GeneGroupAnalysis: a package to perform gene-set significance analysis for data sets with cross-sectional or time series designs.

Alejandro Quiroz-Zárate[1] and John Quackenbush[1,2,3]

[1]Department of Biostatistics, Harvard University
[2]Computational Biology and Functional Genomics Laboratory, Dana-Farber Cancer Institute, Harvard School of Public Health
[3]Center for Cancer Computational Biology, Dana-Farber Cancer Institute

April 10, 2013

Contents

1 Introduction

The *GeneGroupAnalysis* package provides functions to perform statistical identification of gene functional classes that behave in a distinct manner between the phenotypes of interest for data sets 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.

1.1 Installation

GeneGroupAnalysis requires that *MCMCpack*, *AnnotationDbi*, *GO.db*, *annotate*, *tcltk* and *R* ($>=$ 2.10.0) are installed. These should be installed automatically when you install *GeneGroupAnalysis*. To install *GeneGroupAnalysis*, source biocLite from bioconductor:

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

Load the *GeneGroupAnalysis*, into your current workspace:

```
> library(GeneGroupAnalysis)
```

1.2 Further help

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

```
> library(help=GeneGroupAnalysis)
```

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 and Quackenbush J (2012). *Manuscript in preparation* .

2 Two simple cases: from gene expression to functional class selection.

We will very briefly demonstrate the use of some functions in *GeneGroupAnalysis* by providing its application on two data sets with different experimental design.

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. For purposes of providing an example on the use of the functions on the package the data set with cross-sectional design has a 70% of the samples with ER+ and with ER- where selected randomly selected.

For a demonstration on a data set with a time series design we use the *rheumaticConditionWOLLBOLD* data library from Bioconductor. This data set corresponds to the data used in [4]. Wollbold and colleagues used Affymetrix U133plus2 Gene Chips to profile expression of Synovial fibroblasts (SFB) cell cultures derived from six patients with rheumatic diseases, three with rheumatoid arthritis (RA) and three with osteoarthritis (OA). For each patient, SFB in culture were treated independently with TNFα and TGFβ1 and profiled at $0, 1, 2, 4$ and 12 hours after treatment; all analyses were performed in duplicate.

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(breastCancerVDX)
> library(GeneGroupAnalysis)
> library(hgu133a.db)
> library(annotate)
> data(vdx,package="breastCancerVDX")
> #-Normalized expression data set
> minn.data.expr=exprs(vdx)
> #--- Checking that the columns correspond to their respective phenotype data id
> #all(colnames(minn.data.expr)==rownames(pData(vdx)))
```

The data on `minn.data.expr` has the following appearance (maybe with a different column order):

	ER +			ER −		
	Sample 1	...	Sample 209	Sample 1	...	Sample 135
Probe ID 1	7.84	...	7.31	7.11	...	6.98
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Probe ID q	6.14	...	6.81	6.52	...	6.37

The rows correspond the the measurements for the Affymetrix identifiers and the columns correspond to the gene expression measurements on the samples.

The first function from the *GeneGroupAnalysis* package to be used is `ArrayInfoFun`. The function arguments are the normalized data set and the mapping between the probe identifiers and the gene symbols. The output of this functions is a list with four items containing the necessary information from the respective array platform in therms of gene symbols and probe identifiers.

```
> #---Checking the annotation of the data
> #annotation(vdx)
> array.info=ArrayInfoFun(minn.data.expr,hgu133aSYMBOL)
```

The vectors `er.pos` and `er.neg` contain the column identifiers of the samples with ER + and ER − status respectively. These vectors are obtained from the phenotype information provided along with the `vdx` eSet. Our next step is to obtain the probe identifiers with the largest variability between ER status across the samples for which there exist a gene symbol. By applying the `GeneMaxVarFun` we obtain a list the the row numbers of the probe identifiers that presented the largest variability and its associated Affymetrix identifier.

