Enrichment Analysis & Dimensionality Reduction

Pablo Riesgo

2016-09-28

Using Object Oriented (OO) programming

Using OO allows achieving **high cohesion** and **low coupling**, which facilitates all the software development cycle from development to maintenance, including testing. Following the principle **"Don't repeat yourself"** (https://en.wikipedia.org/wiki/Don't_repeat_yourself) is also enhanced with a good OO design. There are 3 OO models in R, we will be using S4 as it is the model most used in Bioconductor. Furthermore, we want to foster **code reuse** from existing Bioconductor S4 classes.

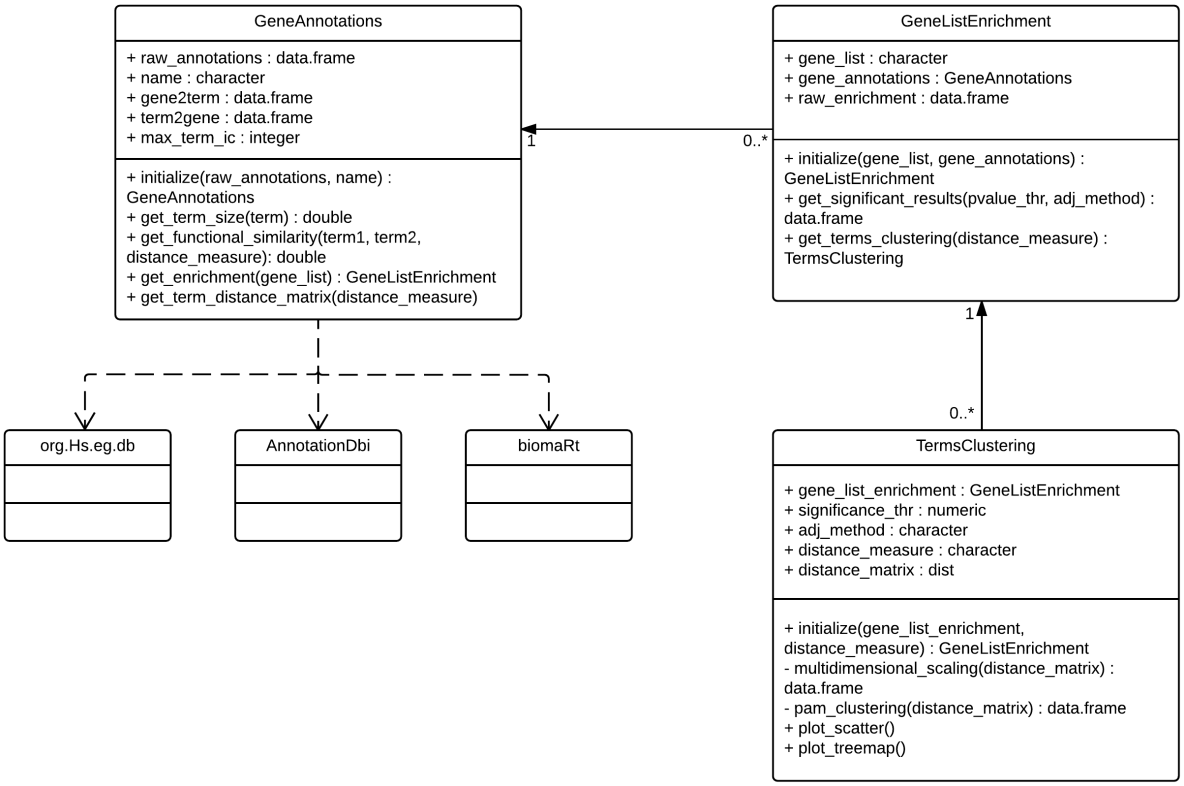


Figure 1: Enrichment class model

Annotations

This model allows us to support multiple annotations for enrichment. So far the supported annotations based on AnnotationDbi data source and some custom resources are:

- Gene Ontology: biological process, molecular function and cell component (from org.Hs.eg.db)
- KEGG pathways (from org.Hs.eg.db)
- OMIM diseases (from org.Hs.eg.db)

- HPO phenotypes (from http://compbio.charite.de/jenkins/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_TYPICAL_FEATURES_phenotype_to_genes.txt)

These annotations can be loaded as follows:

```
goa_bp <- TCGAome::load_goa(ontology = "BP")
#> INFO [2016-09-28 11:43:45] Loading human GOA...
#> INFO [2016-09-28 11:44:26] Loaded 14291 GO terms and 16536 genes
goa_mf <- TCGAome::load_goa(ontology = "MF")
#> INFO [2016-09-28 11:44:26] Loading human GOA...
#> INFO [2016-09-28 11:44:49] Loaded 4263 GO terms and 16619 genes
goa_cc <- TCGAome::load_goa(ontology = "CC")
#> INFO [2016-09-28 11:44:49] Loading human GOA...
#> INFO [2016-09-28 11:45:14] Loaded 1692 GO terms and 17761 genes
kegg <- TCGAome::load_kegg()
#> INFO [2016-09-28 11:45:14] Loading human KEGG...
#> INFO [2016-09-28 11:45:15] Loaded 229 KEGG pathways and 5869 genes
omim <- TCGAome::load_omim()
#> INFO [2016-09-28 11:45:15] Loading human OMIM...
#> INFO [2016-09-28 11:45:16] Loaded 19459 OMIM diseases and 15042 genes
hpo <- TCGAome::load_hpo()
#> INFO [2016-09-28 11:45:16] Loading HPO...
#> INFO [2016-09-28 11:45:44] Loaded 6436 HPO phenotypes and 3520 genes
```

The S4 object created contains the following slots:

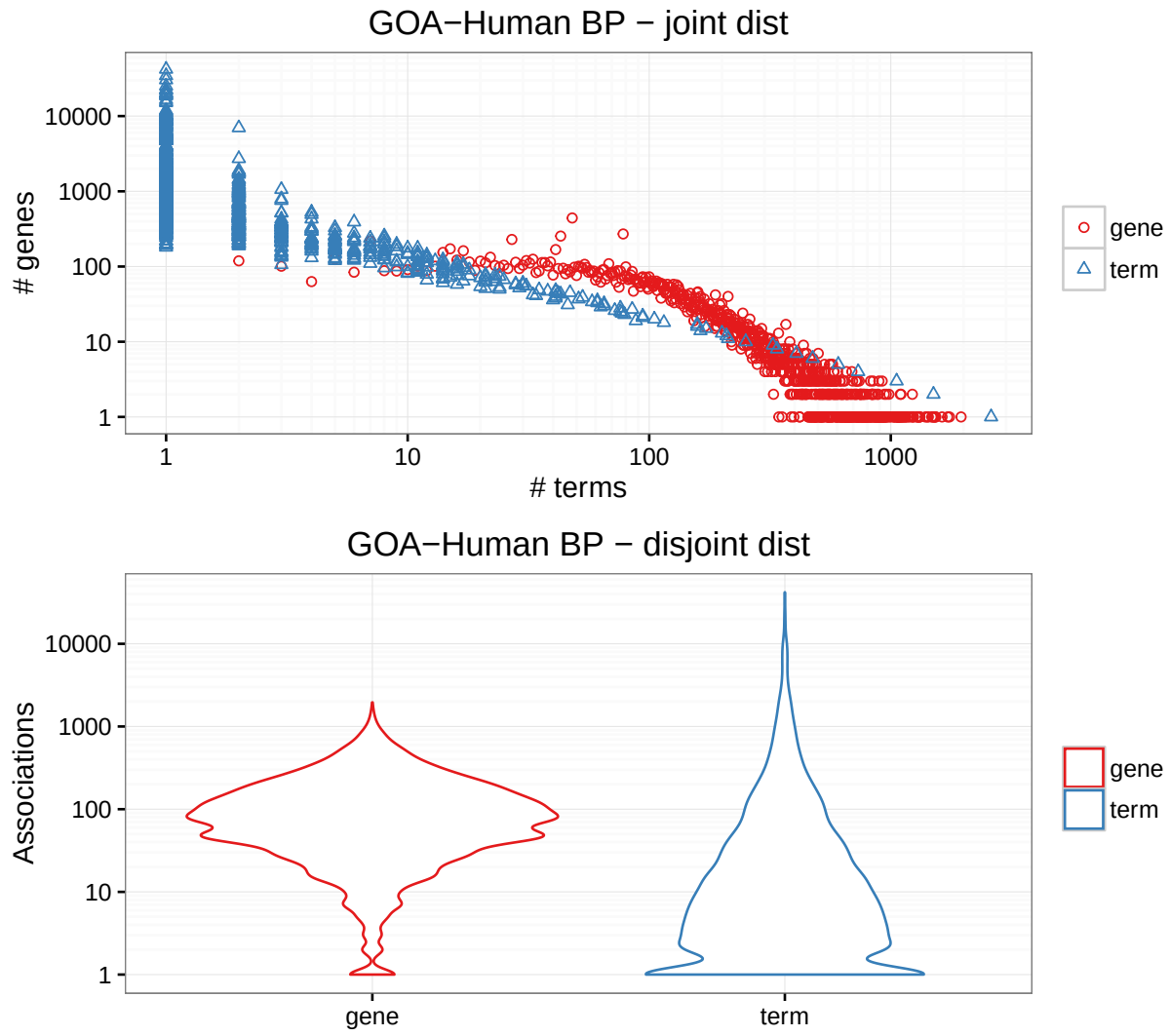
```
str(goa_bp, list.len = 5, vec.len = 5)
#> Formal class 'GeneAnnotations' [package "TCGAome"] with 6 slots
#>   ..@ raw_annotations        :'data.frame':  1988963 obs. of  2 variables:
#>   .. ..$ Gene: chr [1:1988963] "A1BG" "A2M" "A2M" "A2M" "A2M" ...
#>   .. ..$ Term: chr [1:1988963] "GO:0008150" "GO:0001775" "GO:0001867" "GO:0001868" "GO:0001869" ...
#>   ..@ name                   : chr "GOA-Human BP"
#>   ..@ gene2term              :'data.frame':  16536 obs. of  2 variables:
#>   .. ..$ Gene: chr [1:16536] "A1BG" "A1CF" "A2M" "A2ML1" "A4GALT" ...
#>   .. ..$ Term:List of 16536
#>   .. .. ..$ 00001: chr "GO:0008150"
#>   .. .. ..$ 00002: chr [1:48] "GO:0006139" "GO:0006139" "GO:0006396" "GO:0006397" "GO:0006725" ...
#>   .. .. ..$ 00003: chr [1:201] "GO:0001775" "GO:0001867" "GO:0001868" "GO:0001869" "GO:0002252" ...
#>   .. .. ..$ 00004: chr [1:58] "GO:0006508" "GO:0006508" "GO:0008150" "GO:0008150" "GO:0008152" ...
#>   .. .. ..$ 00005: chr [1:88] "GO:0001575" "GO:0001576" "GO:0005975" "GO:0006464" "GO:0006486" ...
#>   .. .. .. [list output truncated]
#>   ..@ term2gene              :'data.frame':  14291 obs. of  2 variables:
#>   .. ..$ Term: chr [1:14291] "GO:0000002" "GO:0000003" "GO:0000011" "GO:0000012" "GO:0000018" ...
#>   .. ..$ Gene:List of 14291
#>   .. .. ..$ 00001: chr [1:31] "PARP1" "SLC25A4" "DNA2" "TYMP" "LIG3" ...
#>   .. .. ..$ 00002: chr [1:1096] "ABAT" "ACR" "ACR" "ACR" "ACR" ...
#>   .. .. ..$ 00003: chr "RBSN"
#>   .. .. ..$ 00004: chr [1:11] "PARP1" "LIG4" "TERF2" "TNP1" "XRCC1" ...
#>   .. .. ..$ 00005: chr [1:58] "BCL6" "BLM" "CD28" "CD40" "CHEK1" ...
#>   .. .. .. [list output truncated]
#>   ..@ max_term_annotations   : int 41987
#>   .. [list output truncated]
```

