How do R-Peridot modules work

This is a complete guide about how R-Peridot modules work and how to create entirely new modules.

Index

Input Files

All R-Peridot analyzes are based on two initial files, which are formatted and filtered versions of files passed by the user:

- ‘rna-seq-input.tsv’: These are the quantitative data provided by the user, now formated to the following format:

gene-id	**<sampleID>**	**<otherSampleID>**	**...**	**<lastSampleID>**
<gene-id-1>	<count-reads>	<count-reads>	...	<count-reads>
<gene-id-2>	<count-reads>	<count-reads>	...	<count-reads>

The columns are tab separated ("\t") and the values of count reads are integers in decimal basis.

- ‘condition-input.tsv’: They are the conditions/groups of each sample in the file 'rna-seq-input.tsv'. Columns are also separated by "\t":

sample	**condition**
<sampleID>	<condition of sampleID>
<otherSampleID>	<condition of otherSampleID>
...	...
<lastSampleID>	<condition of lastSampleID>

Both condition and sample names can not have spaces or special characters.

Module Standardization

In order for R-Peridot to be able to manage different modules in a generic way it was necessary to use a "standardization" for the modules. This standardization involves: what arguments will be passed by the command line to the R process that will execute the module script, a universal way of passing different types of parameters and determining input and output files.

First there are the command line arguments. They are passed directly on the command that starts the instance of R. In this command line, the last 4 arguments are information for the script:

```
$ [default arguments of R] [script folder] [input files folder] [output files folder] [0: if this is the first execution of this script | 1: if it is not the first execution]
```

All scripts, their information and results are placed in a folder called '.r-peridot-files', located in the user's folder, where there are two main folders, 'results' and 'scripts'.

'results' is where the final results of the scripts that have already finished are saved, so they can be used as input by other scripts. In it there is the 'rna-seq-input.tsv' file, which contains the gene expression data provided by the user and the 'condition-input.tsv' file describes which groups each sample belongs to. For each module that was executed there is a file that ends with '.output' containing the text output of the script and also a folder containing all its results.

The 'scripts' folder contains the modules folders, each with:

- The file 'config.txt': contains a table where the parameters are passed from the R-Peridot interface to each module;
- The 'description' file describes information about the module, formally defining it;
- The R script of the module;
- And the 'results' folder: location where the script should put its generated results;

Parameter Types

The parameters are technical information provided to the module from the user interface. Each parameter type has certain possible values:

Type	Values
Float	Real numbers greater than 0.
Integer	Natural numbers equal or greater than 0.
GeneIdType	It is the type of identifiers in the first column: “none”, “kegg”, "ACCNUM", "ENSEMBLTRANS", "EVIDENCEALL" ,"IPI", "ONTOLOGYALL", "PROSITE", "UNIGENE", "ALIAS", "ENTREZID",

	"GENENAME", "MAP", "PATH", "REFSEQ", "UNIPROT", "ENSEMBL", "ENZYME", "GO", "OMIM", "PFAM", "SYMBOL", "ENSEMBLPROT", "EVIDENCE", "GOALL", "ONTOLOGY", "PMID" or "UCSCKG".
Organism	The reference organism. There are three available: “Human”, “Mouse” and “Fly”.

R-Peridot automatically fills in some parameters, even if they are not required for all modules. They and their default values are:

Parameter	Default
pValue (Float)	0.01
fdr (Float)	0.05
log2FoldChange (Float)	0.01
tops (Integer)	0
geneIdType (GeneIdType)	“none”
referenceOrganism (Organism)	“Human”

The “description” file

It is a text file where each line describes an information about the module. Each line begins with an information category between '[' and ']', followed by a space and then the information itself.

Category	Information	Mandatory?
[NAME]	The module name. Do not use spaces.	Yes
[SCRIPT-NAME]	The name of the script file R. It must be in the same folder.	Yes
[RESULT] *or* *[MANDATORY-RESULT]*	The name of a result file generated by the script. If "[MANDATORY-RESULT]" is used, the module will necessarily need to generate this result so that execution is considered to be successful.	At least one
[REQUIRED-INPUT-FILE]	Name of a user-supplied file (described in "Input Files") or generated by another module. In the latter case, the syntax is: <module name>.<AnalysisModule or PostAnalysisModule>/<fileName>	No
[REQUIRED-SCRIPT]	Name of a module required to run this new module.	No

[MAX-2-CONDITIONS]	The input data can group the samples into two or more groups. If the module supports more than 2 groups: "false". Otherwise: "true".	Yes
[NEEDS-REPLICATES]	If the module requires that at least one of the conditions has more than one sample: "true". Otherwise: "false".	Yes
[INFO]	Textual description of the module. Each line starting with "[INFO]" adds a new line in the module description.	At least one line
[REQUIRED-PARAMETER]	A parameter that needs to be supplied to the module: <parameter name> <type>. Read "Parameter types", above.	No

Analysis and Post Analysis Modules

There are 2 different types of modules: analysis and post-analysis. Analysis modules have a well-defined role, they receive as input the gene expression data and use them to select differentially expressed genes. But the post-analysis modules have a freer role. These are necessarily executed after the analysis and can use both the results of the analysis modules and the other post-analysis modules to generate any type of result. R-Peridot identifies a folder as being an analysis module when it has a name terminated with ".AnalysisModule" and as a post-analysis when the name ends with ".PostAnalysisModule".

