CompGO vignette

Sam Bassett, Ash Waardenberg

Developmental and Stem Cell Biology Lab,
Victor Chang Cardiac Research Institute,
Darlinghurst, Sydney, Australia

1 Introduction

This package contains functions to accomplish several tasks relating to Gene Ontology enrichment comparison and visualisation given either .bed files or gene lists. It interfaces with rtracklayer and VariantAnnotation to easily annotate .bed files with genes, and with RDAVIDWebService to generate functional annotation charts based on these gene lists. From here, we provide several methods for comparative visualisation, including viewing the GO term hierarchy in two samples using a directed acyclic graph (DAG), performing z-score standardisation (approximately normal using a log odds-ratio (OR) - Equation 1 and 2) of GO terms returned from DAVID and comparing enrichment via pairwise scatterplots with linear fit and Jaccard metrics (Equation 3). We also provide functions to enable large-scale clustering, KEGG pathway enrichment comparison and PCA. In this vignette, we will comprehensively demonstrate use of this package on the supplied data.

$$Z \sim N(log(OR), \sigma^2) \tag{1}$$

$$z_i = \frac{log(OR)}{SE(OR)} \tag{2}$$

$$J_c = \frac{A \cap B}{A \cup B} \tag{3}$$

2 Annotating .bed files

The first step in this analysis pipeline is to generate individual Gene Ontology data for each dataset. We provide an interface to generate a list of genes from input .bed coordinates with the function anotateBedFromDb, which can take either a system path or a GRanges object as input. Here we use the example files provided with the package.

```
> library(CompGO)
> library(TxDb.Mmusculus.UCSC.mm9.knownGene)
> # load all data packages into R:
> data(gata4)
> data(mef2a)
> data(nkx25)
> data(p300)
> data(srf)
> data(tbx5)
> # Create the GRanges objects:
> gata4.range = GRanges(seqnames=gata4$chromosome,
+     IRanges(start = gata4$start, end = gata4$end))
> mef2a.range = GRanges(seqnames=mef2a$chromosome,
+     IRanges(start = mef2a$start, end = mef2a$end))
> nkx25.range = GRanges(seqnames=nkx25$chromosome,
+     IRanges(start = nkx25$start, end = nkx25$end))
> p300.range = GRanges(seqnames=p300$chromosome,
+     IRanges(start = p300$start, end = p300$end))
> srf.range = GRanges(seqnames=srf$chromosome,
+     IRanges(start = srf$start, end = srf$end))
> tbx5.range = GRanges(seqnames=tbx5$chromosome,
+     IRanges(start = tbx5$start, end = tbx5$end))
> # Note that the window around the .bed region beyond which
> # genes are cutoff is 5kb, this is modifiable
> gata4.annotated = annotateBedFromDb(gRanges = gata4.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> mef2a.annotated = annotateBedFromDb(gRanges = mef2a.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> nkx25.annotated = annotateBedFromDb(gRanges = nkx25.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> p300.annotated = annotateBedFromDb(gRanges = p300.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> srf.annotated = annotateBedFromDb(gRanges = srf.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> tbx5.annotated = annotateBedFromDb(gRanges = tbx5.range,
+     db = TxDb.Mmusculus.UCSC.mm9.knownGene)
> head(gata4.annotated)

GRanges with 6 ranges and 2 metadata columns:
      seqnames               ranges strand |     tx_id         gene_id
         <Rle>            <IRanges>  <Rle> | <integer> <CharacterList>
```

```
  [1]     chr1 [16678537, 16699686]      + |        75           17087
  [2]     chr1 [16678537, 16699686]      + |        76           17087
  [3]     chr1 [34068670, 34365497]      + |       130           13518
  [4]     chr1 [34068670, 34365497]      + |       131           13518
  [5]     chr1 [34272114, 34286091]      + |       134           13518
  [6]     chr1 [34282354, 34319096]      + |       135           13518
  ---
  seqlengths:
           chr1         chr2         chr3 ...  chrY_random chrUn_random
      197195432    181748087    159599783 ...     58682461      5900358
```