TODO: add a print function to annotations object

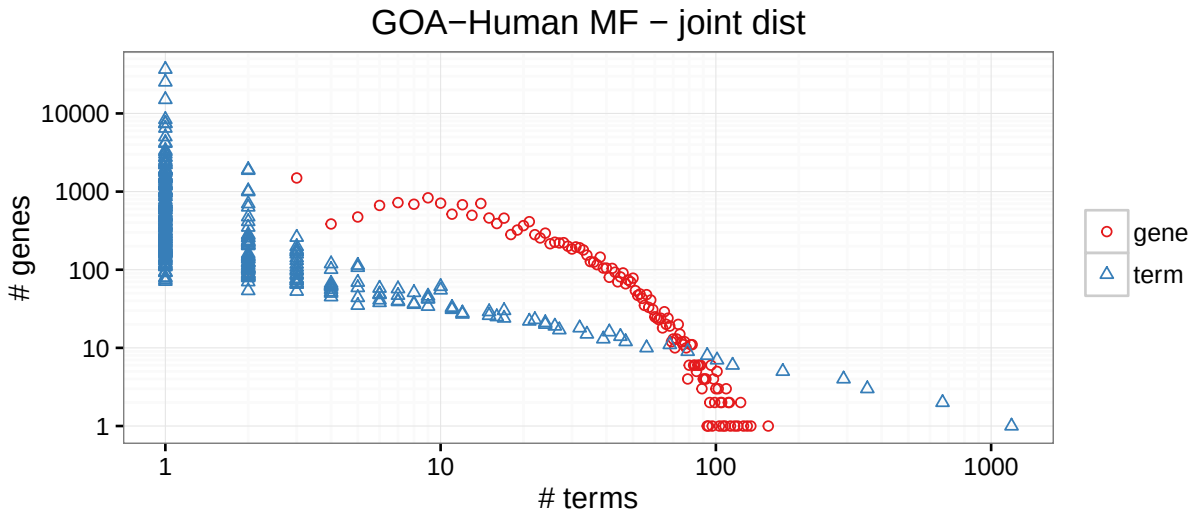
Distribution of annotations

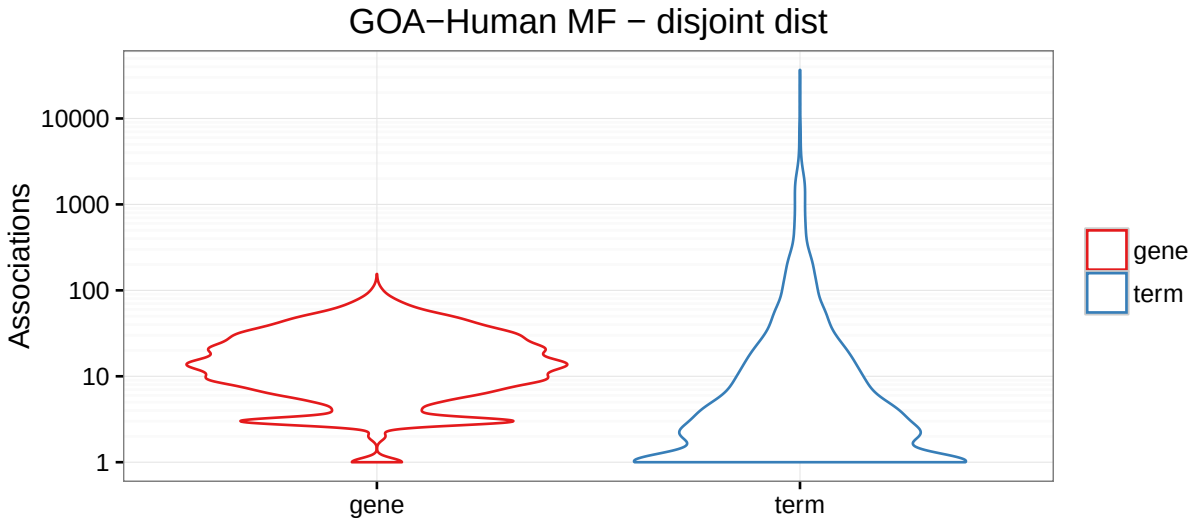
We can visually explore the distribution of annotations and explore the differences between genes and terms for the different resources.

```
plot(goa_bp)
```

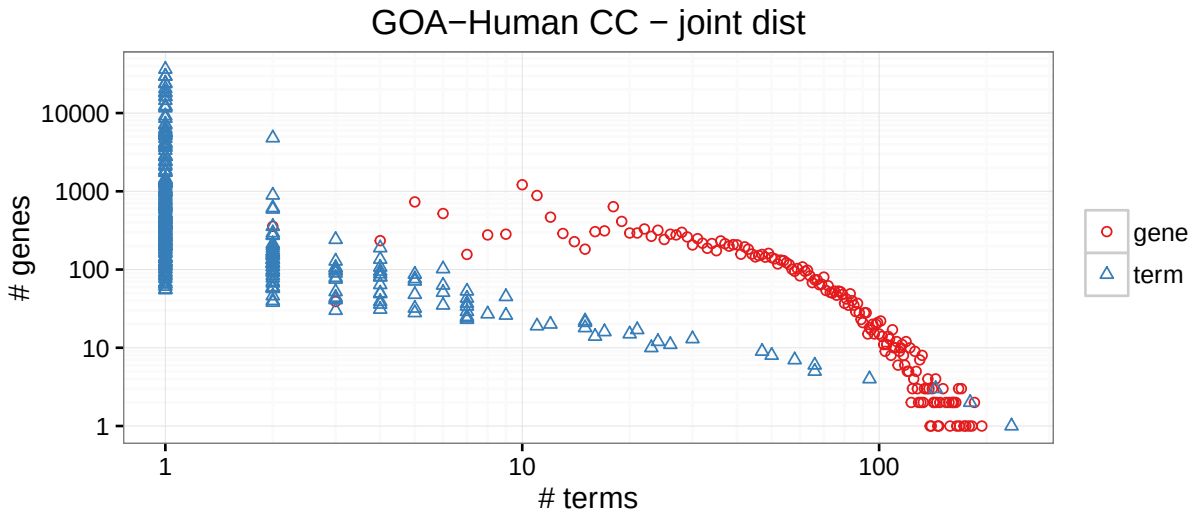


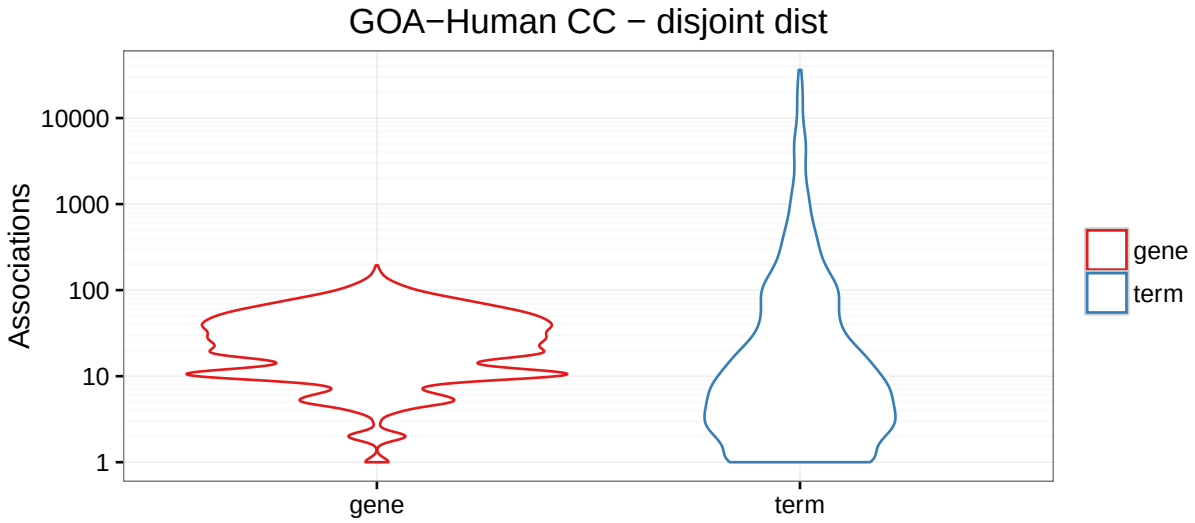

```
plot(goa_mf)
```


GOA−Human MF − joint dist
genes
10000
1000
100
10
1
1
10
100
1000
terms
gene
term

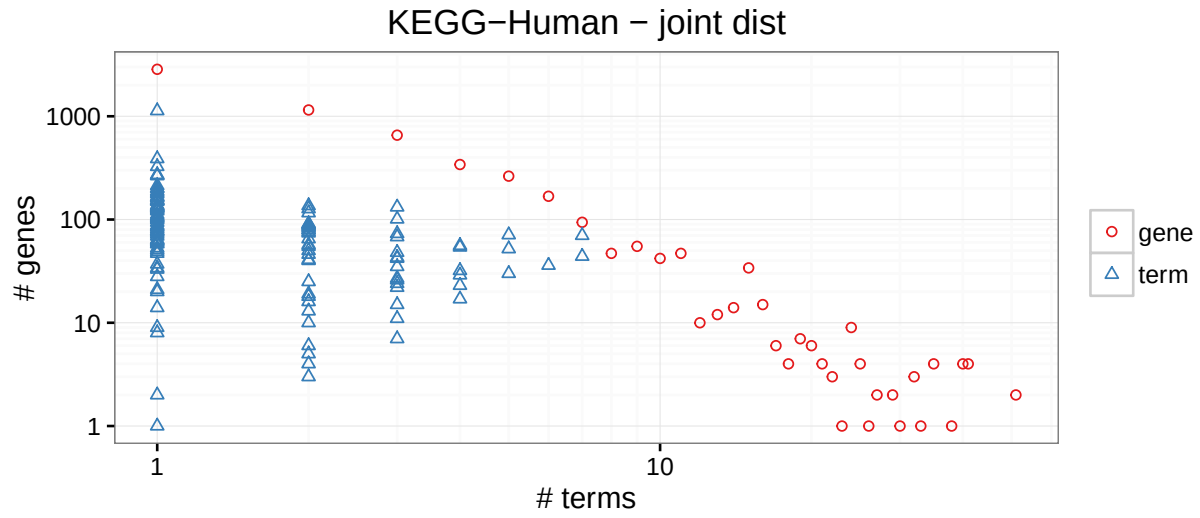

GOA−Human MF − disjoint dist
Associations
10000
1000
100
10
1
gene
term
gene
term

