**RPPA Network Analysis EDA (Workflow) SOP**

Last edited: 10/30/2023

Written by: David Tek

Collaborators: Virginia Espina, Mariaelena Pierobon

**Context**

This workflow is for the analysis of data collected from reverse-phase protein array (RPPA) experiments, with control and experiment groups. The Jupyter Notebook script (available for use locally or through Google Colaboratory) take the RPPA data in the form of a CSV file and calculates the statistical significance of each endpoint (protein). The results of the analysis are then made available in the Jupyter Notebook output cells and in the form of a network visualization in Cytoscape (an open-source network visualization software).

**Requirements**

For Google Colaboratory and Local use:

Cytoscape 3.10.1 or later, installed on user’s computer.

For Local use:

Jupyter notebook v2023.4.1011241018 or later

Python 3.10.9 or later

Python packages:

py4cytoscape

pandas

NumPy

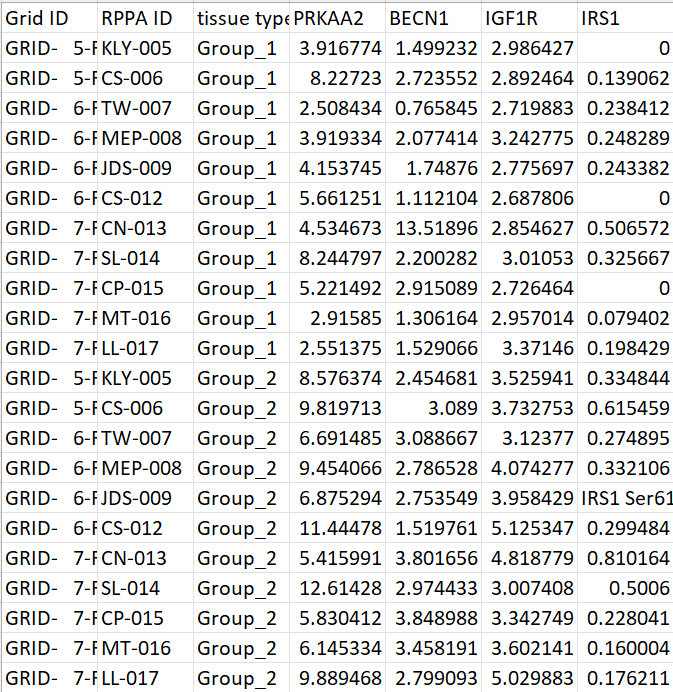
SciPy

**Instructions**

The Endpoint\_data.csv and Network\_map.csv files must be formatted as the example file are for the script to work. The two files must also be exactly named Endpoint\_data.csv and Network\_map.csv unless you are comfortable editing Python scripts. Below are explanations of the two files’ formats.

CSV file formats

Endpoint\_data.csv (See image below)



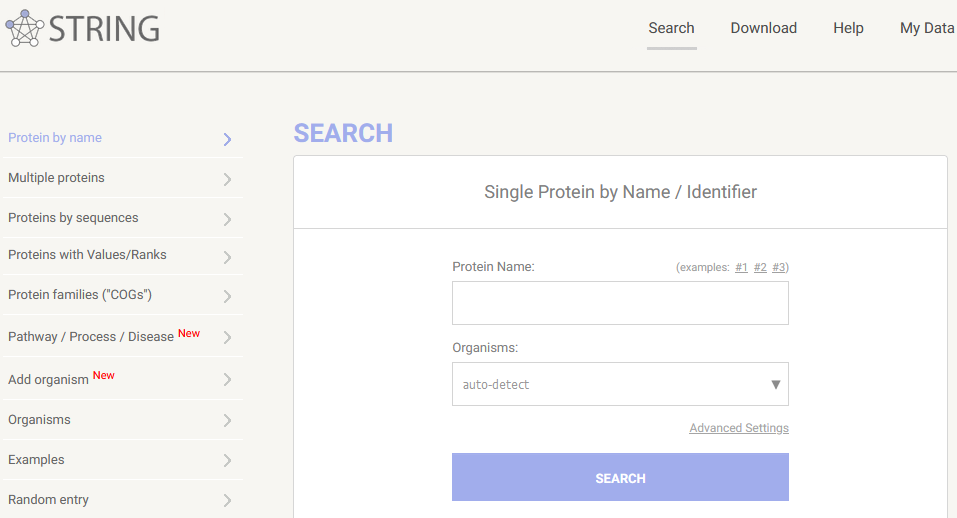
Columns:

1. Column A | Grid ID – This column contains the location on the slide the endpoint data was collected from. This column is not utilized in the script so it can be replaced.
2. Column B | RPPA ID – This column contains the name of the sample/slide/experiment subject the data was collected for. This column is not utilized in the script so it can be replaced.
3. Column C | Tissue type – This column contains the identifier for which experimental group the sample data is for. In this context Group\_1 is the control group and Group\_2 is the experimental/treated group. The spelling for Group\_1 and Group\_2 must be exact for the script unless you are comfortable editing Python scripts.
4. Columns D and forward – These columns contain the numerical values of the experiment samples collected via the RPPA process.