```
> #---ER status of patients
> er.status=pData(vdx)$er
> er.pos=which(er.status==1)
> er.neg=which(er.status==0)
> genes.max.var=GeneMaxVarFun(array.info,minn.data.expr,er.pos,er.neg)
```

Before moving on, we will define the amount of gene functional classes to considered for the analysis. At the moment the *GeneGroupAnalysis* package only considers Gene Ontology (GO) functional classes. For this example we will focus on its Cellular Component (CC) ontology up to a depth of 3 on the CC directed acyclic graph (DAG). For this we use the `CC.trees`. This object contains the GO processes and their associated gene symbols. This annotation is a transformation from the original MSigDB "c5.all.v3.0.symbols.gmt" provided by the Broad Institute. It contains only the information provided by the respective Gene Ontology Gene Sets with no additional information. The function `GeneGrps2AffyGrpsFun` makes the gene symbol to their associated Affymetrix identifier transformation:

```
> GO2Gene.grps=GeneGrps2AffyGrpsFun("CC",3,array.info$genes.name.unique,genes.max.var$IndexMaxVar)
```

Gene functional classes with low number of genes may obscure the potential inferences drawn from the proposed method. With the objective to keep only the meaningful functional classes, the function `SizeGOAffyGrps` selects only those functional classes with our desired size. For this example we chose the minimum group size of 10.

```
> Minn07WrkngGrps=SizeGOAffyGrps(GO2Gene.grps$index.GO.grps,100)
```

In order to implement the Gibb's sampler procedure the data set `minn.data.expr` needs to be transformed to a data set with functional class expression measurements. The `MCMCData.cs` performs this transformation.

```
> Minn07MCMCData=MCMCData.cs(Minn07WrkngGrps$groups,GO2Gene.grps$index.GO.grps,
+ GO2Gene.grps$GO.grps,minn.data.expr,er.pos,er.neg)
```

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):

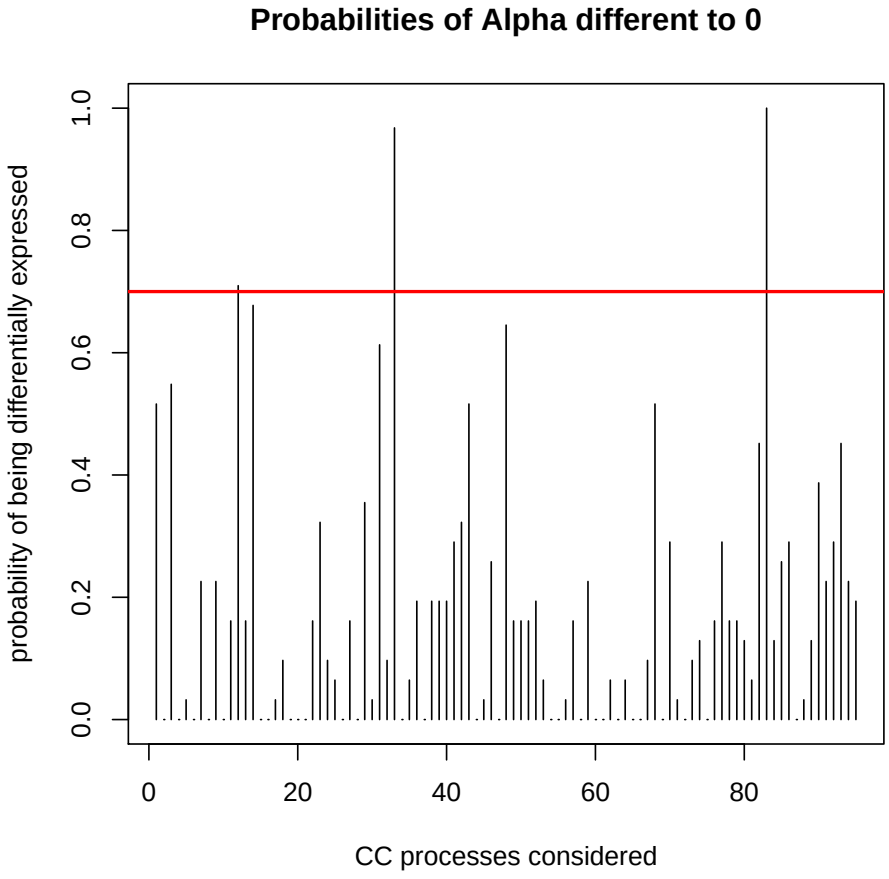
```
> nsim=40
> burn.in=10
> v.SS.A.i=matrix(0,length(Minn07MCMCData$proc.GO),nsim)
> v.SS.B.i=matrix(0,length(Minn07MCMCData$proc.GO),nsim)
> v.SS.Al=rep(0,nsim)
> v.beta.i=matrix(0,length(Minn07MCMCData$proc.GO),nsim)
> v.alfa.i=matrix(0,length(Minn07MCMCData$proc.GO),nsim)
> v.pi.a.i=matrix(0,length(Minn07MCMCData$proc.GO),nsim)
> v.rho.a=rep(0,nsim)
> #----- Initialization of the parameters
> v.beta.i[,1]=rnorm(length(Minn07MCMCData$proc.GO),0,1)
> v.alfa.i[,1]=rnorm(length(Minn07MCMCData$proc.GO),0,1)
> v.SS.Al[1]=0.1
> L0.alfa=0.001
> v.pi.a.i[,1]=runif(length(Minn07MCMCData$proc.GO))
> v.rho.a[1]=0.1
> Grps.apriori.diff.exp=23
> shape=3
> scale=0.3
> mm.pi=0.75
> aa.pi=10
> often=20
> cut.off=0.7
> result.MCMC=GeneGroupAnalysis::GibbsFun.cs(Minn07MCMCData$y.mu.a,Minn07MCMCData$y.mu.b,
+ Minn07WrkngGrps$group.size,v.beta.i,v.alfa.i,v.SS.A.i,v.SS.B.i,v.SS.Al,v.rho.a,v.pi.a.i,mm.pi,shape,
+ scale,aa.pi,Grps.apriori.diff.exp,nsim,burn.in,often,cut.off)

