

# gprege Quick Guide

Alfredo A. Kalaitzis, Neil D. Lawrence

October 13, 2015

## 1 Abstract

The *gprege* package implements our methodology of Gaussian process regression models for the analysis of microarray time series. The package can be used to filter quiet genes and quantify/rank differential expression in time-series expression ratios. The purpose of this quick guide is to present a few examples of using *gprege*.

## 2 Citing *gprege*

Citing *gprege* in publications will involve citing the methodology paper [Kalaitzis and Lawrence, 2011] that the software is based on as well as citing the software package itself.

## 3 Introductory example analysis - TP63 microarray data

In this section we introduce the main functions of the *gprege* package by repeating some of the analysis from the BMC Bioinformatics paper [Kalaitzis and Lawrence, 2011].

### 3.1 Installing the *gprege* package

The recommended way to install *gprege* is to use the `biocLite` function available from the bioconductor website. Installing in this way should ensure that all appropriate dependencies are met.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("gprege")
```

Otherwise, to manually install the *gprege* software, unpack the software and run

```
R CMD INSTALL gprege
```

To load the package start R and run

```
> library(gprege)
```

## 3.2 Loading the data

For convenience, a fragment of the preprocessed data is provided in this package and can be loaded using

```
> data(FragmentDellaGattaData)
```

The full dataset is available at <http://gprege.googlecode.com/svn/trunk/DellaGattaData.RData> and can be loaded using

```
> # Download full Della Gatta dataset.
> load(url('http://gprege.googlecode.com/svn/trunk/DellaGattaData.RData'))
```

Type `?DellaGattaData` for more details on the dataset. If you need to preprocess some gene expression time-series from scratch and the dataset originates from Affymetrix arrays, we highly recommend processing it with `rma` or `mmgmos` from the *affy* and *puma* packages respectively. `mmgmos` extracts error bars on the expression measurements directly from the array data to allow judging the reliability of individual measurements. A detailed guide on how to perform these steps can be found in the *tigre Quick Guide*.

## 3.3 Interactive mode

Let us now examine one by one, in an interactive fashion, a few profiles of the full dataset (not available in the package). We start by downloading it.

```
> # Download full Della Gatta dataset.
> con <- url('http://gprege.googlecode.com/svn/trunk/DellaGattaData.RData')
> while(!exists('DGdata')) try(load(con),TRUE) # close.connection(con)
> # Timepoints / GP inputs.
> tTrue = matrix(seq(0,240,by=20), ncol=1)
> gpregeOptions <- list()
```

These gene expression profiles correspond to the top ranks as direct targets suggested by TSNI [Della Gatta et al., 2008].

```
> # Set index range so that only a top few targets suggested by TSNI be explored.
> gpregeOptions$indexRange <- which(DGatta_labels_byTSNItop100)[1:2]
```

If the download fails, use the following instead.

```
> # Load installed fragment-dataset.
> data(FragmentDellaGattaData)
> # Fragment dataset is comprised of top-ranked targets suggested by TSNI.
> # Explore only the first few.
> gpregeOptions$indexRange <- 1:2
```

The option `gpregeOptions$explore<-TRUE` makes `gprege` wait for a keystroke to proceed to the next profile.

```
> # Explore individual profiles in interactive mode.
> gpregeOptions$explore <- TRUE
> # Exhaustive plot resolution of the LML function.
> gpregeOptions$exhaustPlotRes <- 30
> # Exhaustive plot contour levels.
> gpregeOptions$exhaustPlotLevels <- 10
> # Exhaustive plot maximum lengthscale.
> gpregeOptions$exhaustPlotMaxWidth <- 100
> # Noisy ground truth labels: which genes are in the top 786 ranks of the TSNI ranking.
> gpregeOptions$labels <- DGatta_labels_byTSNItop100
> # SCG optimisation: maximum number of iterations.
> gpregeOptions$iters <- 100
> # SCG optimisation: no messages.
> gpregeOptions$display <- FALSE
> # Matrix of different hyperparameter configurations as rows:
> # [inverse-lengthscale percent-signal-variance percent-noise-variance].
> gpregeOptions$inithypers <-
+   matrix( c( 1/1000,      1e-3,      0.999,
+             1/30,        0.999,      1e-3,
+             1/80,        2/3, 1/3
+           ), ncol=3, byrow=TRUE)
```

`gprege` will generate a report and a few plots for each explored profile. The first line contains the index of the profile in `exprs_tp63_RMA` and its ground truth (according to TSNI). The hyperparameters of the optimised log-marginal likelihood (LML) follow and then the initialisation (indicated by '<-') from which it came. Then a few statistics; the observed standard deviation of the time-series and the sum of the explained std.deviation (by the GP) and noise (Gaussian) std.deviation. Finally the function `exhaustivePlot` will reveal, through an exhaustive search in the hyperparameter space<sup>1</sup>, the approximate true maximum LML and corresponding hyperparameters.