3 Generation of Gene Ontology data

Once we have these annotated GRanges objects, we can derive lists of genes to query DAVID and retrieve a full functional annotation chart. Alternatively, a list of genes can be directly supplied to DAVID. Note that there are several parameters available for tweaking the results returned from DAVID: users can specify p value and count cutoffs; however, by default, no thresholding is performed. Since DAVID requires registration before the use of their Web Service R package, you should register first then substitute the email here for your own. This step can take some time, since it depends upon the latency of DAVID's server.

```
> gata4.fnAnot = getFnAnot_genome(gata4.annotated$gene_id,
+     email = "your@email.com", listName="gata4", getKEGG=TRUE)
> mef2a.fnAnot = getFnAnot_genome(mef2a.annotated$gene_id,
+     email = "your@email.com", listName="mef2a", getKEGG=TRUE)
> nkx25.fnAnot = getFnAnot_genome(nkx25.annotated$gene_id,
+     email = "your@email.com", listName="nkx25", getKEGG=TRUE)
> p300.fnAnot = getFnAnot_genome(p300.annotated$gene_id,
+     email = "your@email.com", listName="p300", getKEGG=TRUE)
> srf.fnAnot = getFnAnot_genome(srf.annotated$gene_id,
+     email = "your@email.com", listName="srf", getKEGG=TRUE)
> tbx5.fnAnot = getFnAnot_genome(tbx5.annotated$gene_id,
+     email = "your@email.com", listName="tbx5", getKEGG=TRUE)
>
> # once the charts have been retrieved, they can be written
> # to disk for subsequent batch processing like so:
> # write.table(funChart1, "./path/to/text.txt")
> # however, here we continue with the generated charts in R.
```

4 Clustering and PCA

When faced with a large volume of data, like we are here, it is useful to first get an overall perspective on how similar or different each set is to each other set. We include a function to do interactive PCA and dendrogram-style clustering on large volumes of data, but that doesn't work well in a non-interactive environment - here, we just use the constituent functions plotPCA and plotDendrogram.

```
> # First, put all the charts from the previous step into a list.
> chartList = list("gata4" = gata4.fnAnot, "mef2a" = mef2a.fnAnot, "nkx25" = nkx25.fnAnot,
+     "p300" = p300.fnAnot, "srf" = srf.fnAnot, "tbx5" = tbx5.fnAnot)
```

```
> plotDendrogram(chartList)
```

