Como os módulos do R-Peridot funcionam

Arquivos de input

Todas as análises do R-Peridot se baseiam em dois arquivos iniciais, os quais são versões formatadas e filtradas dos arquivos passados pelo usuário:

- ‘rna-seq-input.tsv’: São os dados quantitativos de expressão fornecidos pelo usuário, no seguinte formato:

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

As colunas são separadas por tabulação (“\t”) e os valores de count reads são números inteiros em base decimal.

- ‘condition-input.tsv’: São as condições/grupos de cada amostra no arquivo ‘rna-seq-input.tsv’. As colunas também são separadas por “\t”:

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

Tanto os nomes de condições quanto de amostras não podem ter espaços ou caracteres especiais.

Padronização de módulos

Para que R-Peridot fosse capaz de gerenciar diferentes módulos de uma forma genérica foi necessário utilizar uma “padronização” para os módulos. Essa padronização envolve: quais argumentos serão passados pela linha de comando para o processo do R que irá executar o script do módulo, uma forma universal de passar diferentes tipos de parâmetros e determinação de arquivos de *input* e *output*.

Primeiramente há os argumentos de linha de comando. Eles são passados diretamente no comando que inicia a instância do R. Nesta linha de comando, os 4 últimos argumentos são informações para o script:

$ [argumentos padrão do R] [pasta do script] [pasta dos arquivos de entrada] [pasta dos arquivos de saída] [0: caso seja a primeira execução deste script | 1: caso não seja a primeira execução]

Todos os scripts, suas informações e os resultados ficam numa pasta chamada '.r-peridot-files', localizada na pasta do usuário, onde há duas pastas principais, a 'results' e a 'scripts'.

A 'results' é onde ficam guardados os resultados finais dos scripts que já finalizaram, de forma que eles podem ser usados como entrada por outros scripts. Nela há o arquivo 'rna-seq-input.tsv', que contém os dados de expressão gênica fornecidos pelo usuário e o arquivo 'condition-input.tsv' descreve a quais grupos cada amostra pertence. Para cada módulo que foi executado há um arquivo que termina com '.output' contendo a saída de texto do script e também uma pasta contendo todos os seus resultados.

Já a pasta 'scripts' contém as pastas dos módulos, cada uma com:

- O arquivo 'config.txt': contém uma tabela onde estão os parâmetros passados da interface do R-Peridot para cada módulo;
- O arquivo 'description': descreve informações sobre o módulo, definindo formalmente ele;
- O script em R do módulo;
- E a pasta 'results': local onde o script deve colocar os seus resultados gerados;

Tipos de parâmetros

Os parâmetros são informações técnicas fornecidas ao módulo a partir da interface de usuário. Cada tipo de parâmetro tem certos valores possíveis:

Tipo	Valores
Float	Números reais maiores que 0.
Integer	Números naturais maiores ou iguais a 0.
GeneIdType	É o tipo dos identificadores, na primeira coluna: "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" ou "UCSCKG".
Organism	O organismo de referência. Há três disponíveis: "Human", "Mouse" e "Fly".

O R-Peridot preenche automaticamente alguns parâmetros, mesmo que não sejam necessários para todos os módulos. Eles e seus valores padrão são:

Parâmetro	Valor Padrão
pValue (Float)	0.01
fdr (Float)	0.05
log2FoldChange (Float)	0.01
tops (Integer)	0
geneIdType (GeneIdType)	“none”
referenceOrganism (Organism)	“Human”

O arquivo “description”

É um arquivo de texto onde cada linha descreve uma informação sobre o módulo. Cada linha começa com um categoria de informação entre ‘[‘ e ‘]’, seguida de um espaço e então a informação em si.