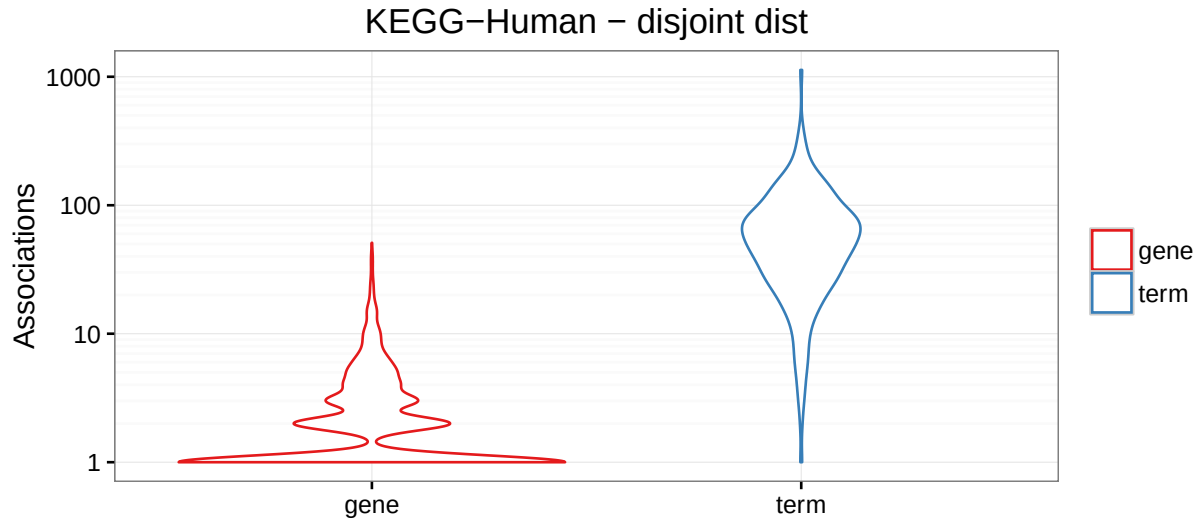
```
plot(goa_cc)
```


GOA−Human CC − joint dist
genes
10000
1000
100
10
1
1
10
100
terms
gene
term

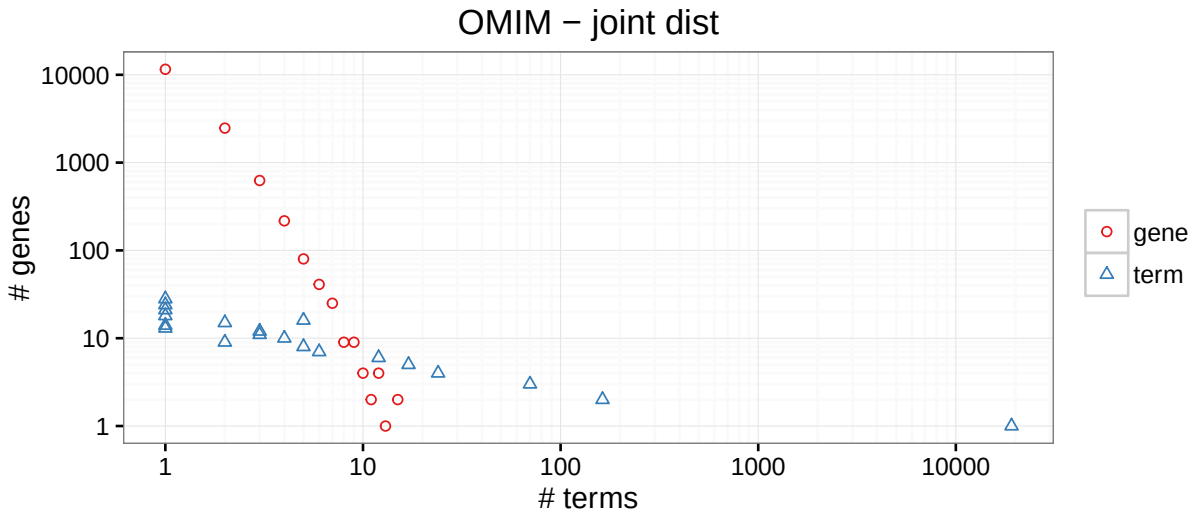

GOA−Human CC − disjoint dist
Associations
10000
1000
100
10
1
gene
term
gene
term

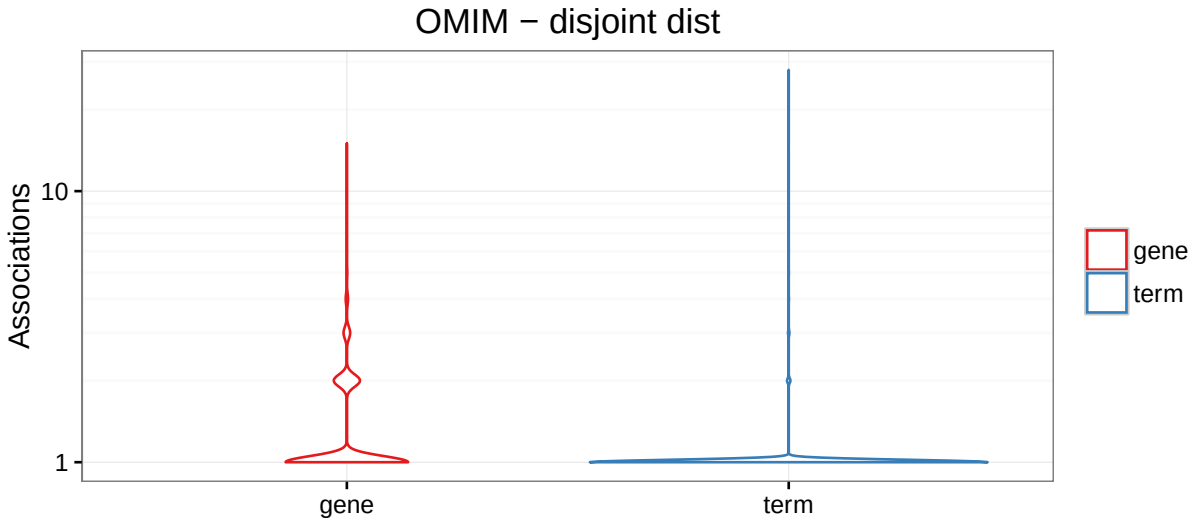
```
plot(kegg)
```


KEGG−Human − joint dist
genes
1000
100
10
1
1
10
terms
gene
term

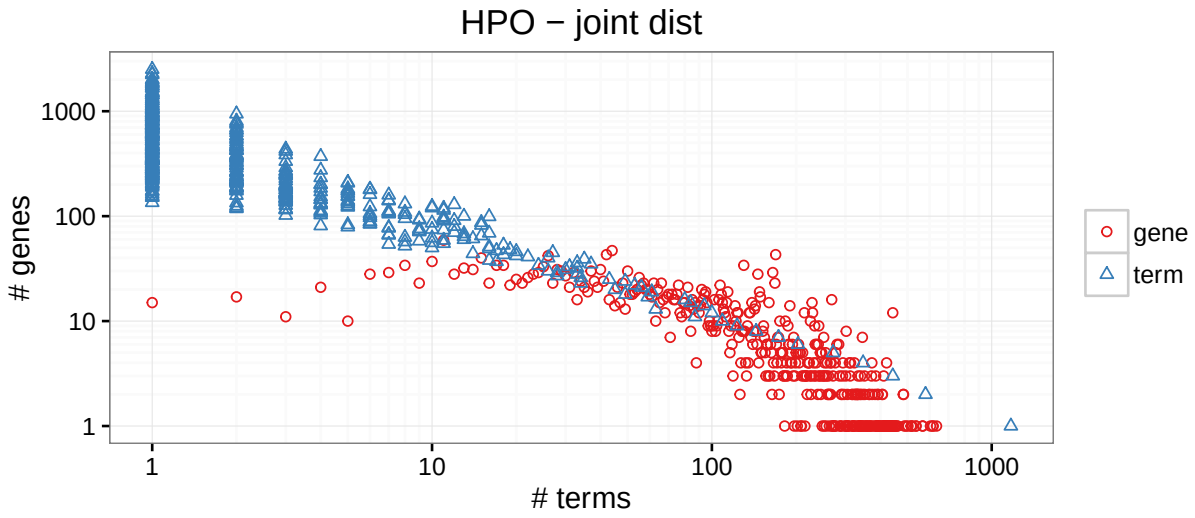

KEGG−Human − disjoint dist
Associations
1000
100
10
1
gene
term
gene
term

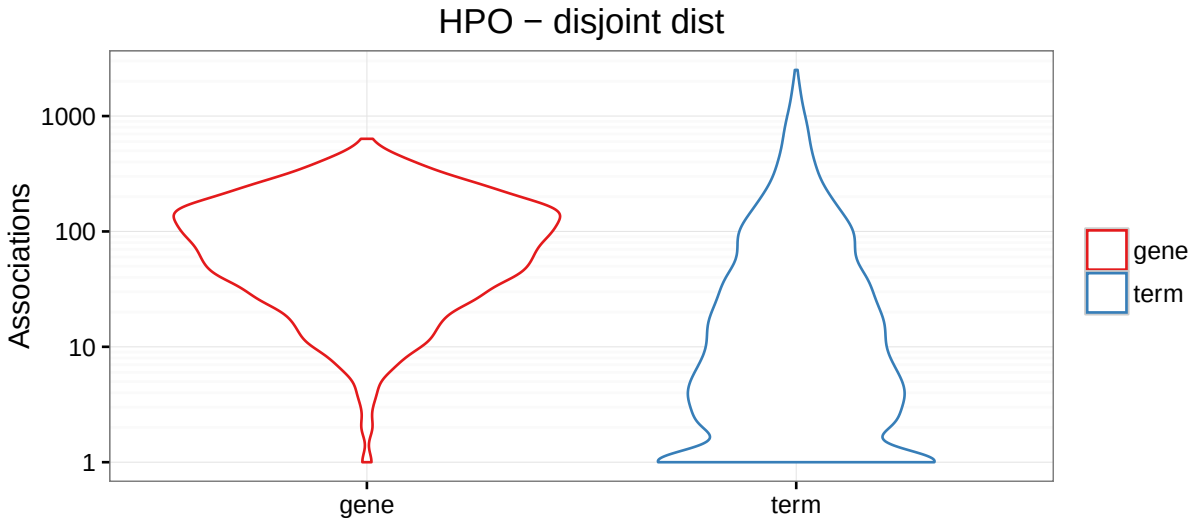
```
plot(omim)
```


OMIM − joint dist
10000
1000
100
10
1
genes
1
10
100
1000
10000
terms
gene
term


OMIM − disjoint dist
Associations
10
1
gene
term
gene
term

```
plot(hpo)
```





Metrics

For any term within the **TCGAome::GeneAnnotations** object we can obtain its **relative frequency** of annotation, which might be useful to compute the **Information Content** within a term.

```
random_term = goa_bp@term2gene$Term[runif(1, max = length(goa_bp@term2gene$Term))]
random_term
#> [1] "GO:0036260"
TCGAome::get_term_freq(goa_bp, random_term)
#> [1] 0.00107176
```

For any two terms within the GeneAnnotations object we can obtain its **functional similarity** based on the binary distances implemented in TCGAome.

```
random_term1 = goa_bp@term2gene$Term[runif(1, max = length(goa_bp@term2gene$Term))]
random_term2 = goa_bp@term2gene$Term[runif(1, max = length(goa_bp@term2gene$Term))]
random_term1
#> [1] "GO:0014037"
```

```
random_term2
#> [1] "GO:1904231"
TCGAome::get_functional_similarity(goa_bp, random_term1, random_term2, distance_measure = "cosine")
#> [1] 0
```

The supported binary distances are: UI, cosine, Bray-Curtis and binary.

