

BAGS: A Bayesian Approach for Geneset Selection.

Alejandro Quiroz-Zárate¹, Benjamin Haibe-Kains², and John Quackenbush¹

¹Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Boston, Massachusetts, United States of America

²Bioinformatics and Computational Genomics Laboratory, Institut de Recherches Cliniques de Montréal, Montreal, Quebec, Canada.

April 15, 2025

Contents

1	Introduction	2
1.1	Installation	2
1.2	Further help	2
1.3	Citing	2
2	An application in Breast cancer.	3
2.1	Example: Data analysis under a cross-sectional setting.	3
2.1.1	Data preprocessing stage	3
2.1.2	Gibb's sampler executable example	4
3	Session Info	5

1 Introduction

The *BAGS* package provides functions to perform statistical identification of gene functional classes that behave in a distinct manner between the phenotypes of interest for datasets under cross-sectional or time series designs. This package includes (i) functions to perform gene set comparison (ii) examples to visualize the results of such comparisons.

The *BAGS* package provides functions to perform statistical identification of gene functional classes that behave in a distinct manner on datasets with cross-sectional or time series design, having 2 and up to 5 different phenotypes of interest or 2 up to 5 different time points

1.1 Installation

BAGS requires *R* ($\geq 2.10.0$) installed. To install *BAGS* from bioconductor:

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+   install.packages("BiocManager")
> BiocManager::install("BAGS")
```

Load the *BAGS*, into your current workspace:

```
> library(BAGS)
```

1.2 Further help

To view the *BAGS* description and a summary of all the functions within *BAGS*, type the following:

```
> library(help=BAGS)
```

1.3 Citing

We are delighted if you use this package. Please do email us if you find a bug or have a suggestion. We would be very grateful if you could cite:

Quiroz-Zarate A, Haibe-Kains B and Quackenbush J (2013). *Manuscript in preparation*

2 An application in Breast cancer.

We will very briefly demonstrate the use of some functions in *BAGS* by providing its application on a cross-sectional datasets.

We use the *breastCancerVDX* data library from Bioconductor for demonstration purposes under a cross-sectional design. This data set corresponds to the data set from [1]. Minn, AJ and colleagues used Affymetrix U133A Gene Chips to profile gene expression in 286 fresh-frozen tumor samples from patients with lymph-node-negative breast cancer who were treated during 1980 – 95, but who did not receive systemic neoadjuvant or adjuvant therapy. These samples correspond from the data set used in [3] with GEO reference accession number GSE2034, from the tumor bank at the Erasmus Medical Center in Rotterdam, Netherlands. An additional 58 estrogen receptor-negative samples were added from [1] GEO (GSE5327). In total 209 tumor samples are classified as ER+ and 135 as ER-. Even though this data set comes from a 5-year follow-up design, the way the data is conceived for this analysis is cross-sectional.

2.1 Example: Data analysis under a cross-sectional setting.

This is an example on how to perform an analysis with the proposed method in [2] for a data set with cross-sectional design. This example is divided in two parts. The data preparation and the execution of the Gibb's sampler function.

2.1.1 Data preprocessing stage

The original gene expression data set Minn AJ and colleagues [1] has a U133A Affymetrix platform. The normalized data set was saved to the variable *vdx* in the *breastCancerVDX* data library from Bioconductor.

```
> library(BAGS)
> library(breastCancerVDX)
> library(Biobase)
> data(vdx)
> gene.expr=exprs(vdx) # Gene expression of the package
> vdx.annot=fData(vdx) # Annotation associated to the dataset
> vdx.clinc=pData(vdx) # Clinical information associated to the dataset
> # Identifying the sample identifiers associated to ER+ and ER- breast cancer
> er.pos=which(vdx.clinc$er==1)
> er.neg=which(vdx.clinc$er==0)
> # Only keep columns 1 and 3, probeset identifiers and Gene symbols respectively
> vdx.annot=vdx.annot[,c(1,3)]
> # Checking if the probeset are ordered with respect to the dataset
> all(rownames(gene.expr)==as.character(vdx.annot[,1]))
```

```

[1] TRUE

> # Checking if the sample identifiers are order with respect to the dataset
> all(colnames(gene.expr)==as.character(vdx.clinc[,1]))

[1] TRUE

> # Changing the row identifiers to the gene identifiers of interest
> rownames(gene.expr)=as.character(vdx.annot[,2])
> #= Because we have several measurements for a gene, we filter the genes
> # Function to obtain the genes with highest variability among phenotypes
> gene.nms.u=unique(rownames(gene.expr))
> gene.nms=rownames(gene.expr)
> indices=NULL
> for(i in 1:length(gene.nms.u))
+ {
+     aux=which(gene.nms==gene.nms.u[i])
+     if(length(aux)>1){
+         var.r = apply(cbind(apply(gene.expr[aux,er.pos],1,mean),
+                                 apply(gene.expr[aux,er.neg],1,mean)),1,var)
+         aux=aux[which.max(var.r)]
+     }
+     indices=c(indices,aux)
+ }
> #==== Only keep the genes with most variability among the phenotypes of interest
> gene.expr=gene.expr[indices,]
> gene.nams=rownames(gene.expr) # The gene symbols of interest are stored here

```

In order to implement the Gibb's sampler procedure the dataset needs to be transformed to a data set with functional class expression measurements. Only Molecular functions from GO with at least 5 genes are considered for this analysis

```

> data(AnnotationMFGO,package="BAGS")
> data.gene.grps=DataGeneSets(AnnotationMFGO,gene.nams,5)
> phntp.list=list(er.pos,er.neg)
> data.mcmc=MCMCDataSet(gene.expr,data.gene.grps$DataGeneSetsIds,phntp.list)

```

2.1.2 Gibb's sampler executable example

To implement the Gibbs sampler we need to define the empty objects in which the posterior samples of the parameters of interest are to be kept. The Gibb's sampler implementation is in the following way (toy example):