How to get the gene names:

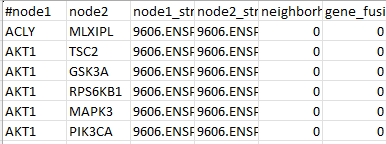
The contents of Rows 2 and greater should be of a numerical value. If a non-numerical value is detected it will be removed and the cell will be imputed by the smallest value in the column of the endpoint and of the group (Group\_1 or Group\_2) that the cell belongs to divided by 2.

The contents of Row 1 should be the gene name of the endpoint/protein. The gene name must be the name used in the STRING database. This gene name can be obtained by querying the STRING database using the name of the endpoint/protein (See image below).



If the RPPA experiments used replicates outliers and errors should be removed before the data is averaged and added to the Endpoint\_data.csv file.

Network\_map.csv (See image below)



Columns:

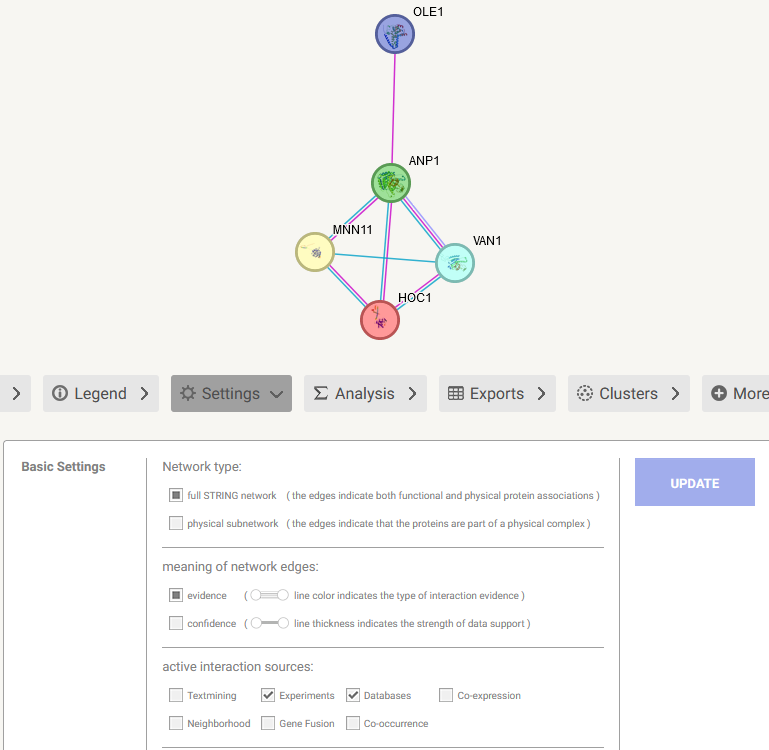
1. Columns A and B – These columns contain the pairs of gene names used to link the genes in the network visualization. These columns are utilized by the script so the spelling of the gene names should match the spelling of the gene/endpoint/protein column labels in the Endpoint\_data.csv file.
2. Columns C and forward – These columns contain numerical values that indicate how the genes are related according to the STRING database. These columns are not utilized by the script.

How to generate the Network\_map.csv file:

* 1. Convert all endpoint/proteins to gene names the STRING database uses.
  2. Use STRING database’s “Multiple proteins” query. (See image below).
     1. Paste the gene names in a list with one gene per line.
     2. Select the correct organism based on which organism your samples were collected sourced from.
     3. After pressing search. A list will appear to have you select which genes you want to use as some genes have different variations.
     4. The next screen will have an interactive network visualization with a legend.
     5. Go to the “Settings” tab below the network visual.
     6. In the “Basic Settings” section and “active interaction sources:” subsection, uncheck all the boxes except **Experiments** and **Databases**. (See image below).
        1. This filters the network so only known protein-protein interactions are shown on the network.
     7. Click the blue “UPDATE” button on the right side.
     8. Once you have the network visual filtered go to the “Exports” tab below the network visual.
     9. Click the “download” button for the “…as short tabular text output” to get a TSV file containing the network interactions. Save this file to your computer.
     10. Open the TSV file with Excel to convert it. Save this file as a “CSV UTF-8 (Comma delimited)” file.

