Tutorial:

NCBI2R – An R package for annotating SNPs, Genes and microsatellites.

September 2010 – release 1.3.1

Table of Contents

1. Introduction

NCBI2R annotates lists of SNPs and/or genes, with current information from NCBI. Functions are provided that with one command will annotate the results from genome wide association studies to provide a broader context of their meaning. Other functions enable comparisons between a user's GWA results, and candidate snp/gene lists that are created from keywords, such as specific diseases, phenotypes or gene ontology terms. Commands are simple to follow and designed to work with R objects to integrate into existing workflows. The output produces text fields and web links to more information for items such as: gene descriptions, OMIM, pathways, phenotypes, and lists of interacting and neighboring genes. Annotation can then be used in R for further analysis, or the objects can be customized for use in spreadsheet programs or web browsers. The NCBI2R package

was designed to allow those performing the genome analysis to produce output that could easily be understood by a person not familiar with R.

I have no affiliation with NCBI, and if you have any queries about these functions, check the FAQ on the website and if that doesn't help, send me an email (ncbi2r@gmail.com). Packages will be available for download at the website (http://drop.io/NCBI2R_package Note the underscore between the wordsNCBI2R and package in the URL) as well as the CRAN website (http://cran.r-project.org/). Additional features are being tested at the moment and so a few updates are expected in the July-September period. The website incorporates chat functions and presentations so I can describe the new features when they become available. You can subscribe to the website to find out when it is updated.

2. Downloading, Installing and basic use

It's as simple as following the standard instructions for installing a package. For example, if you are using the Microsoft Windows operating system, open R (you will need administrator rights), and go to the Packages menu and select Install packages. Select the nearest location to you, and then find NCBI2Rfrom the list. To install on a LINUX/UNIX server, please ask your administrator. The installation only needs to be performed once on each machine, but each time you wish to use the commands, you will need to bring them into the current workspace. To do this, type the following:

library("NCBI2R")

These functions make queries of NCBI and there are delays built into the NCBI2R package to stop flooding the NCBI servers too much. The average user querying their results will not have any problems. The NCBI2R functions use a NCBI sanction method called eutils (short for eutilities) and there are two main rules about the eutils - No more than three queries per second (functions have built in delays to offset this), and no more than 100 queries in a batch. If you're going to run a lot of things, NCBI would like you to run things outside peak hours (see The NCBI site under User Requirements), or just put some pauses in your scripts with this function:

Delay(60) # This function will insert a delay for sixty seconds

Most of the NCBI2R functions group together many pieces of information into a single request of NCBI, and then parses a very large file. For example, running **GetSNPInfo** in the following example, only uses a single query of NCBI because it groups all the SNPs together. If the function contains an argument called 'batchsize' this will divide up your query by this number.

snplist<-c("rs1234567", "rs334", "rs848484")

GetSNPInfo(snplist) #runs all three SNPs as one single request (default is 200 together)

GetSNPInfo(snplist,batchsize=1) #runs each SNP individually.

The results are exactly the same depending on how you split them up – the only thing that will change is the speed. Depending on your internet download speed, and your processing speed, you might be better changing the batchsize variable.

References:

http://eutils.ncbi.nlm.nih.gov/#UserSystemRequirements

http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html

3. Brief examples of some functions

3.1 Obtaining SNP Information with GetSNPInfo

snplist<-c("rs1234567","rs334","rs848484")

snpdf<-GetSNPInfo(snplist)

print(snpdf) #the locusID is the NCBI Entrez locus reference number

marker genesymbol locusID chr chrpos fxn_class

1 rs1234567 7 97263169

2 rs334 HBB 3043 11 5204809 frameshift,missense,reference

3 rs848484 PHTF2 57157 7 77387258 intron

3.2 Annotation – the easy way

One of the simplest commands is to annotate a list of snps. These might be the best snps from your association study. The list could be hundreds of items long. It tells you the gene symbol, the EntrezID (known in NCBI as locusID), the chromosome, the position on the chromosome, fxn_class (eg intron, exon, UTR), and the species.

snplist<-c("rs1234567","rs333","rs848484")

scottsobject<-AnnotateSNPList(snplist,"Scottsoutputfile.html")

The last command will create a HTML output file for you, and also create the R object (scottsobject) in case you wish to use the results for other work. You can see in the html file that the SNP rs333 was found on two lines and split in the HTML table – one row for each gene, and the SNP is displayed in bold. Similar to this command are two others– AnnotateSNPFile and AnnotateDataframe. All three take aparticular type of starting object and complete the annotation. AnnotateSNPfile takes as input a filewhich contains the list of SNPs, and AnnotateDataframe takes a dataframe as input (eg from genomewide association software). This last function also allows you to specify columns from your results that you might want to include, such as p values and effect sizes. These three functions, like all the functions in this vignette, are described in greater detail in the manual.