[1] 20
[1] 10
[1] 40
[1] 3
```

2.1.3 Visualizing which gene groups are differentially expressed between phenotypes.

To plot the probabilities of the functional classes of being differentially expressed we proposed the following:

```
> #------------------------------------------------
> #- Executable example after the application of GibbsFun.cs.
> #------------------------------------------------
> total=length(burn.in:nsim)
> prob.a=apply(result.MCMC$Alfa[,burn.in:nsim],1,function(x) length(which(x!=0))/total)
> no.groups=length(Minn07MCMCData$proc.GO)
> plot(1:no.groups,prob.a,type="h",main="Probabilities of Alpha different to 0",
+ xlab="CC processes considered",ylab="probability of being differentially expressed")
> abline(h=cut.off,lwd=2,col="red")
```



2.2 Example: Data analysis under a time series setting.

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

2.2.1 Data preprocessing stage

The original gene expression data set Wollbold J and colleagues [4] use has a U133plus2 Affymetrix platform. The normalized data set was saved to the variable `wollbold` in the *rheumaticConditionWOLLBOLD* data library from Bioconductor.

```
> library(GeneGroupAnalysis)
> library(annotate)
> library(rheumaticConditionWOLLBOLD)
> library(hgu133plus2.db)
> data(wollbold,package="rheumaticConditionWOLLBOLD")
> #----Normalized expression data set
> woll.data.exp=exprs(wollbold)
```

The data on `woll.data.exp` has the following appearance:

	TGF							TNF						
	0hrs			...	12hrs			0hrs			...	12hrs		
	Sbj 1	...	Sbj 6	...	Sbj 1	...	Sbj 6	Sbj 1	...	Sbj 6	...	Sbj 1	...	Sbj 6
Probe ID 1	7.84	...	7.31	...	7.11	...	6.98	7.05	...	6.73	...	6.45	...	6.89
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Probe ID q	5.23	...	5.09	...	8.93	...	8.34	6.12	...	6.32	...	9.05	...	8.89

The rows correspond the the measurements for the Affymetrix identifiers. The subjects between stimuli are different, but are the same across time, meaning that Sbj 1 at time 0 hrs under TGF is the same as Sbj 1 at time 12 hrs under TGF, but different from the one on the same times but under TNF.

The first function from the *GeneGroupAnalysis* package to be used is `ArrayInfoFun`. The function arguments are the normalized data set and the mapping between the probe identifiers and the gene symbols. The output of this functions is a list with four items containing the necessary information from the respective array platform in therms of gene symbols and probe identifiers.

```
> #---Checking the annotation of the data
> #annotation(wollbold)
> array.info=ArrayInfoFun(woll.data.exp,hgu133plus2SYMBOL)
```

The vectors `TGF.exp` and `TNF.exp` contain the column identifiers of the samples with TGF$\beta 1$ and TNFα stimulus respectively. These vectors are obtained from the phenotype

information provided along with the wollbold eSet. Our next step is to obtain the probe identifiers with the largest variability between stimuli status across the samples for which there exist a gene symbol. By applying the GeneMaxVarFun we obtain a list the the row numbers of the probe identifiers that presented the largest variability and its associated Affymetrix identifier.