The minimum requirements for a module to be an analysis module are to generate a table (tab-separated) with the lines of "rna-seq-input.tsv" that were considered differentially expressed, a MA plot, a Volcano plot, a Histogram and a PDF file containing the generated plots mentioned. In addition, there are also certain standard parameters required. In the "description" file, it looks like this:

```
[RESULT]     MAPlot.png
[RESULT]     histogram.png
[RESULT]     plots.pdf
[MANDATORY-RESULT]  res.tsv
[RESULT]     volcanoPlot.png
[REQUIRED-INPUT-FILE]     condition-input.tsv
[REQUIRED-INPUT-FILE]     rna-seq-input.tsv
[REQUIRED-PARAMETER]      fdr    Float
[REQUIRED-PARAMETER]      log2FoldChange      Float
[REQUIRED-PARAMETER]      pValue Float
[REQUIRED-PARAMETER]      tops   Integer
```

Unlike analysis modules, which have a specific purpose, the post-analysis modules are free as to which parameters, input files, and results to define.

Coding R Scripts to R-Peridot

If we are to describe what a script in an R module does in a simple way it would be “read 'config.txt' parameters, read input files and then generate result files". Both input and output files must reside in folders defined by R-Peridot, which are passed as command-line arguments to R. Reading these arguments can be done as follows:

```
args = commandArgs(trailingOnly = F)
#Read the path to the local dir
localDir <- args[length(args)-3]
localDir
#Read the path to the input files directory
inputFilesDir <- args[length(args)-2]
inputFilesDir
#Read the path to the output (result) files directory
outputFilesDir <- args[length(args)-1]
outputFilesDir
#1 in case the script has already been executed 1 or more times with the
#same expression data. This is useful if you want to store temp. variables
#in a “.RData” file or something similar.
notFirstRun <- args[length(args)]
notFirstRun

setwd(localDir)
```

And to instantiate a table in R with the parameters, you can use the following code:

```
#Read table with the parameters
FileConfigPath = paste(localDir, "config.txt", sep = "/")
FileConfig = read.table(FileConfigPath, header = TRUE, row.names = 1, sep =
"|")
#To access the “fdr”: FileConfig$fdr. For the “pValue”: FileConfig$pValue
```

Input files have to be read from the input files directory:

```
peridotConditions = read.table(paste(inputFilesDir, "condition-input.tsv",
      sep =   "/"), header=TRUE, row.names=1)
peridotConditions
#Read file path
inputTableFile = paste(inputFilesDir, "rna-seq-input.tsv", sep = "/")
#Read file
peridotCountTable = read.table(inputTableFile, header=TRUE, row.names=1)
```

After performing all the processing, it is time to save the results. The following example comes from an analysis module:

```
png(filename = paste(outputFilesDir, "volcanoPlot.png", sep = "/"),
```

```
       width=600, height=600)

#Volcano Plot
with(res, plot(logFC, -log10(PValue), pch=20, main="Volcano plot"))
with(subset(res, FDR<FileConfig$fdr), points(logFC, -log10(PValue), pch=20,
       col="red"))
abline(v=c(FileConfig$log2FoldChange, FileConfig$log2FoldChange*(-1)),
        col="blue")
legend('topleft', c(paste("FDR < ", FileConfig$fdr, sep = ""),
       paste("Log2FoldChange = mod(", FileConfig$log2FoldChange, ")",
        sep = "")),
       col = c("red", "blue"), bty = 'o', pch = c(15, NA), lty = c(NA, 1),
        bg = "white", cex = 0.8)
dev.off()

resSig = subset(res, PValue <FileConfig$pValue)
write.table(resSig, paste(outputFilesDir, "res.tsv", sep = "/"), sep =
"\t")
```

Conclusion

So now you have all the information necessary to create modules for R-Peridot. Just write a script file, a “description” file and put them in a directory named *something*.<AnalysisModule or PostAnalysisModule>. Then, move your new module to “YOUR_USER_DIR/.r-peridot-files/scripts/”. If you did it right, R-Peridot will load your module next time it is opened. If it did not, check the log file in “YOUR_USER_DIR/.r-peridot-files/log.txt”, it will probably have information pointing to where exactly you made a mistake.

If you don’t like doing all of this directly, you can use the Modules Manager to create modules more easily. Don’t forget that your module is completely portable: besides by copying the module directory, you can also Import/Export modules through the GUI.

Pitágoras Alves <alves.pitagoras@gmail.com>
November 2 of 2017
Natal-RN, Brazil