```
> gpregeOutput<-gprege(data=exprs_tp63_RMA,inputs=tTrue,gpregeOptions=gpregeOptions)
```

```
=====
Profile 249                                     Label: 1
=====
Length-scale      Signal      Noise
84.73947          0.22901      0.03297

Init.le           LML          Best
32                2.95540268
5                 16.69472737
```

<sup>1</sup>`gpregeOptions$exhaustPlotMaxWidth` defines the limit of the *lengthscale* search range.

```

9                                16.69472737          <--

Total st.dev.      Estim.sig + noise
0.200493           0.261985

Log-ratio (max(LML[2:end]) - LML[1])
13.73932
===== EXHAUSTIVE LML SEARCH =====
Length-scale      Signal          Noise
72.68966          0.16579         0.03470

Max LML           Estim. sig + noise
16.32234561       0.200493

ENTER to continue

=====
Profile  370                                Label:  1
=====
Length-scale      Signal          Noise
185.38939         0.37661         0.10492

Init.le          LML              Best
32               0.25444534
5                6.37127488          <--
9                6.37127488

Total st.dev.      Estim.sig + noise
0.246752           0.481530

Log-ratio (max(LML[2:end]) - LML[1])
6.11683
===== EXHAUSTIVE LML SEARCH =====
Length-scale      Signal          Noise
62.44828          0.15310         0.09366