```
> #---ER status of patients
> TGF.exp=1:30
> TNF.exp=31:60
> genes.max.var=GeneMaxVarFun(array.info,woll.data.exp,TGF.exp,TNF.exp)
```

Before moving on, we will define the amount of gene functional classes to considered for the analysis. At the moment the *GeneGroupAnalysis* package only considers Gene Ontology (GO) functional classes. For this example we will focus on its Molecular Function (MF) ontology up to a depth of 3 on the MF directed acyclic graph (DAG). For this we use the MF.trees. This object contains the GO processes and their associated gene symbols. This annotation is a transformation from the origial MSigDB "c5.all.v3.0.symbols.gmt" provided by the Broad Institute. It contains only the information provided by the respective Gene Ontology Gene Sets with no additional information. The function GeneGrps2AffyGrpsFun makes the gene symbol to their associated Affymetrix identifier transformation:

```
> GO2Gene.grps=GeneGrps2AffyGrpsFun("MF",3,array.info$genes.name.unique,genes.max.var$IndexMaxVar)
```

Gene functional classes with low number of genes may obscure the potential inferences drawn from the proposed method. With the objective to keep only the meaningful functional classes, the function SizeGOAffyGrps selects only those functional classes with our desired size. For this example we chose the minimum group size of 10.

```
> Wollbold09WrkngGrps=SizeGOAffyGrps(GO2Gene.grps$index.GO.grps,100)
```

In order to implement the Gibb's sampler procedure the data set woll.data.exp needs to be transformed to a data set with functional class expression measurements. The MCMCData.ts performs this transformation.

```
> indexes.1=c(1,2,3,4,5,6)
> Wollbold09MCMCData=MCMCData.ts(Wollbold09WrkngGrps$groups,GO2Gene.grps$index.GO.grps,
+ GO2Gene.grps$GO.grps,woll.data.exp,TGF.exp,TNF.exp,indexes.1,5)
```

The function MCMCData.ts differs from the MCMCData.cs function in the last two arguments. The first argument of these two is indexes.1. This is a vector with the order in which the subjects are recorded across the time course considered on woll.data.exp. In this case indexes.1$= c(1, 2, 3, 4, 5, 6)$. The next argument is the number of time points in the time series. In woll.data.exp the number of time points considered is 5: 0hrs, 1hrs, 2hrs, 4hrs and 12hrs.

2.2.2 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):

```
> nsim=40
> burn.in=10
> Grps.apriori.diff.exp=23
> shape=3
> scale=0.1
> mm.pi=0.75
> aa.pi=10
> often=20
> cut.off=0.72
> df.lambda=10
> no.time.pnts=5
> no.pcnts=6
> v.SS.i=array(0,c(length(Wollbold09MCMCData$proc.GO),no.time.pnts,no.time.pnts,2))
> v.SS.Al=array(0,c((no.time.pnts-1),(no.time.pnts-1),2))
> v.beta.i=array(0,c(length(Wollbold09MCMCData$proc.GO),no.time.pnts,nsim))
> v.alfa.0.i=matrix(0,length(Wollbold09MCMCData$proc.GO),nsim)
> v.alfa.i=array(0,c(length(Wollbold09MCMCData$proc.GO),(no.time.pnts-1),nsim))
> v.pi.a.i=matrix(0,length(Wollbold09MCMCData$proc.GO),nsim)
> v.rho.a=rep(0,nsim)
> v.beta.i[,,1]=rnorm(length(Wollbold09MCMCData$proc.GO),0,1)
> v.alfa.i[,,1]=rnorm(length(Wollbold09MCMCData$proc.GO),0,1)
> v.alfa.0.i[,1]=rnorm(length(Wollbold09MCMCData$proc.GO),0,1)
> v.SS.Al[,,1]=diag(0.1,(no.time.pnts-1))
> v.pi.a.i[,1]=runif(length(Wollbold09MCMCData$proc.GO))
> v.rho.a[1]=0.1
> L0.alfa=diag(0.005,(no.time.pnts-1))
> indexes.1=c(0,6,12,18,24)
> indexes.2=c(0,6,12,18)
> results.MCMC=GeneGroupAnalysis::GibbsNPFun.ts(Wollbold09MCMCData$y.mu.a,
+ Wollbold09MCMCData$y.mu.b,Wollbold09WrkngGrps$group.size,v.beta.i,v.alfa.0.i,
+ v.alfa.i,v.SS.Al,v.rho.a,v.pi.a.i,mm.pi,aa.pi,Wollbold09MCMCData$lambda,df.lambda,L0.alfa,
+ indexes.1,indexes.2,no.time.pnts,no.pcnts,Grps.apriori.diff.exp,nsim,burn.in,often,cut.off)

