GWAS Enrichment: Example Workflow Using Current Tools

Brian Lee
Florida Atlantic University

Abstract

Genome Wide Association Studies are popular in helping to identify genetic variants associated with specific phenotypes, especially disease-based phenotypes. The data from the studies can be analyzed in many ways, depending on the goals of research. One popular analysis is enrichment analysis, which commonly includes both Gene Ontology analysis and biological processes analysis. There are many tools available for these analysis, but deciding which tools to use can be a daunting task for new users. This paper proposes a workflow with example scripts that new users can build upon to get started in the field.

I. Introduction

With the mass amount of data that is continuing to be produced during this genomics era, the use of computer analysis is paramount in digesting this information in order to form hypothesis and further advance knowledge, especially in the area of disease genetic variation and disease gene identification through results obtained via Genome Wide Association Studies (GWAS). With that, many tools that have been produced to help analyze this data. The trouble is that a lot of early software or even newer software either becomes obsolete or completely abandoned. Google searches for tools can be misleading in recommending tools that were once popular, but are now outdated. The massive development of the field has also brought about a massive influx of tools that may be useful, but troubleshooting issues becomes difficult because of the lack of a significant user base.

The purpose of this paper is to highlight some useful tools that can help users begin analyzing GWAS results. This will start with translating GWAS results to genetic symbols to be used for further analysis, followed by enrichment analysis' for both Gene Ontologies (GO) and biological pathways via ReactomeDB.

II. Methods

The first criteria for choosing appropriate tools were to find tools which had the greatest extensibility. Extensible tools allow the user to get many types of output data from their set of input data, and they also allow easy integration of allowing the output of one analysis to be the input of a second analysis of a different type. Through this criteria, standalone GUI applications and web services fall behind programming based tools. Standalone/web-based applications can be somewhat restrictive in their parameters, but more importantly it is much more difficult to develop an efficient workflow when relying on these types of tools as compared to programming based tools which can easily produce all types of desired data quickly through scripting, thus producing more results in less time. The two programming languages that are generally recommended for biological computations are Python, mostly through Biopython, and R, mostly through BioconductoR.

The second criteria was accessibility to support through a userbase. This criteria allows a user to get help and advice with issues they may run into while using these tools, a characteristic that is indispensable. Perhaps the biggest communities for bioinformatics discussion are BioStars, SeqAnswers, and, arguably, the #bioinformatics IRC channel. SeqAnswers tends to deal with topics involving meta-analysis of microarray and NGS data, rather than further downstream analysis that we focus on in this paper. BioStars discussions are heavily geared toward R over Python, and the same is true for discussions on #bioinformatics IRC channel. After narrowing down the scope of tools to those that are R/BioconductoR based, the tools used and example scripts in this paper were produced with help and recommendations of the BioStars, #bioinformatics, and #R communities.

In order to demonstrate the process of using these tools, we will attempt to do GO enrichment and biological pathway enrichment using GWAS data of coronary artery disease obtained from Kumar and Elliot (2018).

III. Results

Kumar and Elliot give GWAS data results in reference SNP ID number format. In order to convert this to gene symbol, we can use the *rsnps* library (Chamberlain, Ushey, & Zhu, 2018). An example for how this is done is shown below:

```
### Load appropriate libraries
library(scales)
library(rsnps)
### Put SNPs from Kumar and Elliot into vector
SNPs <- c("rs17114036", "rs11206510", "rs515135", "rs646776",
      "rs4845625", "rs17464857", "rs16986953", "rs6544713",
      "rs1561198", "rs2252641", "rs6725887", "rs9818870",
      "rs17087335", "rs1878406", "rs7692387", "rs273909",
      "rs12526453", "rs6903956", "rs17609940", "rs10947789",
      "rs12190287", "rs2048327", "rs4252120", "rs2023938",
      "rs10953541", "rs11556924", "rs3918226", "rs264",
      "rs2954029", "rs3217992", "rs579459", "rs2505083",
      "rs2047009", "rs11203042", "rs12413409", "rs10840293",
      "rs974819", "rs964184", "rs7136259", "rs3184504",
      "rs11830157", "rs9319428", "rs4773144", "rs2895811",
      "rs56062135", "rs7173743", "rs8042271", "rs17514846",
      "rs216172", "rs12936587", "rs46522", "rs7212798",
      "rs663129", "rs2075650", "rs12976411", "rs1122608",
      "rs9982601", "rs180803")
### use rsnps ncbi_query function to map reference snp id's to
      genes via NCBI's dbSNP and create a table of output. Note,
      this will drop many id's that aren't associated with genes
SNP2Gene <- data.frame("SNPs" = SNPs, "Genes" = gsub(":.*", "",
      ncbi_snp_summary(SNPs)$gene2), stringsAsFactors=FALSE)
### Show the output data
SNP2Gene
### Indicate the percentage of rsid's that were dropped
paste(percent(sum(is.na(SNP2Gene$Gene))/length(SNP2Gene$Gene)),
      "of SNPs could not be mapped to Genes", sep=" ")
```