Max LML           Estim. sig + noise
5.08007151        0.246752

ENTER to continue

```

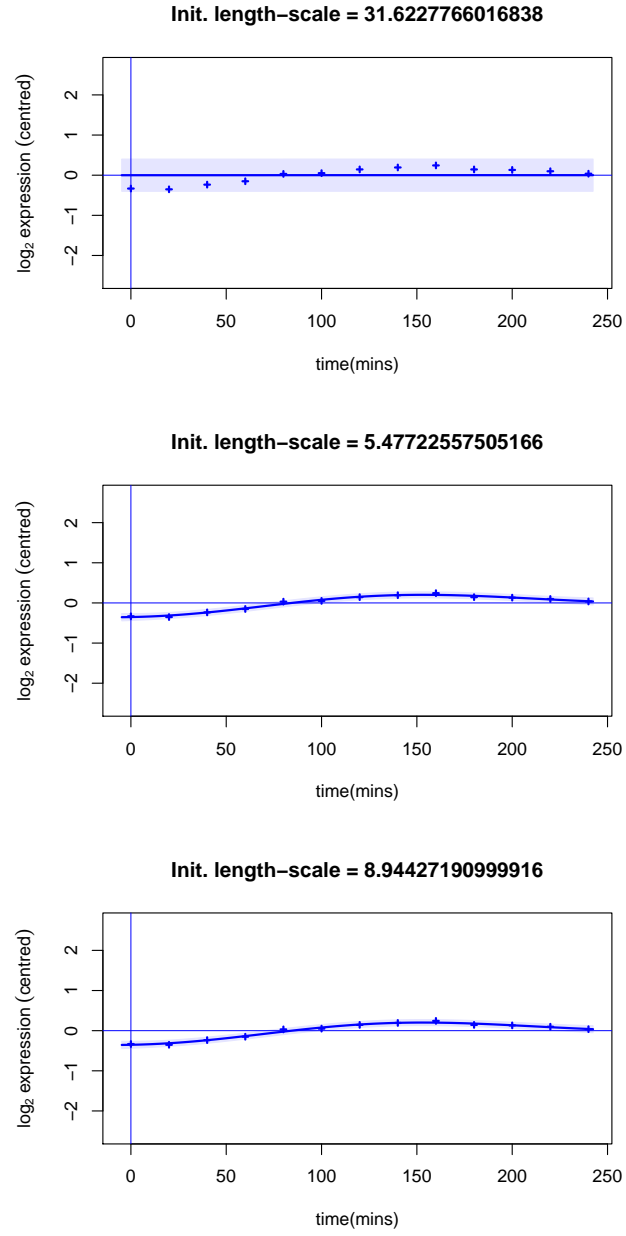


Figure 1: GP fit with different initialisations on profile #249.



(a)



(b)

Figure 2: Profile #249 : (a) Log-marginal likelihood (LML) contour. (b) GP fit with maximum LML hyperparameters from the exhaustive search.

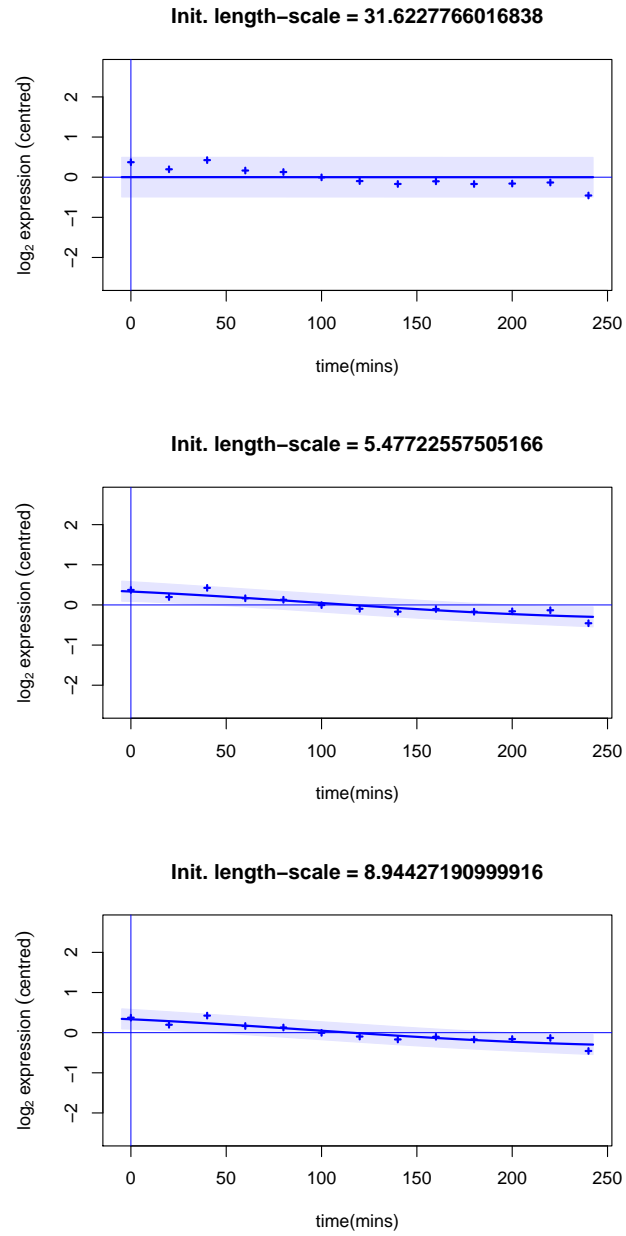


Figure 3: GP fit with different initialisations on profile #370.



Figure 4: Profile #370 : (a) Log-marginal likelihood (LML) contour. (b) GP fit with maximum LML hyperparameters from the exhaustive search.



### 3.4 Comparing against BATS

Now we demonstrate `compareROC`, a facility for comparing the performance of `gprege` on a dataset with some other method (see figures below). In this example, we reproduce the main result presented in [Kalaitzis and Lawrence, 2011]<sup>2</sup>. Specifically, we apply standard Gaussian process regression and BATS [Angelini et al., 2007] on experimental gene expression data, where only the top 100 ranks of TSNI were labelled as truly differentially expressed in the noisy ground truth. From the output of each model, a ranking of differential expression is produced and assessed with ROC curves to quantify how well in accordance to the noisy ground truth each method performs. For convenience, `gprege` was already run on the full DellaGatta dataset and the resulting rank metrics are stored in `gpregeOutput$rankingScores` (see `demTp63Gp1(fullldataset=TRUE)`).

```
> # Load fragment dataset.
> data(FragmentDellaGattaData)
> # Download BATS rankings (Angelini, 2007)
> # Case 1: Delta error prior, case 2: Inverse Gamma error prior,
> # case 3: Double Exponential error prior.
> BATSranking = matrix(0, length(DGatta_labels_byTSNItop100), 3)
> for (i in 1:3){
+   # Read gene numbers.
+   tmp = NULL
+   con <- url(paste('http://gprege.googlecode.com/svn/trunk/DGdat_p63_case',i,'_GL.txt'),s
+   while(is.null(tmp)) try(tmp <- read.table(con, skip=1), TRUE)
+   genenumbers <- as.numeric(lapply( as.character(tmp[,2]), function(x) x=substr(x,2,ncha
+   # Sort rankings by gene numbers.
+   BATSranking[,i] <- tmp[sort(genenumbers, index.return=TRUE)$ix, 4]
+ }
```

---

<sup>2</sup>Results on experimental data are slightly better because more initialisation points were used in optimising the likelihood wrt the kernel hyperparameters (`gpregeOptions$inithypers`), and the optimisation with the best log-marginal likelihood was always used in the end.