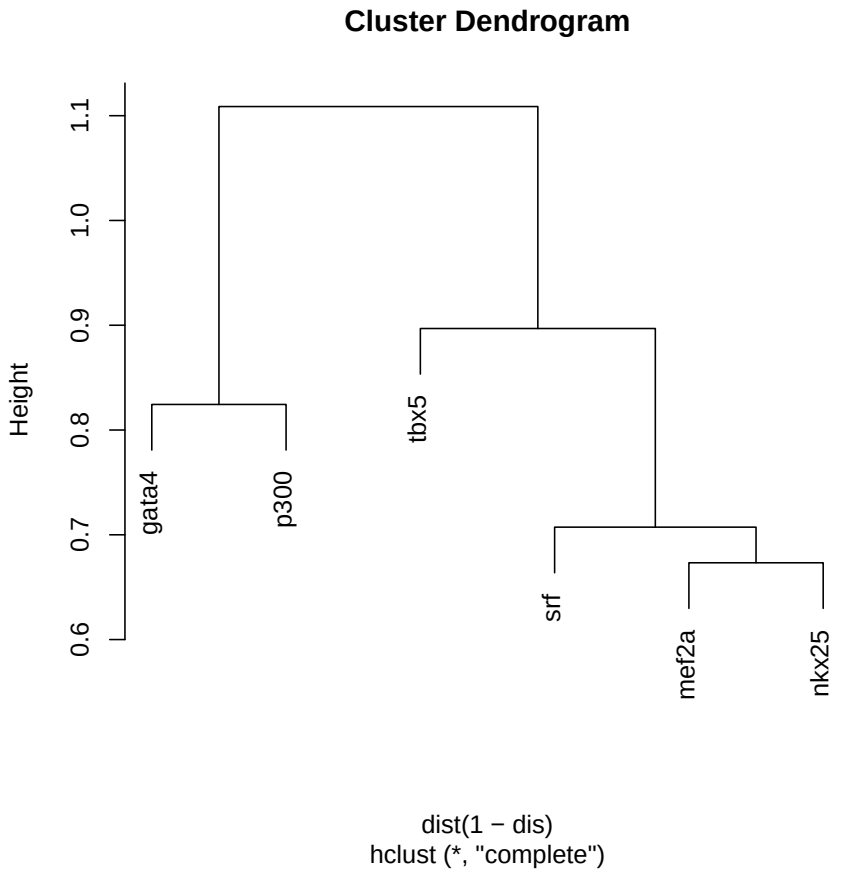

Figure 1: A dendrogram of the sample input data. This gives a good general overview of similarity between individual input sets, later methods show exactly how similar they are (and why).

```
> plotPCA(chartList)
```

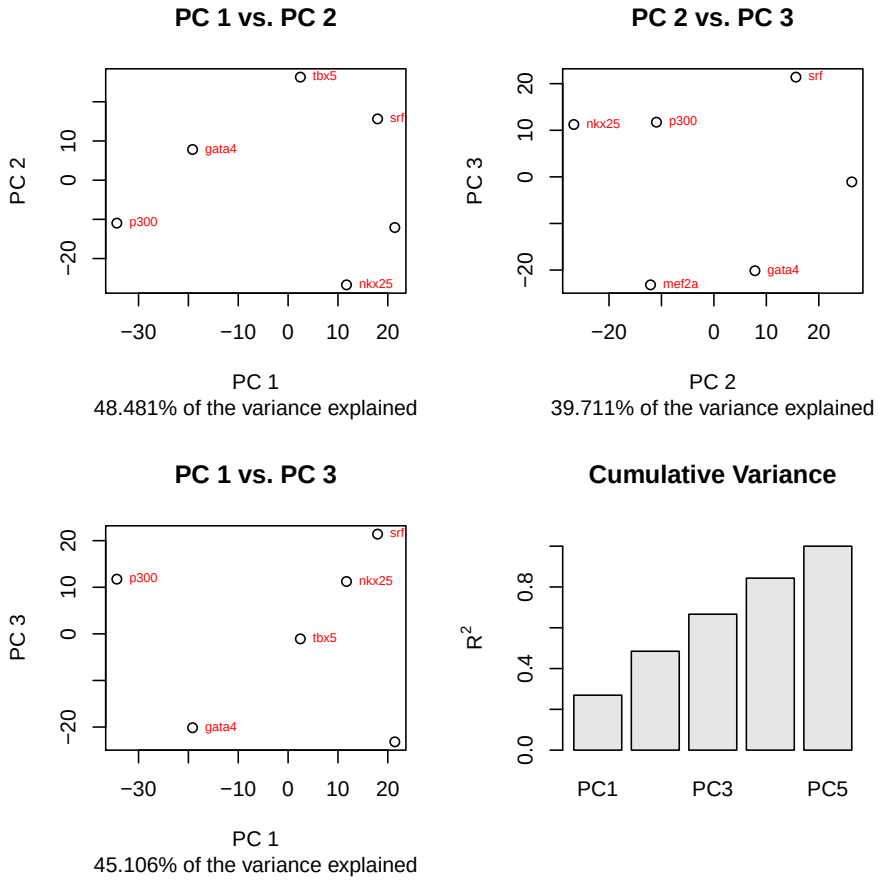

Figure 2: A PCA of the sample input data

5 Result refinement and visualisation

Now that a general overview of similarity has been generated using PCA and clustering, it is helpful to understand the differences between specific transcription factor binding sites. The pairwise plots of z-transformed GO terms is very useful as a metric for overall similarity, while the GO term graph shows the specific differences between two sets.

```
> plotZScores(nkx25.fnAnot, mef2a.fnAnot)

[1] "0.5785"
```