Once we have mapped the rsid's to gene symbols, we can next perform GO enrichment analysis using *clusterProfiler* (Yu, Wang, Han, & He, 2012).

```
### load appropriate libraries
library(org.Hs.eg.db)
library(clusterProfiler)
library(enrichplot)
### Take the gene list from the table, and insert them
      into a new vector
CAD_GENES <- SNP2Gene$Gene
### Convert gene symbols to Entrez ID's that clusterProfiler can
      use
CAD_ENTREZ <- bitr(CAD_GENES, fromType="SYMBOL",
toType="ENTREZID", OrgDb=org.Hs.eg.db)
### Place the Entrez ID's into a new vector
CAD_ENTREZ <- CAD_ENTREZ$ENTREZID
### Perform the enrichment Analysis for GO Biological Processes
CAD_BP_GO <- enrichGO(gene=CAD_ENTREZ, OrgDb=org.Hs.eg.db,
ont="BP", pAdjustMethod="bonferroni", pvalueCutoff=0.05)
### Put the results into a nice looking table, and write the
      table to a file
CAD_BP_GO <- setReadable(CAD_BP_GO, OrgDb=org.Hs.eg.db,
keyType="ENTREZID")
write.csv(CAD_BP_GO, file = "CAD_BP_GO.csv", row.names=FALSE)
### Also show the output in a dotplot
dev.new()
dotplot(CAD_BP_GO, title="CAD GO Enrichment: Biological
      Processes")
### These steps can be extended to also enrich molecular
      function and cellular component enrichments
```

In addition to GO enrichment analysis, we can then easily use the same data to perform biological pathway analysis using *ReactomePA* (Yu & He, 2016).

```
### load appropriate libraries
library(ReactomePA)
library(enrichplot)
### Perform the enrichment using the same CAD_ENTREZ list vector
CAD_PA <- enrichPathway(gene=CAD_ENTREZ, pvalueCutoff=0.05,
minGSSize=8)
### Put the results into a nice looking table, and write the
      table to a file.
CAD_PA <- setReadable(CAD_PA, OrgDb=org.Hs.eg.db,
keyType="ENTREZID")
write.csv(CAD_PA, file = "CAD_PA.csv", row.names=FALSE)
### Also show the output in a dotplot
dev.new()
dotplot(CAD_PA, title="CAD Biological Pathway Enrichment")
```

The results obtained from these scripts can be found at Github: https://github.com/BLee3478/GWAS_Enrichment

IV. Discussion

All of the above scripts can be combined into one large R script. The benefit is that multiple sets of data can be obtained with the click of a button, once a vector containing the SNP rsid's is set. The script can be extended to include other R tools to obtain even more data. In addition to this, any changes that need to be made will not be a time-consuming endeavor. For example, if more SNPs were found and needed to be included, the SNP vector can be edited to include the new SNPs and the script rerun. This is not true for other solutions that use multiple standalone tools or web services, in which any changes in data requires multiple reruns navigating several different tools.

V. Conclusion

There is an overwhelming number of tools currently available to do enrichment analysis using GWAS data. The most efficient workflows are ones that keep all data manipulation in one centralized location, as opposed to fragmented analysis. This paper provides example scripts that will allow new users to quickly dive into enrichment analysis, which is beneficial in doing hands on learning by getting a basic script set up quickly and can serve as a foundation for further detailed analysis.

VI. References

Chamberlain, S., Ushey, K., & Zhu, H. (2018). rsnps: Get "SNP" ("Single-Nucleotide" 'Polymorphism') Data on the Web (Version 0.3.0). Retrieved from https://CRAN.R-project.org/package=rsnps

Kumar, D., & Elliott, P. (Eds.). (2018). *Cardiovascular Genetics and Genomics: Principles and Clinical Practice*. Retrieved from https://www.springer.com/us/book/9783319661124

Yu, G., & He, Q.-Y. (2016). ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization. *Molecular BioSystems*, *12*(2), 477–479. https://doi.org/10.1039/C5MB00663E

Yu, G., Wang, L.-G., Han, Y., & He, Q.-Y. (2012). clusterProfiler: an R Package for Comparing Biological Themes Among Gene Clusters. *OMICS: A Journal of Integrative Biology*, *16*(5), 284–287. https://doi.org/10.1089/omi.2011.0118