```

> # The smaller a BATS-rank metric is, the better the rank of the gene
> # reporter. Invert those rank metrics to compare on a common ground
> # with gprege.
> BATSranking = 1/BATSranking
> compareROC(output=gpregeOutput$rankingScores,
+           groundTruthLabels=DGatta_labels_byTSNItop100,
+           compareToRanking=BATSranking)

[1] 0.876575

```

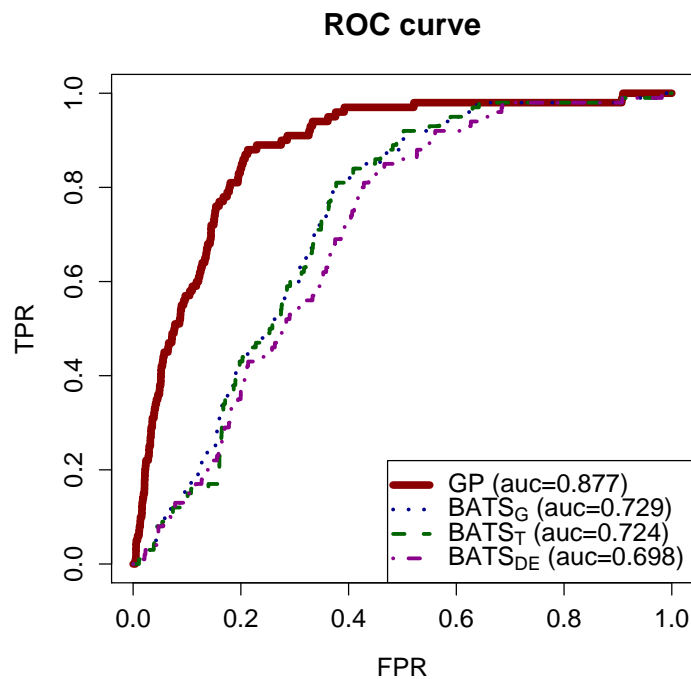


Figure 5: ROC comparison on experimental data from [Della Gatta et al., 2008]. One curve for the GP method and three for BATS, using a different noise model (subscript 'G' for Gaussian, 'T' for Student's-t and 'DE' for double exponential marginal distributions of error), followed by the area under the corresponding curve (AUC).

### 3.5 Ranking differential genes

Finally, the bulk ranking for differential expression is done with the full dataset, no interactive mode, and a ROC comparison in the end. Total computation time is approximately 45 minutes on a desktop running Ubuntu 10.04 with a dual-core CPU at 2.8 GHz and 3.2 GiB of memory.

```

> # Download full Della Gatta dataset.
> #con <- url('http://gprege.googlecode.com/svn/trunk/DellaGattaData.RData')
> #while(!exists('DGdata')) try(load(con),TRUE); close.connection(con)
> # Timepoints / GP inputs.
> tTrue = matrix(seq(0,240,by=20), ncol=1)
> gpregeOptions <- list()
> # Explore individual profiles in interactive mode.
> gpregeOptions$explore <- FALSE
> # Exhaustive plot resolution of the LML function.
> gpregeOptions$exhaustPlotRes <- 30
> # Exhaustive plot contour levels.
> gpregeOptions$exhaustPlotLevels <- 10
> # Exhaustive plot maximum lengthscale.
> gpregeOptions$exhaustPlotMaxWidth <- 100
> # Noisy ground truth labels: which genes are in the top 786 ranks of the TSNI ranking.
> gpregeOptions$labels <- DGatta_labels_byTSNItop100
> # SCG optimisation: maximum number of iterations.
> gpregeOptions$iters <- 100
> # SCG optimisation: no messages.
> gpregeOptions$display <- FALSE
> # Matrix of different hyperparameter configurations as rows:
> # [inverse-lengthscale percent-signal-variance percent-noise-variance].
> gpregeOptions$inithypers <-
+   matrix( c( 1/1000, 1e-3, 0.999,
+             1/8, 0.999, 1e-3,
+             1/80, 2/3, 1/3
+           ), ncol=3, byrow=TRUE)
> gpregeOutput<-gprege(data=exprs_tp63_RMA,inputs=tTrue,gpregeOptions=gpregeOptions)

```

## References

- C. Angelini, D. De Canditiis, M. Mutarelli, and M. Pensky. A Bayesian approach to estimation and testing in time-course microarray experiments. *Stat Appl Genet Mol Biol*, 6:24, 2007.
- G. Della Gatta, M. Bansal, A. Ambesi-Impimbato, D. Antonini, C. Missero, and D. di Bernardo. Direct targets of the TRP63 transcription factor revealed by a combination of gene expression profiling and reverse engineering. *Genome research*, 18(6):939, 2008.
- Alfredo A. Kalaitzis and Neil D. Lawrence. A simple approach to ranking differentially expressed gene expression time courses through gaussian process regression. *BMC Bioinformatics*, 12(180), 2011. doi: 10.1186/1471-2105-12-180.