Categoria	Informação	Obrigatório
[NAME]	O nome do módulo. Não usar espaços.	Sim
[SCRIPT-NAME]	O nome do arquivo do script R. Deve estar na mesma pasta.	Sim
[RESULT] *ou* *[MANDATORY-RESULT]*	O nome de um arquivo de resultado gerado pelo script. Caso “[MANDATORY-RESULT]” seja usado, o módulo precisará necessariamente gerar esse resultado para que a execução seja considerada como bem sucedida.	Pelo menos um
[REQUIRED-INPUT-FILE]	Nome de um arquivo fornecido pelo usuário (descritos em “Arquivos de input”) ou gerado por outro módulo. Nesse último caso, a sintaxe é: <module name>.<AnalysisModule or PostAnalysisModule>/<fileName>	Não
[REQUIRED-SCRIPT]	Nome de um módulo necessário para a execução deste novo módulo.	Não
[MAX-2-CONDITIONS]	Os dados de entrada podem agrupar as amostras em dois ou mais grupos. Caso o módulo tenha suporte a mais de 2 grupos: “false”. Caso contrário: “true”.	Sim
[NEEDS-REPLICATES]	Caso o módulo precise que pelo menos uma das condições tenha mais de uma amostra: “true”.	Sim

	Caso contrário: “false”.	
[INFO]	Descrição textual do módulo. Cada linha iniciada com “[INFO]” acrescenta uma nova linha na descrição do módulo.	Pelo menos uma linha
[REQUIRED-PARAMETER]	Um parâmetro que precisa ser fornecido para o módulo: <parameter name> <type>. Consulte “**Tipos de parâmetros”**.	Não

Módulos de análise e de pós-análise

Há 2 tipos diferentes de módulos: os de análise e os de pós-análise. Os de análise tem um papel bem definido, recebem como entrada os dados de expressão gênica e os usam para selecionar genes diferencialmente expressos. Já os de pós-análise tem um papel mais livre. Estes são executados necessariamente após os de análise e podem usar tanto os resultados dos scripts de análise quanto dos de pós-análise, para gerar qualquer tipo de resultado. R-Peridot identifica uma pasta como sendo um módulo de análise quando ela tem um nome terminado em “.AnalysisModule” e como um de pós análise quando o nome termina em “.PostAnalysisModule”.

Os requisitos mínimos para um módulo ser um módulo de análise são gerar uma tabela, separada por tabulação (“\t”), com as linhas do “rna-seq-input.tsv” que foram consideradas diferencialmente expressas, um M. A. plot, um Volcano plot, um Histograma e um arquivo PDF contendo os plots gerados. Além disso, também há certos parâmetros padrão necessários. No arquivo “description”, fica da seguinte forma:

```
[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
```

Ao contrário dos módulos de análise, que tem um propósito específico, os de pós-análise são livres quanto à quais parâmetros, arquivos de entrada e resultados definir.

Escrevendo scripts R para o R-Peridot

Se formos descrever o que um script de um módulo do R faz de forma simples seria “ler parâmetros do ‘config.txt’, ler arquivos de entrada e então gerar arquivos de resultado”. Tanto os arquivos de entrada quanto os de resultado tem que ficar em pastas definidas pelo

R-Peridot, as quais são passadas como argumentos de linha de comando para o R. A leitura desses argumentos pode ser feita da seguinte forma:

```
args = commandArgs(trailingOnly = F)
#Ler o caminho do diretório atual
localDir <- args[length(args)-3]
localDir
#Ler o caminho do diretório onde estão os arquivos de entrada
inputFilesDir <- args[length(args)-2]
inputFilesDir
#Ler o caminho do diretório onde devem ser salvos os resultados
outputFilesDir <- args[length(args)-1]
outputFilesDir
#1 caso o script já esteja sendo executado pela segunda vez ou mais com os
#mesmos dados, 0 caso contrário. Isso é útil caso queira armazenar
#variáveis temporárias num arquivo “.RData”.
notFirstRun <- args[length(args)]
notFirstRun

setwd(localDir)
```

Já para instanciar uma tabela no R com os parâmetros, você pode usar o seguinte código:

```
#Ler tabela com os parâmetros
FileConfigPath = paste(localDir, "config.txt", sep = "/")
FileConfig = read.table(FileConfigPath, header = TRUE, row.names = 1, sep =
"|")
#Para acessar o “fdr”: FileConfig$fdr. Para o “pValue”: FileConfig$pValue
```

Arquivos de entrada tem que ser lidos a partir do diretório de arquivos de entrada:

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

Depois de realizar todo o processamento, é hora de salvar os resultados. O seguinte exemplo vem de um módulo de análise:

```
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")
```