This model should be extended to support **semantic similarity** measures based on the ontology structure. This is a special case for some of these annotation resources which are also backed by an ontology, like GO and HPO. On the previous implementation of TCGAome we used **GoSemSim** for the semantic similarity within GO terms. We need to study if this can be easily extended to other ontologies.

Extend annotation support

It is relatively simple to extend the support to additional annotations, we just need to provide the association between genes and terms in a tall-skinny data frame with columns "Gene" and "Term".

```
uniKeys <- AnnotationDbi::keys(org.Hs.eg.db::org.Hs.eg.db, keytype="SYMBOL")
cols <- c("PATH")
kegg_raw <- AnnotationDbi::select(org.Hs.eg.db::org.Hs.eg.db, keys=uniKeys, columns=cols, keytype="SYMB
kegg_raw <- kegg_raw[, c(1, 2)]
colnames(kegg_raw) <- c("Gene", "Term")
kegg <- TCGAome::GeneAnnotations(raw_annotations = kegg_raw, name="KEGG-Human")
```

Enrichment

The previous TCGAome version used the package **topGO** for computing the enrichment of GO terms. This package is limited to GO. The computation employed was a Fisher's test, we were not making use of the advanced functionality in topGO. Thus, in order to gain flexibility the enrichment computation was reimplemented inside the class **TCGAome::GeneListEnrichment**.

We can compute enrichment for a given list of genes based on a preloaded annotation:

```
gene_list <- c("ZNF638", "HNRNPU", "PPIAL4G", "RAPH1", "USP7", "SUMO1P3",
          "TMEM189.UBE2V1", "ZNF837", "LPCAT4", "ZFPL1", "STAT3", "XRCC1",
          "STMN1", "PGR", "RB1", "KDR", "YBX1", "YAP1", "FOXO3", "SYK", "RAB17",
          "TTC8", "SLC22A5", "C3orf18", "ANKRA2", "LBR", "B3GNT5", "ANP32E",
          "JOSD1", "ZNF695", "ESR1", "INPP4B", "PDK1", "TSC2", "AR", "HSPA1A",
          "CDH3", "SMAD4", "CASP7", "GMPS", "NDC80", "EZH2", "MELK", "CDC45",
          "CRY2", "KLHDC1", "MEIS3P1", "FBXL5", "EHD2", "CCNB1", "GSK3A",
          "DVL3", "NFKB1", "COL6A1", "CCND1", "BAK1")
goa_bp_enrichment <- TCGAome::get_enrichment(goa_bp, gene_list = gene_list)
#> INFO [2016-09-28 11:45:48] Initializing GeneListEnrichment...
#> INFO [2016-09-28 11:45:48] Input gene list is of length : 56
#> INFO [2016-09-28 11:45:48] Checking object's validity...
#> INFO [2016-09-28 11:45:48] Object is valid!
#> INFO [2016-09-28 11:45:48] After removing missing values input gene list is of length : 56
#> INFO [2016-09-28 11:45:48] Computing Fisher test enrichment on the gene list...
#> INFO [2016-09-28 11:46:08] Most significant p-value 7.98211266671077e-06 and less significant 1
str(goa_bp_enrichment, list.len = 5, vec.len = 5)
#> Formal class 'GeneListEnrichment' [package "TCGAome"] with 2 slots
#>   ..@ gene_list     : atomic [1:56] ZNF638 HNRNPU PPIAL4G RAPH1 USP7 ...
```

```
#>   .. ..- attr(*, ".match.hash")=Class 'match.hash' <externalptr>
#>   ..@ raw_enrichment:'data.frame':   14291 obs. of  2 variables:
#>   .. ..$ Term  : chr [1:14291] "GO:0000002" "GO:0000003" "GO:0000011" "GO:0000012" "GO:0000018" ...
#>   .. ..$ pvalue: num [1:14291] 1 0.1604 1 0.0366 1 1 ...
```

Or alternatively:

```
goa_bp_enrichment <- TCGAome::GeneListEnrichment(gene_annotations = goa_bp, gene_list = gene_list)
```

And extract significant results:

```
head(TCGAome::get_significant_results(goa_bp_enrichment, significance_thr = 0.05, adj_method = "none"))
#>          Term      pvalue  adj_pvalue
#> 4  GO:0000012 0.036627643 0.036627643
#> 25 GO:0000070 0.010717793 0.010717793
#> 26 GO:0000075 0.013881082 0.013881082
#> 27 GO:0000076 0.046388923 0.046388923
#> 30 GO:0000082 0.001953379 0.001953379
#> 31 GO:0000083 0.003742984 0.003742984
```

TODO: visualize enrichment results

TODO: add a print function to enrichment object

Visualizing enrichment results (clustering + MDS)

To visualize enriched terms we use a dimensionality reduction approach based on clustering our results, select then a representative member of each cluster and plot them into 2D by using multidimensional scaling.

Tu run this pipeline we need to create an object of type "TermsClustering" as follows:

```
goa_bp_term_clustering <- TCGAome::TermsClustering(goa_bp, goa_bp_enrichment, distance_measure = "cosin
#> INFO [2016-09-28 11:46:08] Initializing TermsClustering...
#> INFO [2016-09-28 11:46:08] Distance metric : cosine
#> INFO [2016-09-28 11:46:08] Significance threshold : 0.05
#> INFO [2016-09-28 11:46:08] Getting significantly enriched terms...
#> INFO [2016-09-28 11:46:08] # of significant terms : 897
#> INFO [2016-09-28 11:46:08] Getting distance matrix for the signifcant terms...
#> INFO [2016-09-28 11:49:01] Got it!
#> INFO [2016-09-28 11:49:01] Checking object's validity...
#> INFO [2016-09-28 11:49:01] Object is valid!
#> INFO [2016-09-28 11:49:01] Clustering significant terms with max clusters = 10 ...
#> INFO [2016-09-28 11:49:02] Found 10 clusters
#> INFO [2016-09-28 11:49:02] Computing MDS on the significant terms...
#> INFO [2016-09-28 11:49:03] MDS done
#> INFO [2016-09-28 11:49:03] Computing the annotation frequency of significant terms...
#> INFO [2016-09-28 11:49:03] Done
#> INFO [2016-09-28 11:49:03] Select representative terms for clusters...
#> INFO [2016-09-28 11:49:03] Done
#> INFO [2016-09-28 11:49:03] Compute MDS again on cluster representatives...
#> INFO [2016-09-28 11:49:03] Done
str(goa_bp_term_clustering@distance_matrix, list.len = 5, vec.len = 5)
```

```
#> Class 'dist'  atomic [1:401856] 1 1 1 1 1 1 ...
#>   ..- attr(*, "Labels")= chr [1:897] "GO:0000012" "GO:0000070" "GO:0000075" "GO:0000076" "GO:0000082
#>   ..- attr(*, "Size")= int 897
#>   ..- attr(*, "call")= language as.dist.default(m = distance_matrix)
#>   ..- attr(*, "Diag")= logi FALSE
#>   ..- attr(*, "Upper")= logi FALSE
str(goa_bp_term_clustering@significant_results, list.len = 5, vec.len = 5)
#> 'data.frame':    897 obs. of  12 variables:
#>  $ Term      : chr  "GO:0000012" "GO:0000070" "GO:0000075" "GO:0000076" "GO:0000082" ...
#>  $ Cluster   : int  1 2 2 2 2 2 2 3 4 2 2 2 ...
#>  $ pvalue    : num  0.03663 0.01072 0.01388 0.04639 0.00195 0.00374 ...
#>  $ adj_pvalue: num  0.03663 0.01072 0.01388 0.04639 0.00195 0.00374 ...
#>  $ pc1       : num  -0.15455 -0.16853 -0.02928 -0.17788 0.00833 -0.14575 ...
#>   [list output truncated]
```

Or alternatively:

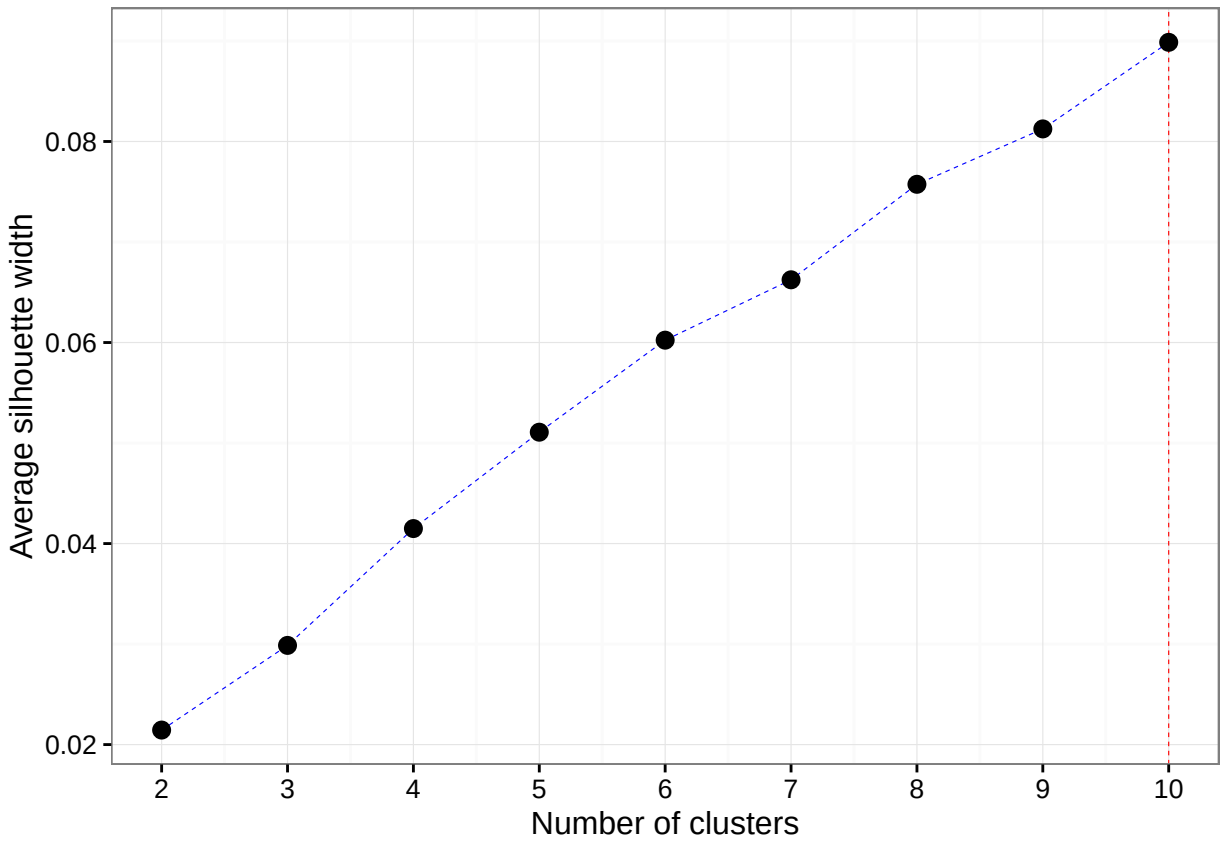
```
goa_bp_term_clustering <- TCGAome::get_terms_clustering(goa_bp_enrichment, goa_bp, distance_measure = "
```

The clustering is performed based on a similarity metric between terms. The supported similarity metrics are based on the genes annotated to each term. As genes are functional elements we called it functional similarity. We support so far several metrics based on binary vector distances, every term is associated to a binary vector with the length of the total number of genes that our term annotations contain. Each position in the vector contains 1 or 0 if the given term is annotated with that gene or not. So far, we support a number of binary distances: Union-Intersection (UI), cosine, Bray-Curtis and binary.

To decide the optimal number of clusters we perform a silhouette analysis. We use by default a maximum of clusters of 10 as otherwise results become quickly complex, this can be changed using parameter *max_clusters*.

Evaluate the optimal number of clusters with silhouette analysis:

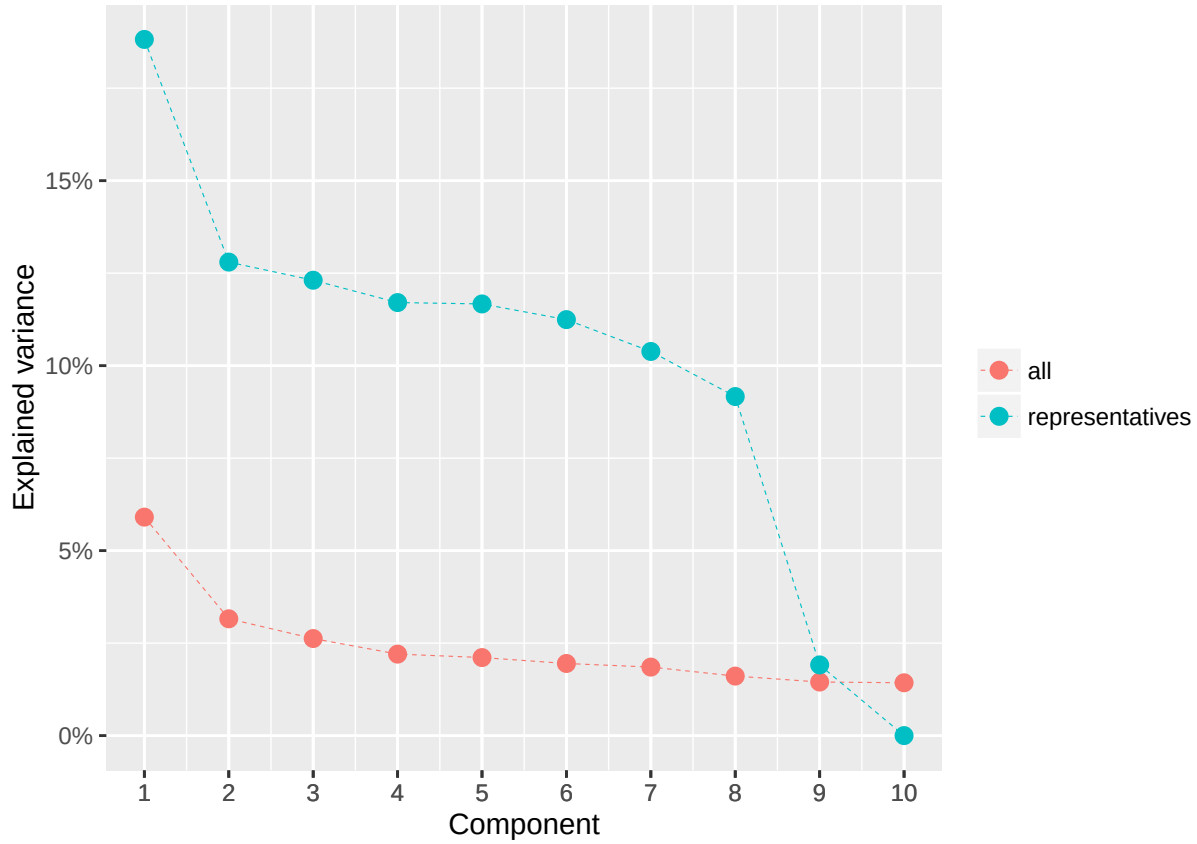
```
TCGAome::plot_silhouette_analysis(goa_bp_term_clustering)
```



We perform a PAM clustering with the obtained optimal number of clusters and select a term representative for each of the clusters. The criteria to select the term representative is as follows: for each cluster if there is any term with a frequency of annotation below a given threshold select the term with the lowest enrichment p-value; otherwise select the term with the lowest p-value across all terms independently of their frequency of annotation.

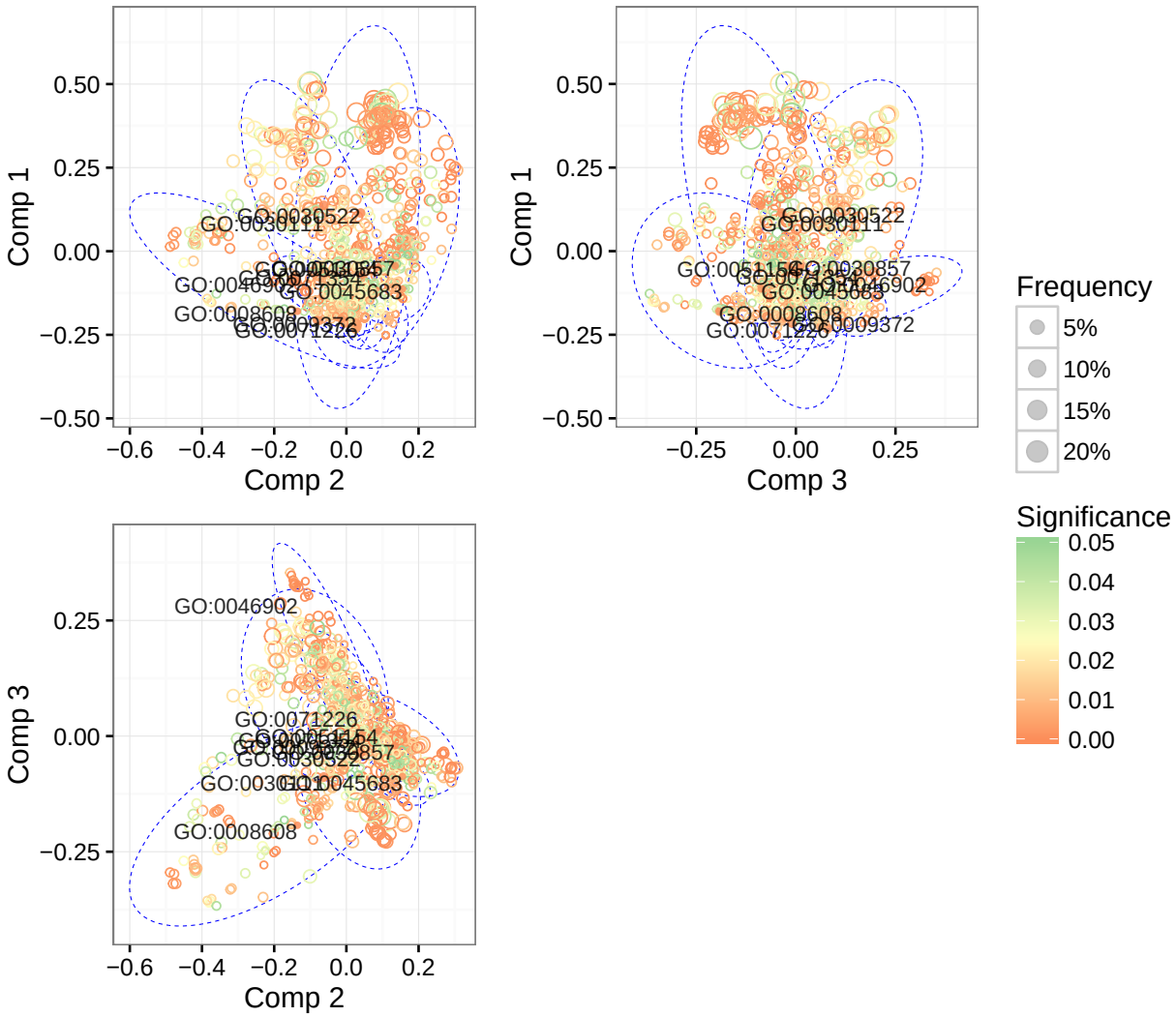
Also based on the similarity metrics between terms we can perform a Multidimensional Scaling to plot our clusters in a reduced number of dimensions. We have two alternatives: perform the MDS on every significant term that was clustered or perform it just on the subset of cluster representatives. To evaluate the MDS we need to inspect the explained variance by the first 3 components, which are the ones we will be visualizing.

```
TCGAome::plot_explained_variance(goa_bp_term_clustering)
```