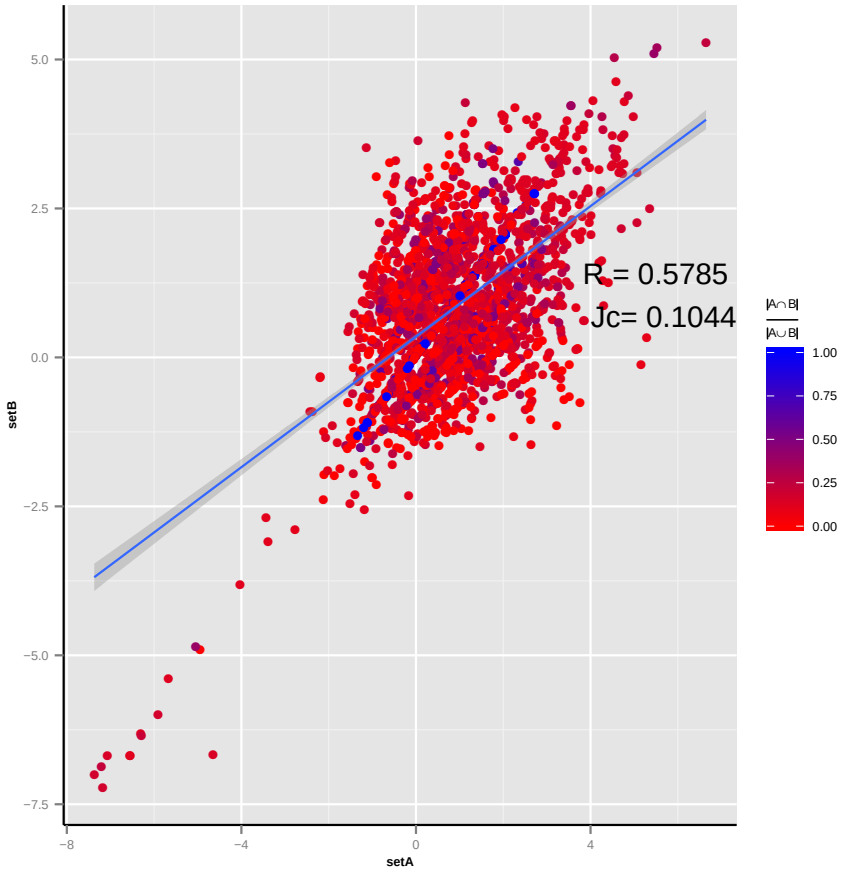

Figure 3: Non-thresholded pairwise plot of Z-scores between Nkx2.5 and Mef2a, two of the more similar sets from the dendrogram. Each point is a GO term, and the colour of each point represents the overlap of genes associated with that term (the Jaccard coefficient). While the Jaccard coefficient remains very low at 0.1, the overall correlation is much higher at 0.58.

```
> plotZScores(p300.fnAnot, mef2a.fnAnot)

[1] "0.3808"
```