3.3 Obtaining Gene Information

Each gene has a unique Entrez ID number within the NCBI system. The same gene name, in different species, will have different locusID numbers. These numbers are the unique identifier and are used throughout the NCBI2R package so the command we will use is GetIDs, which allows you to find out the name of the number. The defaults are best described in the manual, but briefly, the defaults are human genes only, current genes only, and the database used is the gene database.

GetIDs("CLN5[sym]")

The answer returned is "1203", the same as the Entrez gene number you'll find on the NCBI website.[sym] stands for gene symbol and applies a filter to the results to prevent us from obtaining multiple non-specific matches. Instead of [sym] perhaps you could also use [pref] as detailed on the NCBI website.

For more information on filters and limits that might be used, see the manual

GetIDs("bone[DIS]")

Returns a lot of Entrez IDs of genes involved in bone disease. Of course if you wanted to work with these numbers you would store the results into the R workspace by typing the following:

a<-GetIDs("bone[DIS]")

b<-GetGeneInfo(a) #Works with multiple ID numbers and returns lots of information about the

specified genes.

MakeHTML(b,"output_for_someone.html")

Other examples:

GetIDs("CLN5") #remember the default species is human, see the manual for how to change that.

It returns four genes ID numbers because the search is quite non specific, just as the NCBI website. Thesearch term will match to any gene that mentions that term. To match the actual gene symbol use [sym]as described above.

c<-GetIDs("ENST0041422") #ensembl reference

d<-GetIDs("protein binding[GO] ") #gene ontology term

e<-GetIDs("KEGG pathway: Cytokine-cytokine receptor interaction")

3.4 UniSTSFromName – Finding the position of microsatellites

Perhaps you have linkage results, and want to figure out the physical position on sequence maps

GetUniSTSFromName("D3S1234")

Returns three dataframes, one containing brief information about a gene, if the marker is located within one. Another dataframe details any of the maps (linkage and sequence) that the marker is found in, along with their position. A third dataframe provides any information about errors displayed on theNCBI website, such as the marker appearing in more than one position on the genome.

3.5 Exons, Introns, Interactions and Ontologies.

number<-GetIDs("CLN5[sym]")

GetGeneTable(number) #shows the sizes and positions of the exons, introns (Set refers to multiple transcript) and identifiers of DNA and Prot sequence.

GetInteractions(number) #provides a dataframe of interaction genes, and links/ID numbers

GetGOs(number) #provides a dataframe of gene ontology information.

3.6 GetPubMed

The following code will create an R object, and download any papers that are freely available on PubMed central. You can also create a spreadsheet in Microsoft Excel, that will contain basic information eg author, year, title, journal, volume, issue, pages. The spreadsheet will also contain a link to the local copy (if it was downloaded), and a link to the PubMed page for you to follow the journal links where other access methods to the papers may be available to you depending on the access of your group.

Make a simple Microsoft Excel document of all the available references found. The Excel document willalso create clickable links to the online pubmed abstracts (where you will find links to other articles).Ifyou downloaded the free articles, there will be a link to the local copy from this spreadsheet.

asd<-GetPubMed("CLN5",”References.tab”,download=TRUE)

The spreadsheet is really a tab delimited file, with a few specially formatted columns for Microsoft Excel.

Please see the manual for discussions about additional setting that may be required in your region/language.

3.7 Pathway Analysis

Let's say you have a particular SNP that is located in a gene. You'd like to look at all the SNPs in your association results to see if they were all significant.

favSNP<-"rs4294787"

favSNPInfo<-GetSNPInfo(favSNP)

pathway<-GetPathways(favSNPInfo$locusID)

genes_in_pathway<-GetIDs(pathways$name)

biglist<-GetSNPsInGenes(genes_in_pathway)

3.8 Linkage Disequilibrium Analysis

Check out the help and manual pages for a greater discussion but this is just a quick overview.

qw<-GetSNPProxyInfo("rs1234567")

This will return information on the best proxy SNP in the region. It will only be one row of information unless you change one of the default settings. The keepmode argument of the command allows you to keep either the best SNP, or all of the SNPs – so that you can filter based on your results. There are lots more options available so check out the manual. With keepmode set to 3, all the available information on this SNP (using other default settings of the function) will be downloaded and saved as the variable qt.

qt<-GetSNPProxyInfo("rs1234567",keepmode=3)

Or if you're not interested in a particular SNP, but just to get regional information, you can use the following command.

fg<-GetLDInfo("4",123456789,123556789) #up to a limit of 2Mb only.

But please remember that all positions on these LD functions correspond to the Hapmap build, and the genomic positions may be different on the NCBI build which is used by the other functions. Please be careful when you go between the two sets of functions – The **GetSNPPosHapmap** command could be useful to convert between the two maps.