A screenshot of a search

Description automatically generated



How to use:

Instruction on how to use the RPPA Network Analysis EDA (Workflow) can be found below or within the Jupyter Notebook script.

On your computer (Locally):

1. Start the Cytoscape software on your computer.
2. Open the “RPPA-Network-Analysis-EDA.ipynb” Jupyter Notebook file in your preferred Integrated development environment (IDE).
3. Run all the code cells.
   1. This can be done in Visual Studio Code by pressing “Run All”.
   2. Or by manually pressing the “Run”/Play button next to each code cell.
4. As the code cells run, two file explorers will pop-up.
   1. For the first one, select the file that you have formatted to match the Endpoint\_data.csv example file.
   2. For the second one, select the file that you have formatted to match the Network\_map.csv example file.
5. Let the rest of the code cells run.
6. When all the code cells have completed, the network visualization will appear at the bottom of the Jupyter Notebook and in the Cytoscape application.

On Google Colaboratory:

1. Start the Cytoscape software on your computer.
2. Open the “RPPA-Network-Analysis-EDA.ipynb” Jupyter Notebook file in Google Colaboratory.
3. Scroll down so that you can see both text boxes that say “2.B: Select CSV file containing network structure information” and “2.C: Select CSV file containing endpoint data”.
4. Go to the “Runtime” tab in the top left corner.
5. Press “Run all”.
6. Under the code cell below “2.B: Select CSV file containing network structure information” a grey box that says “Browse…” will appear. Click it and select your “Network\_map.csv” file.
7. Under the code cell below “2.C: Select CSV file containing endpoint data” a grey box that says “Browse…” will appear. Click it and select your “Endpoint\_data.csv” file.
8. Let the rest of the code cells run.
9. When all the code cells have completed, the network visualization will appear at the bottom of the Jupyter Notebook and in the Cytoscape application.

**Appendix A**

How to handle two gene variants in the dataset

In the example files, there were two variants of EGFR (EGFR Y1101 and EGFR Y1148). In the event your dataset has two variants of the same gene follow the below steps.

1. Follow the instructions to assemble the Endpoint\_data.csv and Network\_map.csv files as previously described.
   1. On the STRING database website, the plain gene name (Ex: EGFR) should be selected and added to the network.
2. After exporting the TSV file from STRING database and converting it to a CSV file. Open the CSV file and find all occurrences of the specific gene that are listed in columns A and B.
3. Add a variant identifier to the end of each existing entry for the gene (Ex: EGFR Y1101).
4. Copy the rows in both columns A and B where the gene entry exists with its associated gene/protein/endpoint.
5. Paste these copied cells below the last row in the table. (See image below).
6. Replace the variant identifier in these copied cells with a second unique variant identifier (Ex: EGFR Y1148). You do not need to populate the other columns with information. (See image below).

A screenshot of a table

Description automatically generated

**References**

Coarfa, C., Grimm, S. L., Rajapakshe, K., Perera, D., Lu, H. Y., Wang, X., Christensen, K. R., Mo, Q., Edwards, D. P., & Huang, S. (2021). Reverse-Phase Protein Array: Technology, Application, Data Processing, and Integration. *Journal of biomolecular techniques : JBT*, *32*(1), 15–29. <https://doi.org/10.7171/jbt.21-3202-001>

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & team, J. development. (2016). Jupyter Notebooks—A publishing format for reproducible computational workflows. In F. Loizides & B. Scmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas (pp. 87–90). IOS Press. <https://eprints.soton.ac.uk/403913/>

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>

Otasek, D., Morris, J. H., Bouças, J., Pico, A. R., & Demchak, B. (2019). Cytoscape Automation: empowering workflow-based network analysis. *Genome biology*, *20*(1), 185. <https://doi.org/10.1186/s13059-019-1758-4>

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., & Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, *13*(11), 2498–2504. <https://doi.org/10.1101/gr.1239303>

Szklarczyk, D., Kirsch, R., Koutrouli, M., Nastou, K., Mehryary, F., Hachilif, R., Gable, A. L., Fang, T., Doncheva, N. T., Pyysalo, S., Bork, P., Jensen, L. J., & von Mering, C. (2023). The STRING database in 2023: protein-protein association networks and functional enrichment analyses for any sequenced genome of interest. Nucleic acids research, 51(D1), D638–D646. <https://doi.org/10.1093/nar/gkac1000>

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>