An introduction to PLGEM

Mattia Pelizzola and Norman Pavelka

May 19, 2021

1 Introduction

This document serves as a brief tutorial to the **Power Law Global Error Model** (**PLGEM**) analysis method [1]. **PLGEM** has so far been shown to faithfully model the variance-versus-mean dependence that exists in a wide variety of genome-wide data sets, including microarray [1] and proteomics data [2]. The use of **PLGEM** has furthermore been shown to improve the detection of differentially expressed genes or proteins in these datasets [1, 2].

2 Running PLGEM in *wrapper* mode

A wrapper function (called `run.plgem`) is provided in the package, which performs all the necessary steps to obtain a list of differentially expressed genes or proteins (DEG), starting from a dataset of class *ExpressionSet*.

This input dataset is expected to contain either normalized gene expression values obtained from one-channel microarrays (such as Affymetrix GeneChip [1]), or normalized spectral counts obtained from mass spectrometry-based proteomics methods (such as MudPIT [2]).

The wrapper automatically attempts to find the best solution at each step, and requires only modest or no input decisions by the user. For didactic purposes, we will use here a subset of the microarray dataset used in the original publication about **PLGEM**, containing two replicates of LPS-stimulated dendritic cells ('LPS') and four replicates of untreated dendritic cells ('C'):

```
> library(plgem)
> data(LPSeset)
> set.seed(123)
> LPSdegList <- run.plgem(esdata=LPSeset)
```

The above obtained object `LPSdegList` will contain the list of genes or proteins, selected as significantly changing between experimental condition 'LPS' and baseline condition 'C', at the default significance level 0.001.

3 Running PLGEM in *step-by-step* mode

To provide advanced users with a higher control on the inner workings of the **PLGEM** pipeline, the individual functions called by `run.plgem` are also described in this tutorial. We will use them now, step by step.

3.1 Fitting of the model to a data set

The first step is to fit the model to the microarray or proteomics dataset. By fitting the model we will obtain a mathematical relationship that will allow us to determine the expected standard deviation associated to a given average gene expression value or average protein abundance level.

PLGEM can only be fitted on a set of replicates of a same experimental condition, therefore we first need to choose which condition to use for the fitting step. In our dataset two conditions are provided: 'C' and 'LPS'. Usually the most replicated one is chosen, in this example we will therefore choose the first condition 'C' (i.e. `fitCondition="C"`), because it contains four replicates. Technically, we may have decided to fit the model also on the two 'LPS' samples, as two replicates are required and sufficient to fit a PLGEM. But having 3 or more replicates improves the fitting and is usually recommended [2].

Moreover, we can can change the default values of parameters `p` and `q`. Briefly, `p` represents the number of intervals (or bins) used to partition the expression value range of the dataset. We observed that `p` can be modified over a wide range of values, without any major effects on the final results, except when it was chosen to close to the total number of genes or proteins in the dataset [1]. As a rule of thumb, `p` should be no more than one tenth of the number of genes or proteins. The default of 10 should therefore be appropriate for most microarray experiments, but could be set lower for proteomics data where less than 100 proteins were identified.

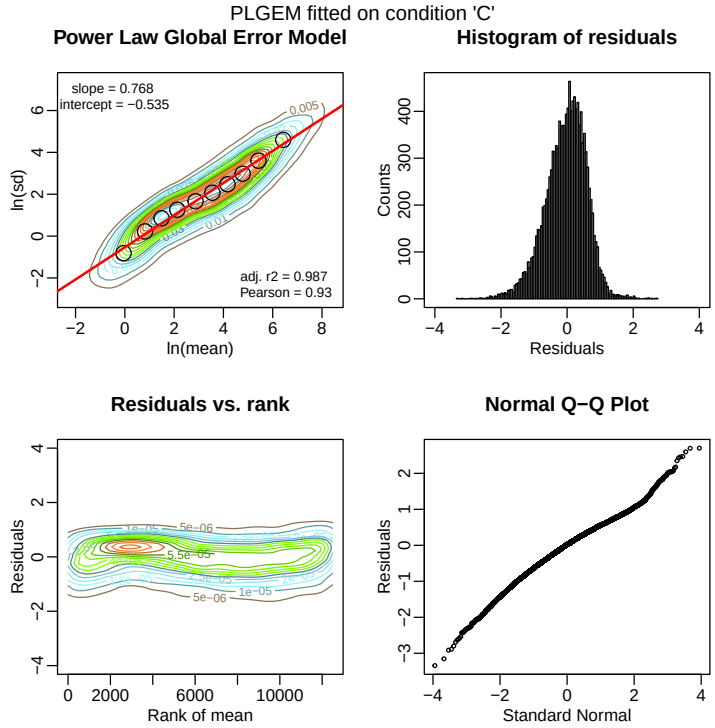
`q` is the quantile of the location-dependent spread used to fit the model. The default of `q` is set to 0.5, because this represents the median value, which is what you are looking for when modeling the variability. We recommend to only modify this parameter for very special purposes, e.g. for the determination of empirical confidence intervals of standard deviation.

Moreover it is possible to evaluate the fitting of the model setting the option `fittingEval`. If this argument is set to `TRUE`, a multi-panel plot is produced where the residuals of the model are evaluated. A good fit is characterized by a near-normal distribution and an horizontal symmetric plot of the ranked residuals.