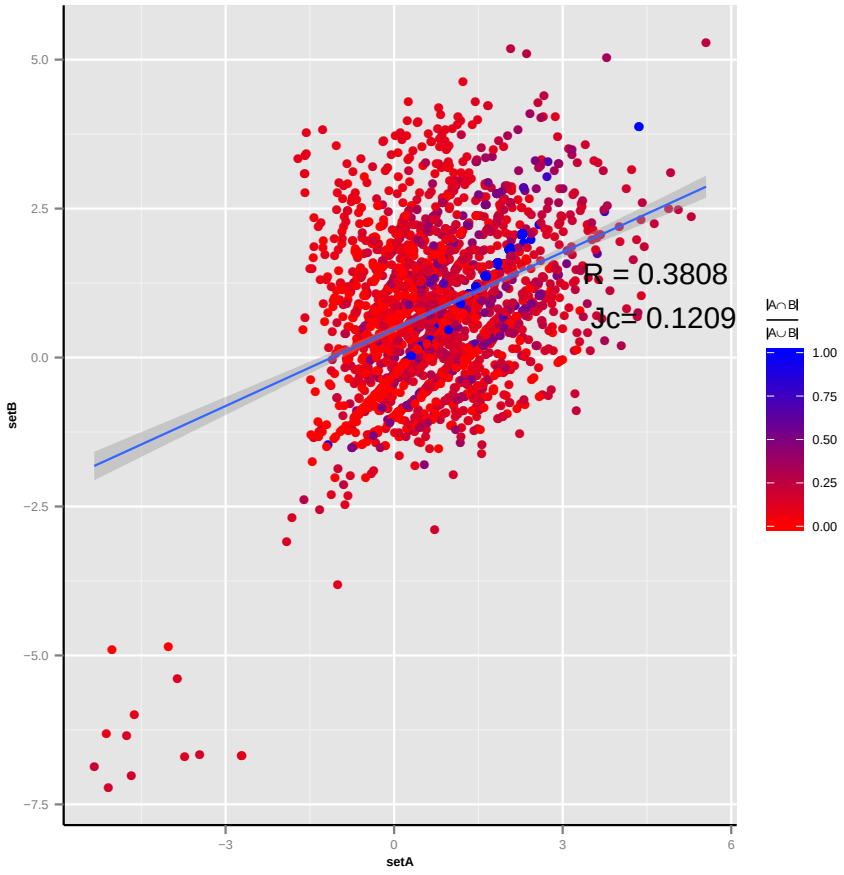

Figure 4: p300 against Mef2a, two of the more different sets. Interestingly, the Jaccard coefficient between these two sets is marginally higher than before, but the overall correlation is reduced to 0.38.

```
> plotZRankedDAG(nkx25.fnAnot, mef2a.fnAnot, ont="MF", n=50)
```