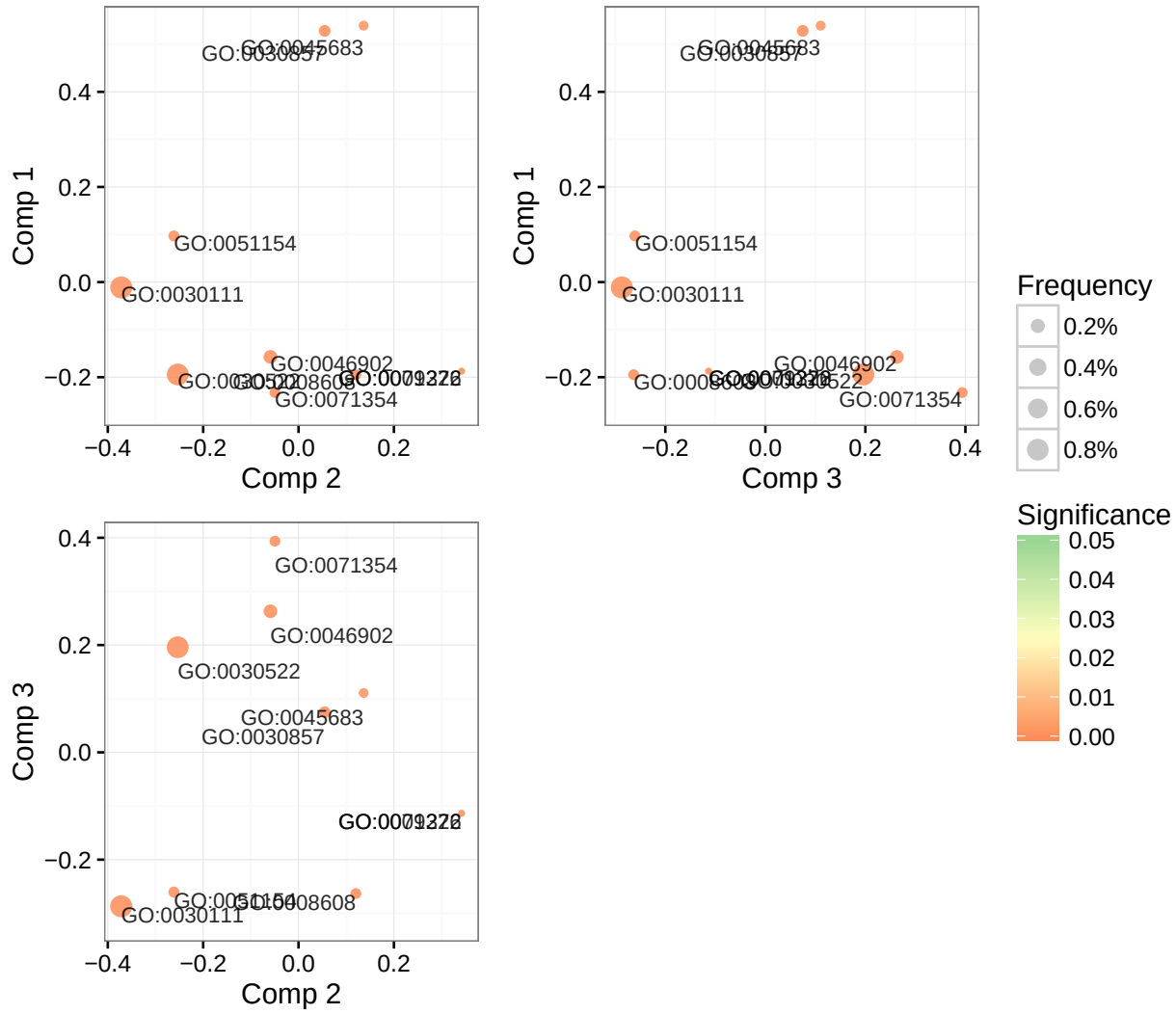
Finally plot the results of the clustering and MDS.

```
TCGAome::plot_mds(goa_bp_term_clustering, all = TRUE)
```



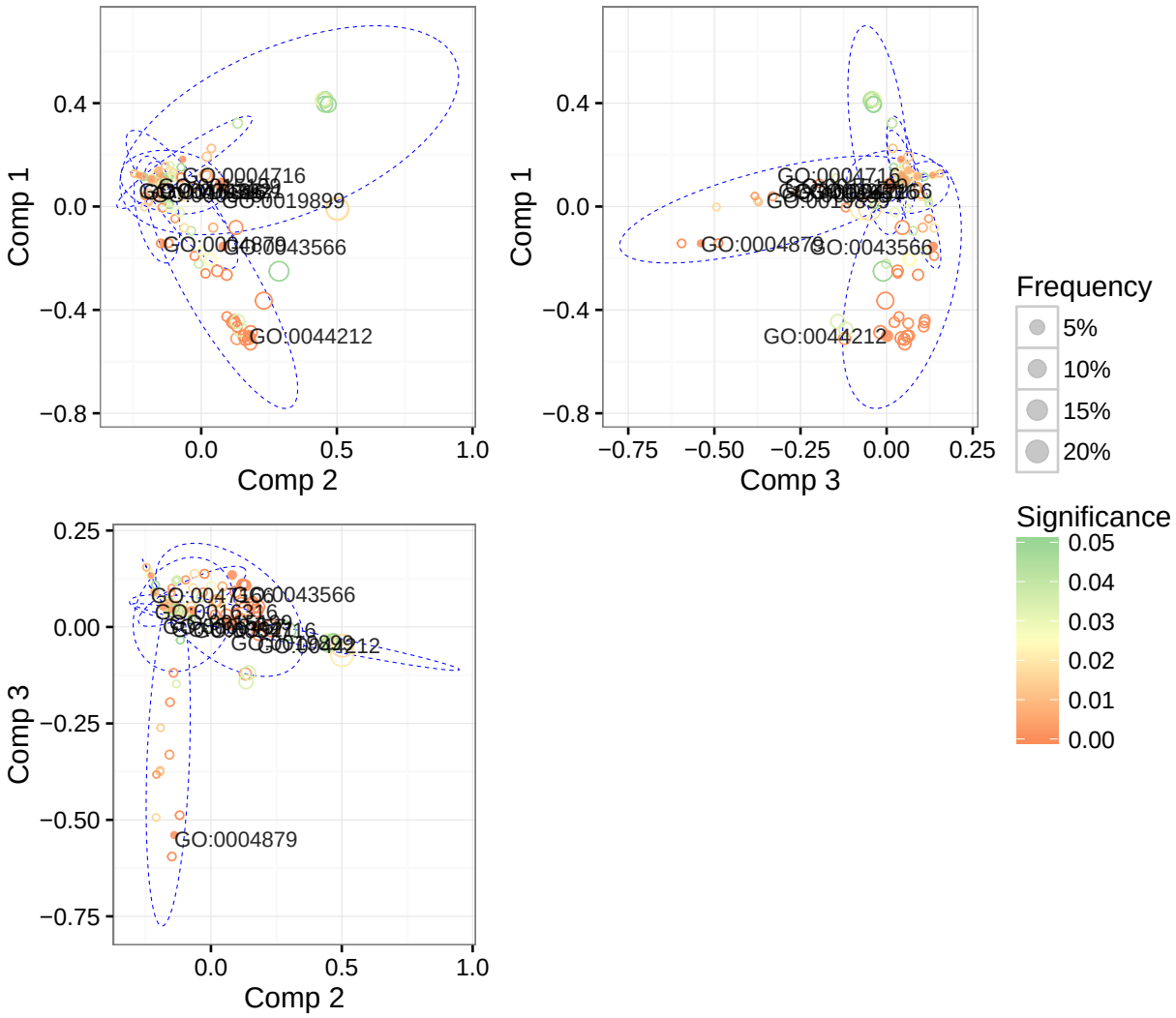
As this scatter plot might quickly get very noisy you may plot only the representative terms for each cluster in the MDS performed over the subset of cluster representatives.

```
TCGAome::plot_mds(goa_bp_term_clustering, all = FALSE)
```

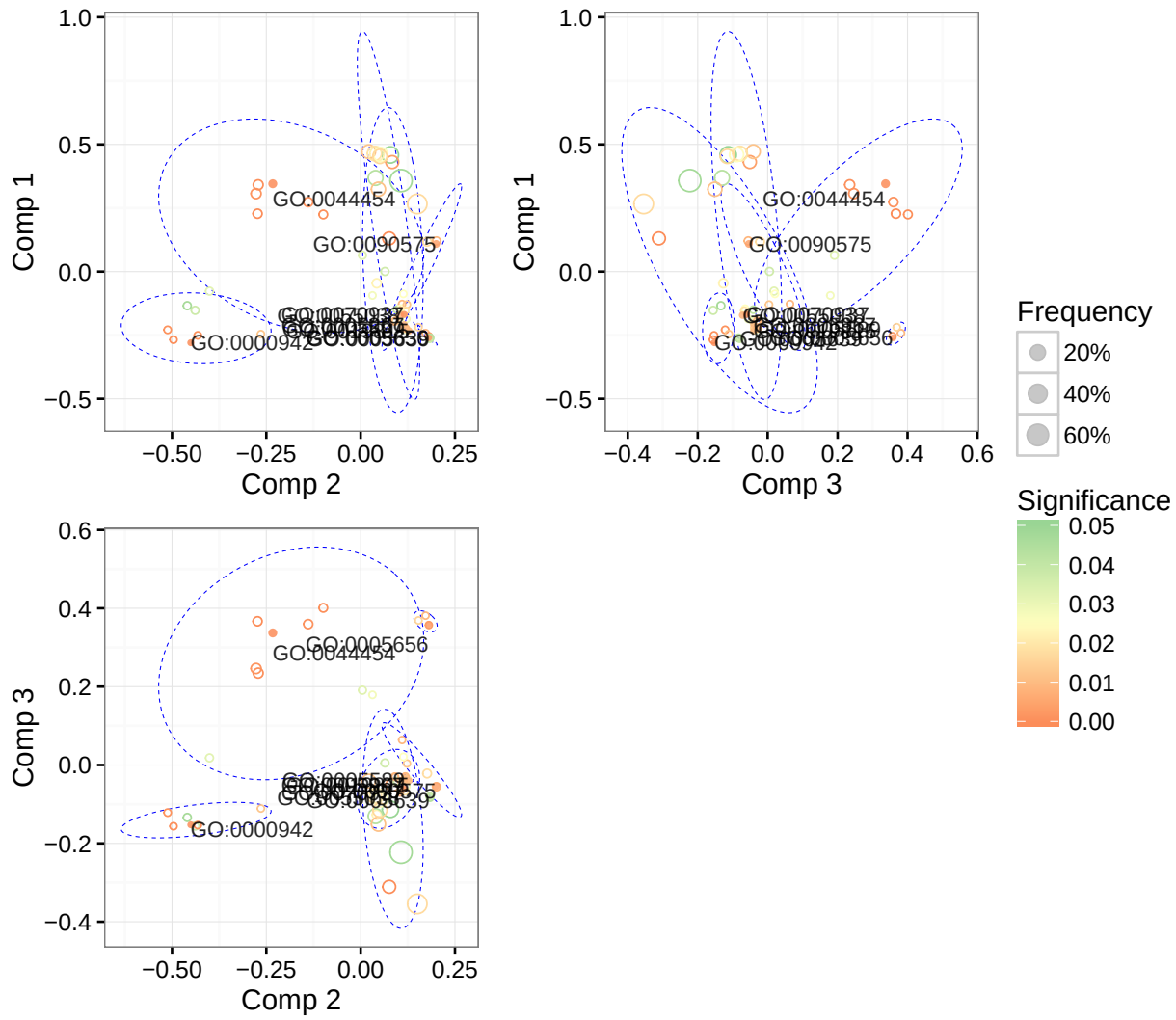


We can do the same for other annotation resources.

