Project_Guide

Kyungjin Sohn

2021 6 11

This document is a simple guide about the "Comparative robustness analysis" project. To execute all the code used in this project, the code must be in the correct directory. Please check the link before executing each code. Before entering a description, this project was developed in the following environment.

```
- R : 4.1.0
- RStudio : 1.4.1717
- OS : Windows10
```

1. Calculate robustness scores

First, we need to compute robustness scores using cDNA data. The data is in the "./data" folder and contains six types of cDNA data; HomoSapiens, MusMusculus, Dmelanogaster, Celegans, Scerevisiae, and Ecoli.

Run the first code named *main_01.R*. This function takes arguments and starts the calculation. If you need more description, run the following command on your server.

```
Rscript main_01.R -h
```

And the sample command to execute the *main_01.R* function is like:

```
nice -n19 Rscript main_01.R -i Scerevisiae.fasta.gz -o Scerevisiae -s Scerevisiae
```

The required functions to run *main_01.R* function.

```
    - cal_robustness_score.R : calculate the robustness score of genes
    - cal_REA4eAI.R : calculate the REA for a specific species
    - cal_prob4eAI.R : calculate probability p(c'|c)
```

2. Categorize score data

To do an in-depth analysis of the score data, we categorized the score data based on gene ontology. (There are three types of gene ontology: BP(=Biological Process), CC(Cellular Component), and MF(Molecular Function)) The methods for matching Go terms and ontologies with score data are species-specific, so we distinguished them by different codes. Besides, we used Nucleotide ID to classify both HomoSapiens and MusMusculus score data, so we used the same code for data classification of both species. The function names for this part all begin with *cateBonto_(species).R*

```
cateBonto_NucleotideID("HomoSapiens", "Homo Sapiens")
cateBonto_Ecoli()
```

Since not all sequences have matching GO terms and ontologies(depending on the database), there will be several sequences that do not belong to any categorized score data. There is a code that recombines categorized data. The result will be score data containing only sequences with matching GO terms and ontologies.

```
combine_ONTO_data("HomoSapiens")
```

3. Create random sequences

There are two ways to create a random sequences. First, we can maintain the original GC-contents and make the rest completely random. On the other hand, we can also create data that maintain amino acids and convert codons into only other synonymous codon. We will maintain the original length of the sequence in both ways.

1) Method 1

The GC-contents are the same as the original data. (The function takes two arguments; a species name and a seed number.)

```
createRandomSeq_GC("Scerevisiae", 1111)
```

2) Method 2

The composition of amino acids is the same as the original data. (The function takes two arguments; species name and seed number.)

```
createRandomSeq_AA("Scerevisiae", 1111)
```

4. Draw a histogram by GO term

1) Compute robustness score means for GO term

Before drawing a histogram, we must select the GO term(s) to be the main sample. Using the categorized data based on ontology, we can calculate the robustness score average for each GO group. According to the law of large numbers, we used only GO terms with more than 30 sequences. The result is saved in a CSV file.

```
scoremeanBGO("HomoSapiens", "BP")
```

To run this function *scoremeanBGO.R*, another function is also needed.

```
- subNmodi_scoreONTO.R :
  subset from the categorized score data and modify score to standardize them
```

2) Draw a histogram

The function *draw_histoBGO.R* draws histograms when you pass the species and ontology arguments. This function draws three types of histograms: using the CSV file calculated above, it draws a histogram of all GOs in the top 10%, all GOs in the bottom 10%, and each GO in the top or bottom 10%. In addition, all of these histograms include multiple distributions of random, selected, and total distributions.

```
draw_histoBGO("Scerevisiae", "MF")
```

The function *draw_histoBGO.R* requires many other functions.

```
    - subNmodi_scoreONTO.R : (described above)
    - subNmodi_score.R : subset from the score data and modify score to standardize them
    - sortGO.R : find GOs ranked in the top or bottom 10%
    - plot_histoBGO.R : draw a histogram according to GO terms and save it as a png file
                        [*Need to change seed number and random data type if necessary]
```

5. Draw a histogram by gene

Function *draw_histoBgene.R* draws histograms when the species arguments is passed as an argument. This function draws two types of histograms: the histogram includes the score distribution for a particular species and the score distribution for random data generated according to the 3-1 or 3-2 method. A special part of the function is that among random score data of the same type, there is a subfunction called "*combine_random_data.R*" that combines data with different seed numbers. Thus, it is necessary to change the seed vector based on the seed numbers used when creating random data.

```
draw_histoBgene("Dmelanogaster")
```