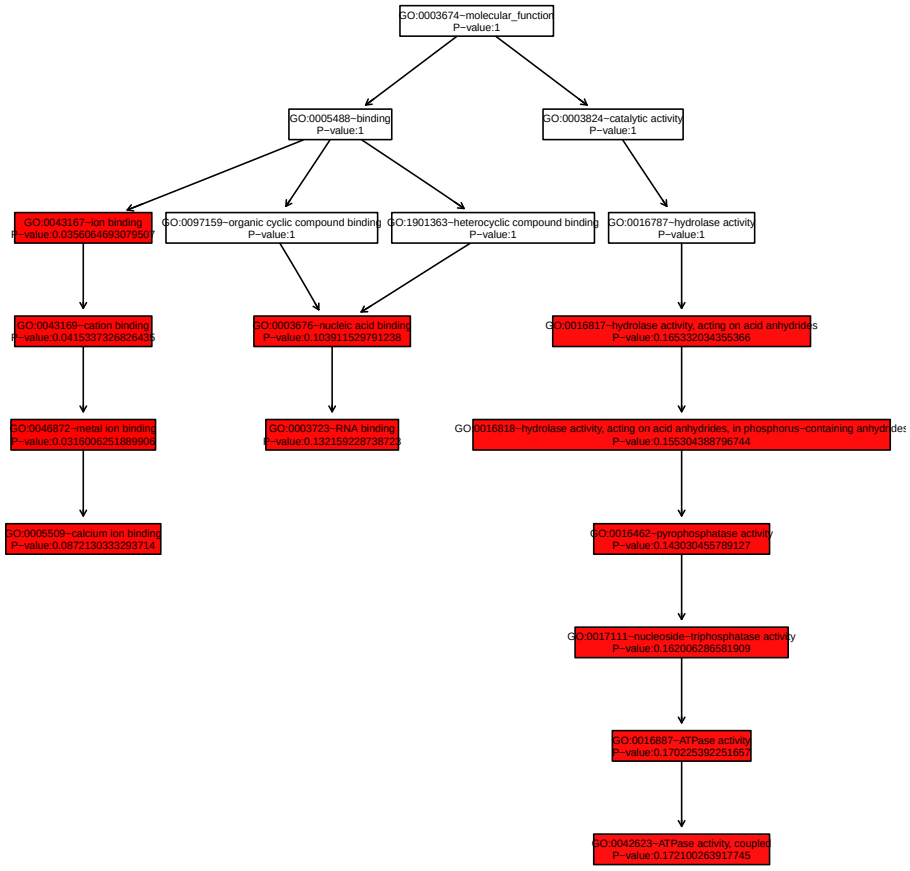

Figure 5: Significant differences between Nkx2.5 and Mef2a in molecular function.

```
> plotZRankedDAG(p300.fnAnot, mef2a.fnAnot, ont="MF", n=50)
```

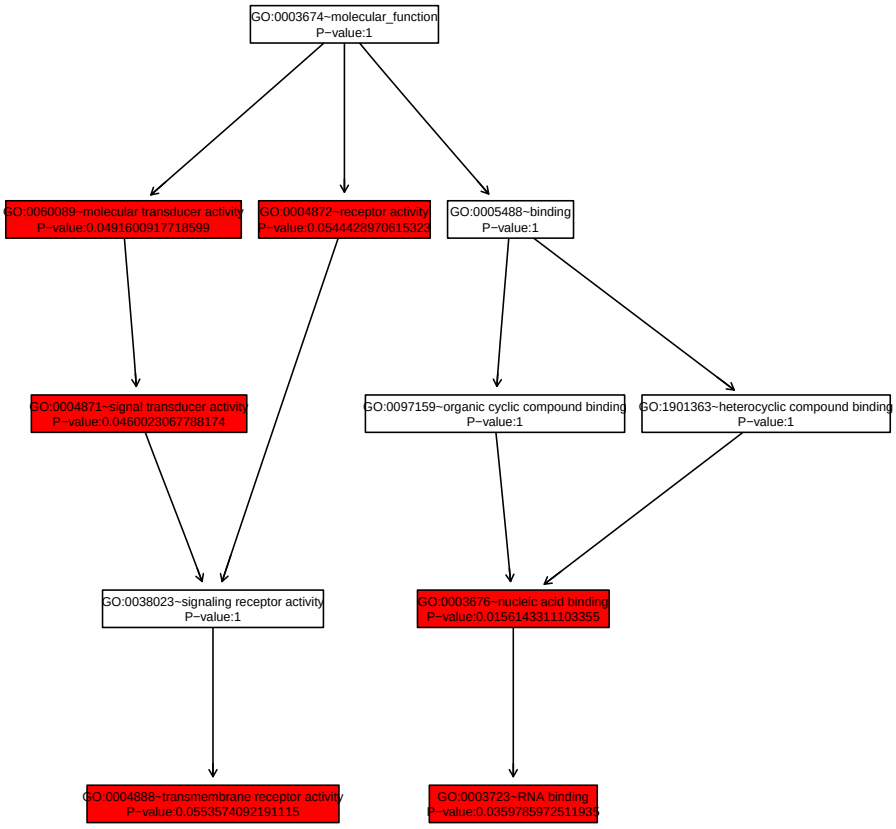

Figure 6: Significant differences between p300 and Mef2a in molecular function.