```

> noRow=dim(data.mcmc$y.mu)[1]
> noCol=unlist(lapply(phntp.list,length))
> iter=10000
> GrpSzs=data.gene.grps$Size
> YMu=data.mcmc$y.mu
> L0=rep(2,2)
> V0=rep(4,2)
> L0A=rep(3,1)
> V0A=rep(3,1)

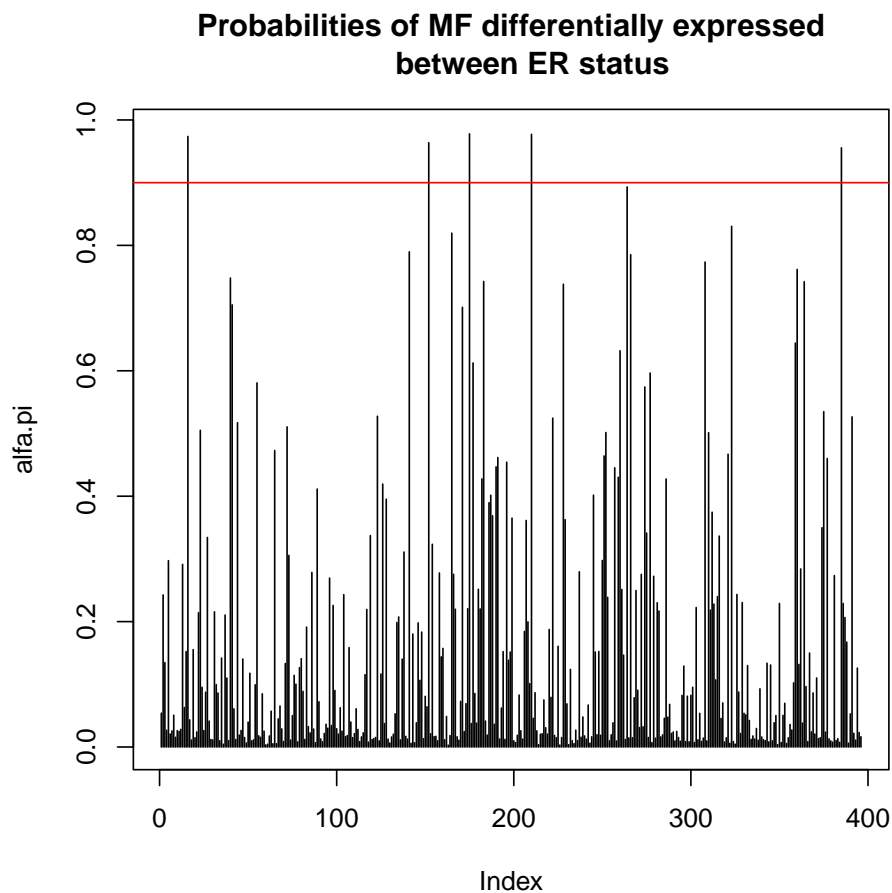
```

```

> MM=0.55
> AAPi=10
> ApriDiffExp=floor(dim(data.mcmc$y.mu)[1]*0.03)
> results=matrix(0,noRow,iter)
> mcmc.chains=Gibbs2(noRow,noCol,iter,GrpSzs,YMu,L0,V0,LOA,VOA,MM,AAPi,ApriDiffExp,
+                   results)

> burn.in=2000
> alfa.pi=apply(mcmc.chains[[1]][,burn.in:iter],1,function(x){
+               y=length(which(x!=0))/length(burn.in:iter);return(y)})
> plot(alfa.pi,type="h",main="Probabilities of MF differentially expressed
+     between ER status")
> cut.off=0.9
> abline(h=cut.off,col="red")
> differential.processes=names(data.gene.grps$Size)[which(alfa.pi>cut.off)]

```



3 Session Info

- R version 4.5.0 RC (2025-04-04 r88126 ucrt), x86_64-w64-mingw32

- Locale: LC_COLLATE=C, LC_CTYPE=English_United States.utf8, LC_MONETARY=English_United States.utf8, LC_NUMERIC=C, LC_TIME=English_United States.utf8
- Time zone: America/New_York
- TZcode source: internal
- Running under: Windows Server 2022 x64 (build 20348)
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: BAGS 2.48.0, Biobase 2.68.0, BiocGenerics 0.54.0, breastCancerVDX 1.45.0, generics 0.1.3
- Loaded via a namespace (and not attached): compiler 4.5.0, tools 4.5.0

References

- [1] Minn AJ, Gupta GP, Padua D, Bos P, Nguyen DX, Nuyten D, Kreike B, Zhang Y, Wang Y, Ishwaran H, Foekens JA, Van de Vijver M and Massagué J: Lung Metastasis Genes Couple Breast Tumor Size and Metastatic Spread. *PNAS*, **104(16)**, 6740-6745. 2007.
- [2] Quiroz-Zarate A and Quackenbush J XXXX: Genes as repeated measures of gene-set significance *Journal* **Vol(Num):Page 1-Page N**. 2011.
- [3] Wang Y, Klijn JGM, Zhang Y, Sieuwerts AM, Look MP, Yanh F, Talantov D, Timmermans M, Gelder, MEMG, Yu J, Jatkoe T, Berns EMJJ, Atkins D and Foekens JA: Gene-expression Profiles to Predict Distant Metastasis of Lymph-Node-Negative Primary Breast Cancer. *Lancet*, **365**, 671-679. 2005.