The required functions to run *draw_histoBgene.R* function.

```
    - subNmodi_score.R : (described above)
    - plot_histoBgene.R :
      draw a histogram according to gene of the species and save it as a png file
    - combine_random_data.R :
      combines data with different seed numbers among random score data of the same type
```

6. Draw multiple histograms of different species by gene

Function *draw_histoBgene_multi.R* draws histograms of the same type as function *draw_histoBgene.R*, but *draw_histoBgene_multi.R* draws histograms for multiple species. You have a choice of "HomoSapiens", "MusMusculus", "Dmelanogaster", "Celegans", "Scerevisiae", and "Ecoli". In addition, you can select and draw only the scores you want to see on the histogram from "wR2N", "CRI2", "eAI", and "sumScore" robustness scores.
* CAUTION: Because this function draws multiple histograms of different species, selecting fewer than two types of species causes errors.

```
draw_histoBgene_multi(
    c("HomoSapiens", "MusMusculus", "Dmelanogaster", "Celegans", "Scerevisiae", "Ecoli"),
    c("wR2N", "CRI2", "eAI", "sumScore"), "AA")

draw_histoBgene_multi(
    c("HomoSapiens", "MusMusculus", "Dmelanogaster"),
    c("wR2N", "sumScore"), "GC")
```

The required functions to run *draw_histoBgene.R* function.

```
- plot_histoBgene_multi.R :
  draw a histogram according to gene of the species and return plots in a list form
- subNmodi_score.R : (described above)
- combine_random_data.R : (described above)
```

7. Draw a heatmap

Function *draw_heatmapBGO.R* draws heat map for multiple species using score data categorized by ontology. Choose species from “Homo sapiens”, “Musculus”, “Dmelanogaster”, “Celegans”, “Serevisiae”, and “Ecoli”, and also, choose ontology from “BP”, “CC”, and “MF”. In addition, you can draw a heat map based on the robustness score itself or the rank of the robustness score by species. The rows of the heat map are common GOs of different species, and the result is saved as PDF file. According to the law of large numbers, we used only GO terms with more than 30 sequences. * CAUTION: Because this function draws heat map of different species, selecting fewer than two types of species causes errors.

```
draw_heatmapBGO(c("Scerevisiae", "Celegans", "Ecoli"), "BP", "rank")

draw_heatmapBGO(c("HomoSapiens", "MusMusculus"), "MF", "score")
```

The required functions to run *draw_heatmapBGO.R* function.

```
- plot_heatmapBGO.R :
  draw a heatmap based on common GOs of different species, and save the result
  in a PDF file
- subNmodi_scoreONTO.R : (described above)
```

8. Survival Test

First, the survival test selects GO that meets the following conditions:

```
1) Select GOs with more than 30 sequences.
2) Select GOs with at least 5 sequences with sequence lengths between q40 and q60.
```

Next, all sequences are then randomly given 100 different mutation scenarios. [= For one GO term, Number of sequences * 100 = Number of scenarios] The function uses BLOSUM62 matrix(as a default) to calculate the rate of difference between the original score. If the rate of difference is 0.8 or below, the sequence dies. [= score after the mutation/original score <= 0.8]

Run the first code named *main_02.R*. This function takes arguments and starts the survival test. If you need more description, run the following command on your server.

```
Rscript main_02.R -h
```

And the sample command to execute the *main_02.R* function is like:

```
nice -n19 Rscript main_02.R -n HomoSapiens -o MF -m BLOSUM62 -s 1234
```

The required functions to run *main_02.R* function.

```
- survivalTest_subMatrix.R : select GO Terms that meets the conditions from
  the categorized score data of the corresponding species and run survival tests
- survivalTest_filter.R : select GO Terms that meets the conditions from
  the categorized score data
- survivalTest_run.R : run survival tests
- change_CToAA.R : change a sequence of codons into amino acids
- cal_substitution_score.R : calculate a substitution score after
  one random mutation occurrence
```

9. Draw an interactive plot

Function *draw_interactiveplotBGO.R* draws an interactive line plot. You must run a survival test for the species you want to draw before executing this function. And if you hand over the variables you used at that time to this function, this function would draw a graph.

```r
draw_interactiveplotBGO("HomoSapiens", "MF", "BLOSUM62", 1234)
```

The required functions to run *draw_interactiveplotBGO.R* function.

```
- combine_survivalTestBGO.R : combine the survival test result data
```