[1] 20
[1] 28
[1] 40
[1] 21
```

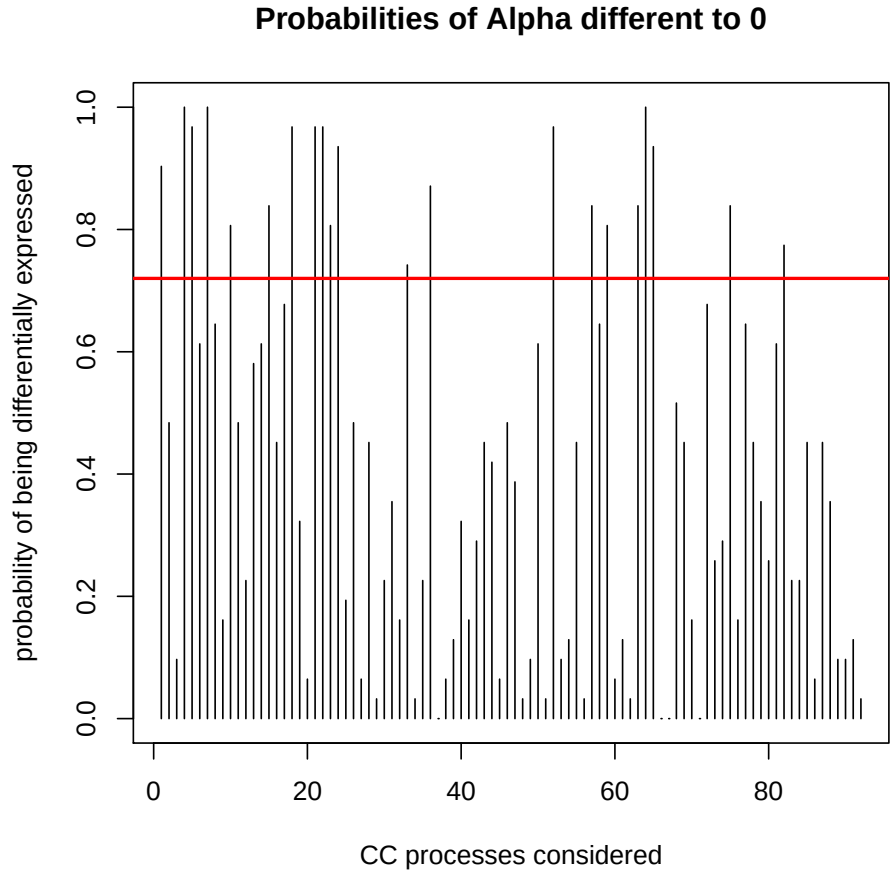
In `GibbsNPFun.ts` the vector `indexes.1` has the indices that indicate the column number on `woll.data.expr` where a new time point period starts. Vector `indexes.2` does the

same for the parameter Alpha (see [2]). For both vectors first column number starts in 0 ONLY for this function.

2.2.3 Visualizing which gene groups are differentially expressed between phenotypes.

To plot the probabilities of the functional classes of being differentially expressed we proposed the following:

```
> #-----------------------------------------------------
> #- Executable example after the application of GibbsNPFun.ts
> #-----------------------------------------------------
> total=length(burn.in:nsim)
> prob.a=apply(results.MCMC$Alfa[,1,burn.in:nsim],1,function(x) length(which(x!=0))/total)
> no.groups=length(Wollbold09MCMCData$proc.GO)
> plot(1:no.groups,prob.a,type="h",main="Probabilities of Alpha different to 0",
+ xlab="CC processes considered",ylab="probability of being differentially expressed")
> abline(h=cut.off,lwd=2,col="red")
```



2.2.4 Obtaining of the functional classes differentially expressed between phenotypes.

