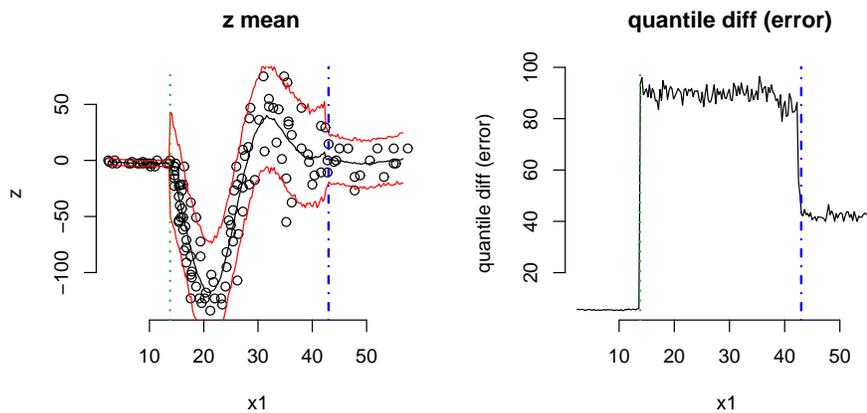


Figure 22: Predictive surfaces (*left*) and error/variance plots (*right*) resulting from three different uses of the `predict.tgp` function: MAP kriging (*top*), sampling from the MAP (*middle*), sampling from the joint posterior and posterior predictive starting from the MAP (*bottom*).