Finally, setting `plot.file=TRUE` saves above diagnostic plot to a png file instead of the default device, therefore we won't change its default.

```
> LPSfit <- plgem.fit(data=LPSeset, covariate=1, fitCondition='C', p=10, q=0.5,
+   plot.file=FALSE, fittingEval=TRUE, verbose=TRUE)

Fitting PLGEM...
samples extracted for fitting:
   conditionName
C1             C
C2             C
C3             C
C4             C
replacing 7 zero standard deviations with smallest non-zero standard deviation...
determining modelling points...
fitting data and modelling points...
done with fitting PLGEM.
```



3.2 Computation of observed signal-to-noise ratios

The next step is the computation of the signal-to-noise ratio (STN) statistics for the detection of differential expression. The STN is determined using the model-derived spread estimates instead of the data-derived ones. Therefore it is necessary to give to the `plgem.obsStn` function the model parameters `slope` and `intercept` determined during the model fitting that are contained in the value returned by `plgem.fit` function. By default, all experimental conditions (according to the values of the `covariate` defined in the *phenoData* slot of the *ExpressionSet*) are compared to the first condition (baseline). If the condition to be treated as the baseline is not the first one, we can change this by modifying the argument `baselineCondition`. A matrix of observed STN is determined, where the number of rows are the number of genes or proteins in the dataset and the number of columns are the number of comparisons that can be performed in the dataset (i.e. the total number of experimental conditions minus one). In this case, the dataset contains only one condition to be compared to the baseline, therefore the matrix will be a one-dimensional array of observed PLGEM-STN values.

```
> LPSobsStn <- plgem.obsStn(data=LPSeset, covariate=1, baselineCondition=1,
+   plgemFit=LPSfit, verbose=TRUE)

calculating observed PLGEM-STN statistics:found 1 condition(s) to compare to the baseline.
working on baseline C ...
C1 C2 C3 C4
working on condition LPS ...
```

```
LPS1 LPS2
done with calculating PLGEM-STN statistics.
```

3.3 Computation of resampled signal-to-noise ratios

In order to get an estimate of the distribution of the test statistic under the null hypothesis of no differential expression, a resampled statistic is determined using the method described in the paper [1]. The number of iterations of the resampling step should be correlated with the total number of replicates that are present in the data set. If this argument is set to `"automatic"`, the number of iterations is automatically determined based on the total number of possible combinations. In this case, an upper threshold of 500 iterations is set to avoid excessive computation time. This should be fine for most purposes.

```
> set.seed(123)
> LPSresampledStn <- plgem.resampledStn(data=LPSeset, plgemFit=LPSfit,
+   iterations="automatic", verbose=TRUE)

calculating resampled PLGEM-STN statistics:found 1 condition(s) to compare to the baseline
baseline samples:
C1 C2 C3 C4
resampling on samples:
C1 C2 C3 C4
Using 500 iterations...
working on cases with 2 replicates...
     Iterations: 100  200  300  400  500
done with calculating resampled PLGEM-STN statistics.
```

3.4 Computation of p-values

Next, p-values are calculated for each observed STN value via a call to function `plgem.pValue`. Resampled STN values are also required by this function, because they will be used to build empirical cumulative distribution functions of the STN values that can be observed under the null hypothesis:

```
> LPSpValues <- plgem.pValue(observedStn=LPSobsStn,
+   plgemResampledStn=LPSresampledStn, verbose=TRUE)

calculating PLGEM p-values... done.

> head(LPSpValues)

             LPS_vs_C
100001_at 0.157816784
100002_at 0.148836323
100003_at 0.001264254
100004_at 0.216837604
100005_at 0.349002883
100006_at 0.510843690
```

3.5 Detection of differentially expressed genes or proteins (DEG)

Finally, DEG are selected at the given significance level `delta` via a call to function `plgem.deg`. The chosen value of `delta` can be seen as an estimate of the False Positive Rate (FPR). Therefore, in case of a microarray dataset with 10,000 genes of which not a single one is truly differentially expressed, choosing `delta=0.001` will select roughly 10 genes by chance:

```
> LPSdegList <- plgem.deg(observedStn=LPSobsStn, plgemPval=LPSpValues,
+   delta=0.001, verbose=TRUE)

selecting significant DEG:found 1 condition(s) compared to the baseline.
Delta =  0.001
        Condition =  LPS_vs_C
delta: 0.001 condition: LPS_vs_C found 360 DEG
done with selecting significant DEG.

> head(LPSdegList$significant[["0.001"]][["LPS_vs_C"]])

[1] "100012_at"   "100213_f_at" "100277_at"   "100278_at"   "100342_i_at"
[6] "100379_f_at"
```

Above function returns a list with a number of items that is equal to the number of different significance levels `delta` used as input. In this case the default single value of 0.001 was used, so the list will contain only one item at this level. This item is again a list, whose number of items correspond to the number of performed comparisons, i.e. the number of conditions in the starting *ExpressionSet* minus the baseline, in this case again only one. In each list-item the values are the observed STN and the names are the IDs of the significantly changing genes or proteins, as defined in the *ExpressionSet*.

Finally, the obtained list of DEG can be written to the current working directory using the `plgem.write.summary` function.

```
> sessionInfo()

R version 4.1.0 (2021-05-18)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.2 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.13-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.13-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
```

```
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] plgem_1.64.0

loaded via a namespace (and not attached):
[1] MASS_7.3-54         compiler_4.1.0      parallel_4.1.0
[4] tools_4.1.0         Biobase_2.52.0      BiocGenerics_0.38.0
```

References

[1] Pavelka, N., Pelizzola, M., Vizzardelli, C., Capozzoli, M., Splendiani, A., Granucci, F. and P. Ricciardi-Castagnoli (2004). A power law global error model for the identification of differentially expressed genes in microarray data. *BMC Bioinformatics*, 5:203.

[2] Pavelka, N., Fournier, M., L., Swanson, S., K., Pelizzola, M., Ricciardi-Castagnoli, P., Florens, L. and M. P. Washburn (2008). Statistical similarities between transcriptomics and quantitative shotgun proteomics data. *Molecular and Cellular Proteomics*, 7(4):631-44.