```
> #-------------------------------------------------
> #- Executable example after the application of GibbsNPFun.ts
> #-------------------------------------------------
> library(annotate)
> library(GO.db)
> selection=which(prob.a>=cut.off)
> selected.GO.groups=GO2Gene.grps$GO.grps[Wollbold09WrkngGrps$groups][selection]
> print(selected.GO.groups)

 [1] "GO:0009055" "GO:0003824" "GO:0004872" "GO:0005215" "GO:0060089"
 [6] "GO:0016787" "GO:0016874" "GO:0022857" "GO:0022892" "GO:0000166"
[11] "GO:0005515" "GO:0043167" "GO:0048037" "GO:0016788" "GO:0016879"
[16] "GO:0004888" "GO:0022891" "GO:0017076" "GO:0032553" "GO:0019899"
[21] "GO:0003723"

> funct.classes=getGOTerm(selected.GO.groups)[[1]]
> print(funct.classes)

                                        GO:0009055
                          "electron carrier activity"
                                        GO:0003824
                                "catalytic activity"
                                        GO:0004872
                                 "receptor activity"
                                        GO:0005215
                              "transporter activity"
                                        GO:0060089
                        "molecular transducer activity"
                                        GO:0016787
                                "hydrolase activity"
                                        GO:0016874
                                   "ligase activity"
                                        GO:0022857
                    "transmembrane transporter activity"
                                        GO:0022892
                "substrate-specific transporter activity"
                                        GO:0000166
                                "nucleotide binding"
                                        GO:0005515
                                   "protein binding"
```

```
                                                GO:0043167
                                              "ion binding"
                                                GO:0048037
                                           "cofactor binding"
                                                GO:0016788
                "hydrolase activity, acting on ester bonds"
                                                GO:0016879
          "ligase activity, forming carbon-nitrogen bonds"
                                                GO:0004888
               "transmembrane signaling receptor activity"
                                                GO:0022891
"substrate-specific transmembrane transporter activity"
                                                GO:0017076
                                "purine nucleotide binding"
                                                GO:0032553
                              "ribonucleotide binding"
                                                GO:0019899
                                      "enzyme binding"
                                                GO:0003723
                                         "RNA binding"
```

3 Session Info

- R version 3.0.0 (2013-04-03), `x86_64-apple-darwin10.8.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: AnnotationDbi 1.22.1, Biobase 2.20.0, BiocGenerics 0.6.0, DBI 0.2-5, GO.db 2.9.0, GeneGroupAnalysis 1.6.0, MASS 7.3-26, MCMCpack 1.2-4.1, RSQLite 0.11.2, annotate 1.38.0, breastCancerVDX 1.0.5, coda 0.16-1, hgu133a.db 2.9.0, hgu133plus2.db 2.9.0, lattice 0.20-15, org.Hs.eg.db 2.9.0, rheumaticConditionWOLLBOLD 1.0.3
- Loaded via a namespace (and not attached): IRanges 1.18.0, XML 3.96-1.1, grid 3.0.0, stats4 3.0.0, tcltk 3.0.0, tools 3.0.0, xtable 1.7-1

4 Functions and Data sets within GeneGroupAnalysis

For references to the following functions, please see [2].

Function	Description
ArrayInfoFun	Function that extracts the required information from the data set
GOGrps2GeneGrpsFun	Function that translates GO functional classes into their respective gene symbol groups and into data matrix row number groups
GOProcsFun	Function that computes the GO functional classes to a desired depth of the GO DAG for the ontology of interest
GeneGroupAnalysis-package	Perform Gene Functional Class Analysis
GeneMaxVarFun	Function to compute the Affymetrix identifiers with the maximum variability between phenotypes
GibbsAllFun.ts	Function to compute the Gibb's sampler for the series data design
GibbsFun.cs	Function to compute the Gibb's sampler for the cross-sectional data design
GibbsNPFun.ts	Function to compute the Gibb's sampler for the series data design
MCMCData.cs	Function that computes the requiered data set objects for the Gibb's sampler under a cross-sectional data design
MCMCData.ts	Function that computes the requiered data set objects for the Gibb's sampler under a time series data design
MeanMatGrps.Aux.ts	Auxiliary function to 'MCMCData.ts'
SizeGOAffyGrps	Function to compute gene functional classes with a desired size

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.

[4] Wollbold J, Huber R, Pohlers D, Koczan D, Guthke R, Kinne RW and Gausmann U: Adapted Boolean Network Models for Extracellular Matrix Formation. *BMC Systems Biology*, **3(77)**, 2009.