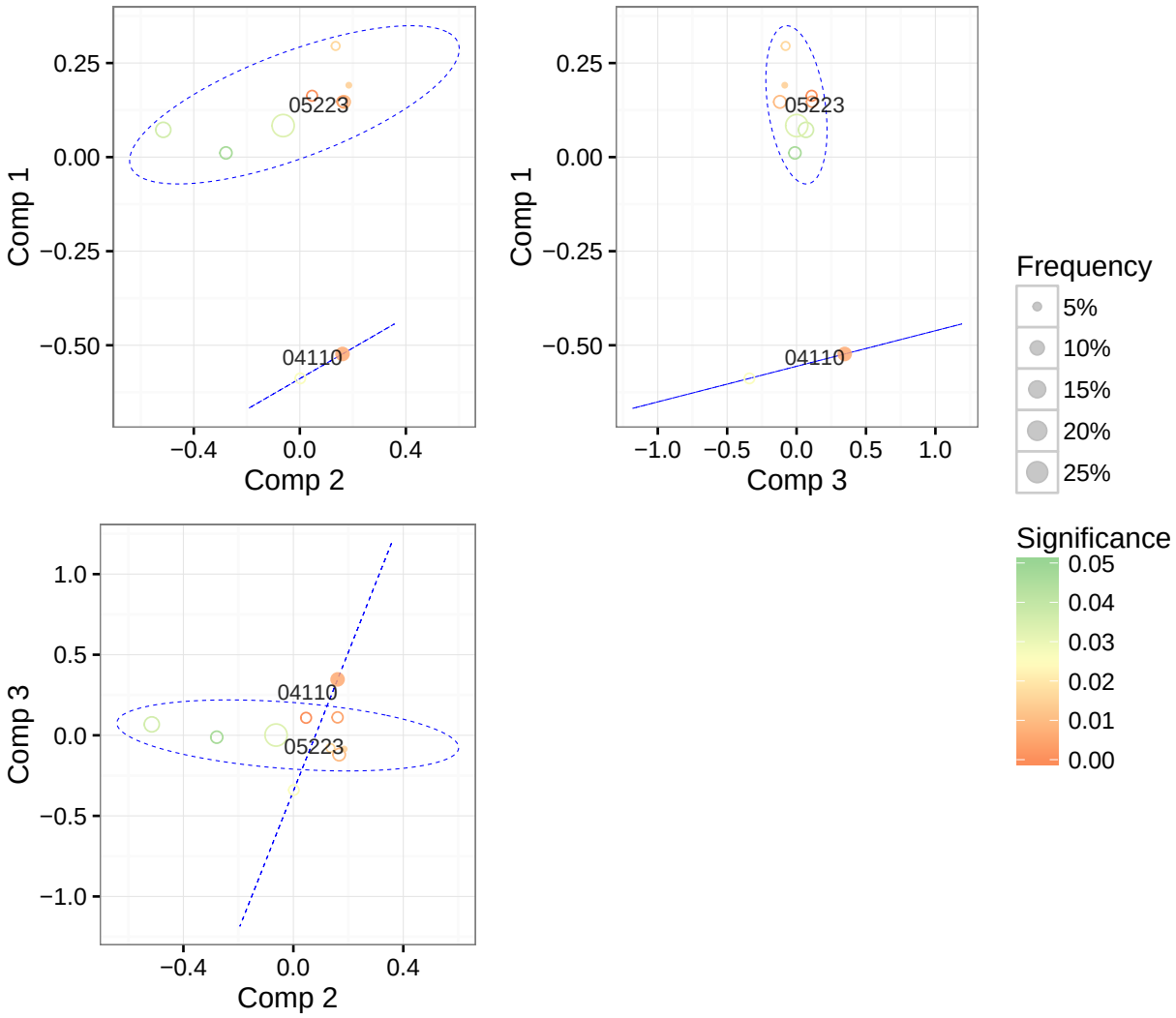
```
goa_mf_enrichment <- TCGAome::get_enrichment(goa_mf, gene_list = gene_list)
goa_mf_term_clustering <- TCGAome::TermsClustering(goa_mf, goa_mf_enrichment, distance_measure = "cosin
TCGAome::plot_mds(goa_mf_term_clustering, all = TRUE)
```



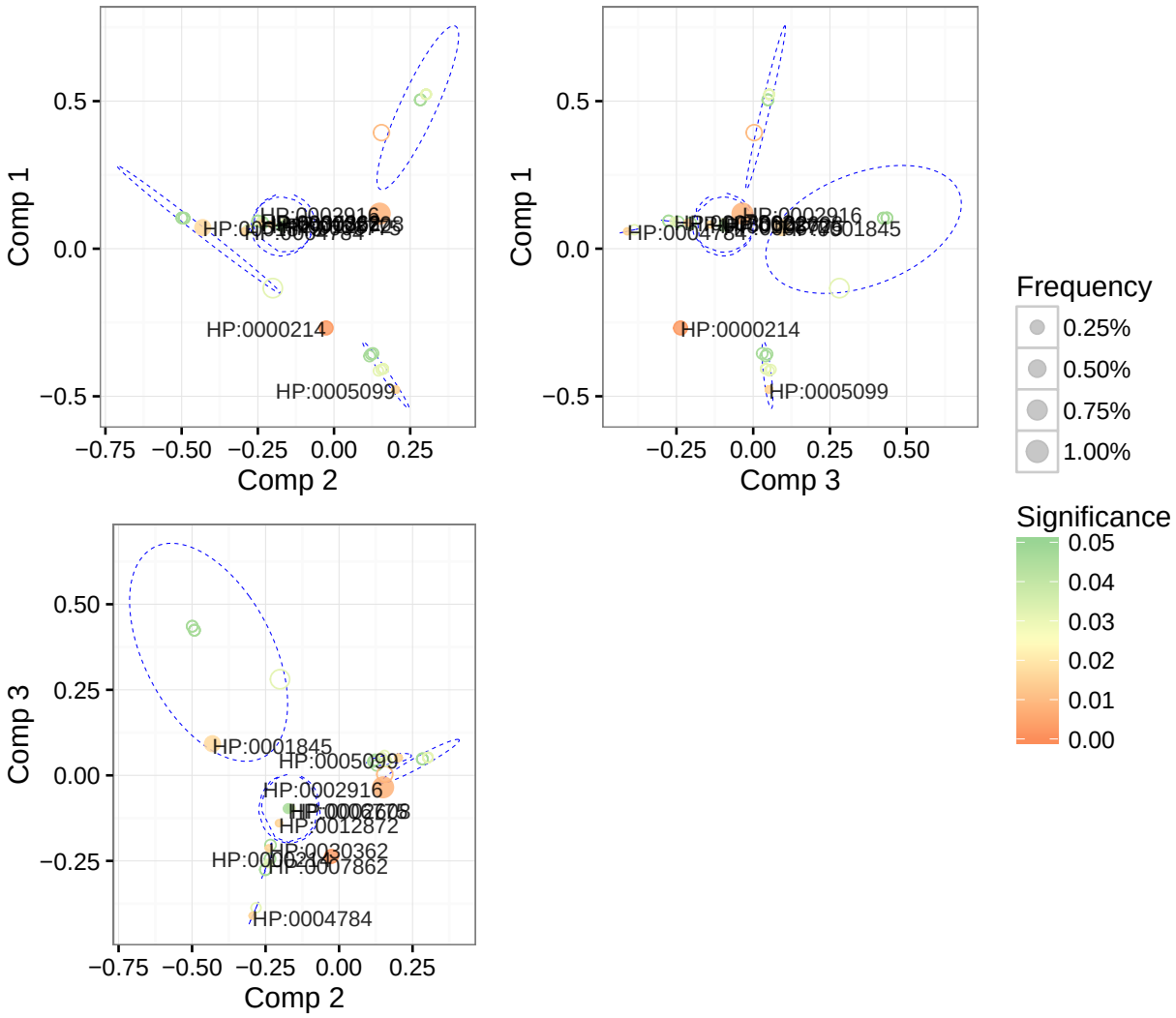

```
goa_cc_enrichment <- TCGAome::get_enrichment(goa_cc, gene_list = gene_list)
goa_cc_term_clustering <- TCGAome::TermsClustering(goa_cc, goa_cc_enrichment, distance_measure = "cosin
TCGAome::plot_mds(goa_cc_term_clustering, all = TRUE)
```



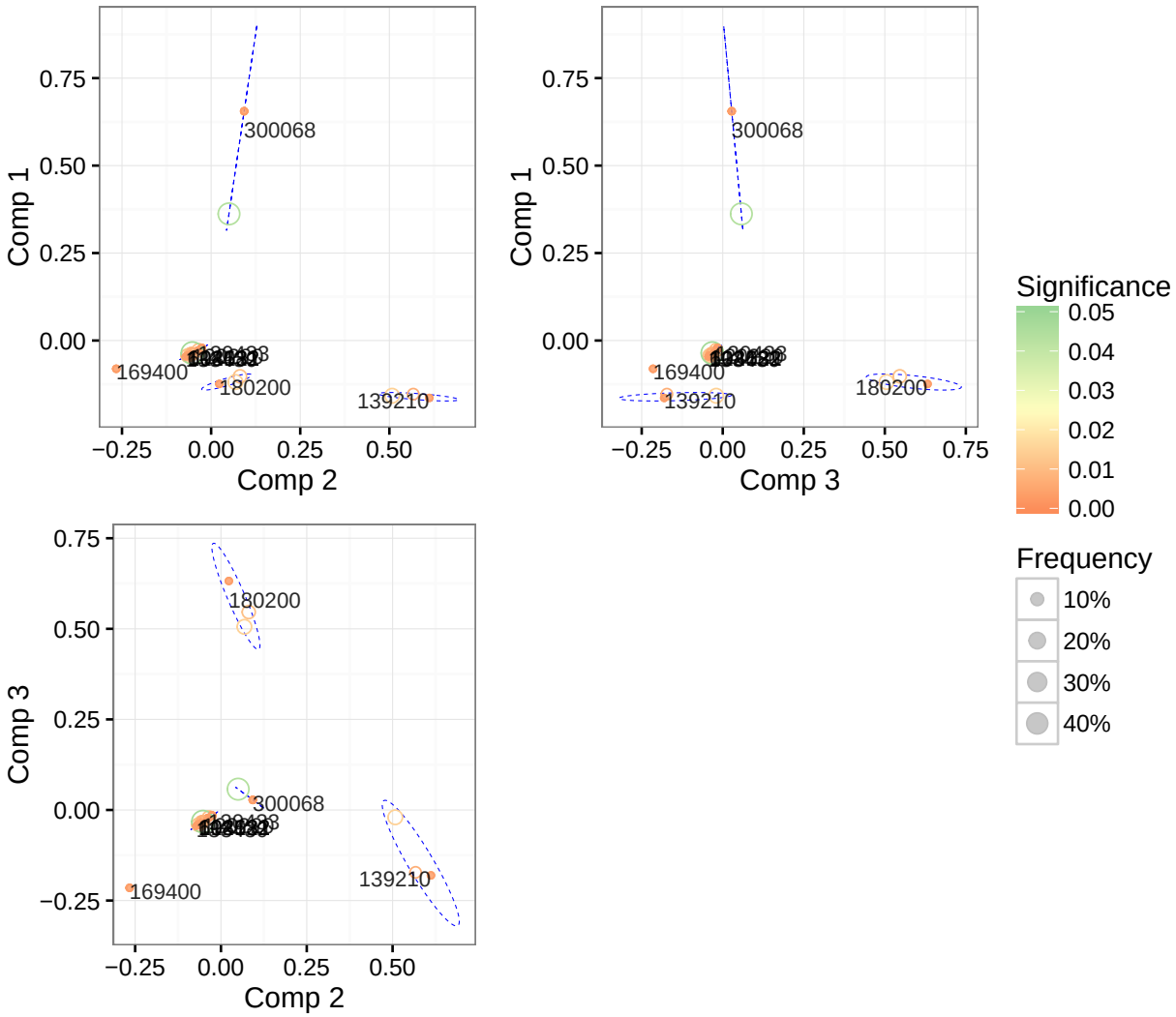

```
kegg_enrichment <- TCGAome::get_enrichment(kegg, gene_list = gene_list)
kegg_term_clustering <- TCGAome::TermsClustering(kegg, kegg_enrichment, distance_measure = "cosine", sig
TCGAome::plot_mds(kegg_term_clustering, all = TRUE)
```




```
hpo_enrichment <- TCGAome::get_enrichment(hpo, gene_list = gene_list)
hpo_term_clustering <- TCGAome::TermsClustering(hpo, hpo_enrichment, distance_measure = "cosine", signi
TCGAome::plot_mds(hpo_term_clustering, all = TRUE)
```



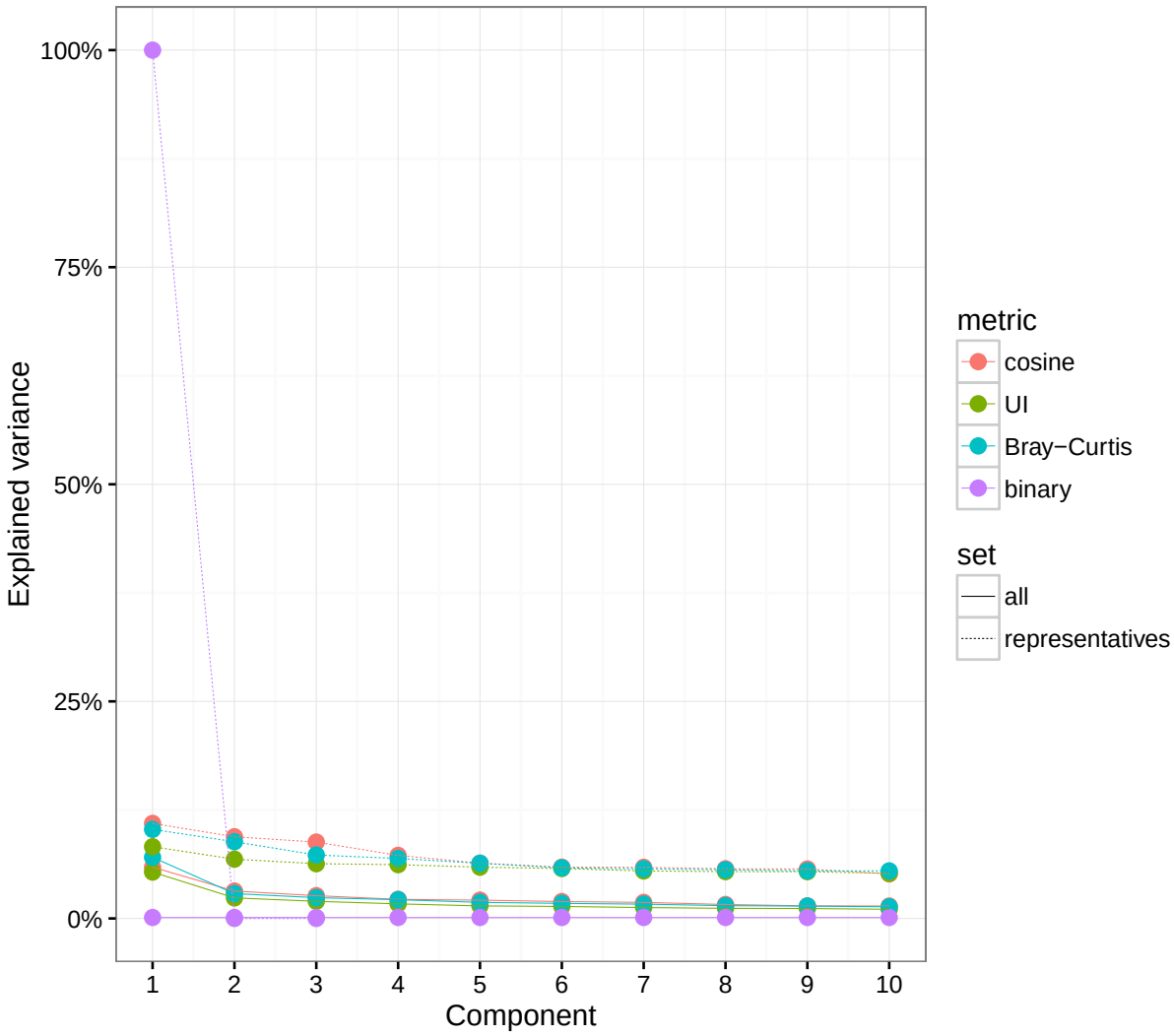

```
omim_enrichment <- TCGAome::get_enrichment(omim, gene_list = gene_list)
omim_term_clustering <- TCGAome::TermsClustering(omim, omim_enrichment, distance_measure = "cosine", sig
TCGAome::plot_mds(omim_term_clustering, all = TRUE)
```



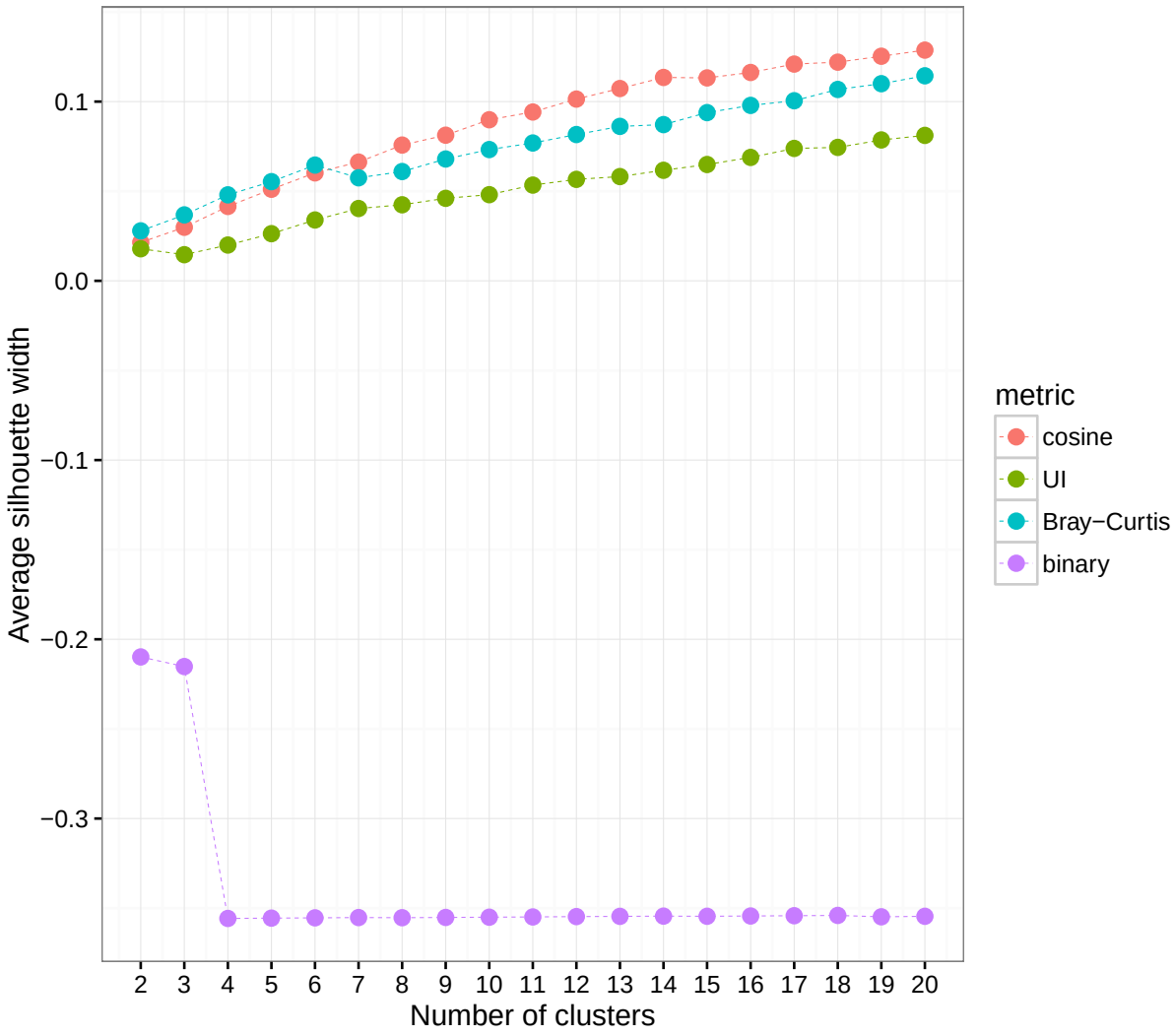
Evaluating the different similarity metrics

The similarity metric employed for clustering and MDS affects how terms are grouped. It is important to assess how well a given similarity metric explains the variance in our annotation terms in the first components of the MDS that will be used to plot our results. This analysis is specific for every annotation and enrichment results, no magic recipe that can be reused.

First thing is to evaluate the explained variance in our MDS space by using each of the similarity metrics.



It is also important to assess how the similarity metric affects the optimal number of clusters.


Average silhouette width
0.1
0.0
−0.1
−0.2
−0.3
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Number of clusters
metric
cosine
UI
Bray−Curtis
binary