Feb 18, 2025 Version 2

Instructions for installation and running code V.2

DOI
https://dx.doi.org/10.17504/protocols.io.81wgbrn4olpk/v2

Vlad Sandulache[1]

[1]Baylor College of Medicine

Vlad Sandulache
Baylor College of Medicine

DOI: https://dx.doi.org/10.17504/protocols.io.81wgbrn4olpk/v2

Protocol Citation: Vlad Sandulache 2025. Instructions for installation and running code. **protocols.io** https://dx.doi.org/10.17504/protocols.io.81wgbrn4olpk/v2Version created by **Vlad Sandulache**

Protocol status: Working
We use this protocol and it's working

Created: February 11, 2025

Last Modified: February 18, 2025

Protocol Integer ID: 119936

Keywords: FPKM values, JupyterLab, Visual basics, code this protocol, protocol detail, use of code, protocol, running code, instructions for installation, instruction, code, installation

Abstract

This protocol details the use of codes for the analysis.

Materials

Requirements:

Install integrated development environment or interpreter for Python. We use both JupyterLab (.ipynb files) and Visual Basics (.py) as explained below, with dependencies import numpy, import pandas, import scipy.stats, and import binom libraries.

Overview of code utilized in analysis

1 Eight python scripts are used in tandem to run the analysis and QC pipeline. These scripts along with the Gene_GC.csv file used as a reference input are all freely available in the Github repository at : https://github.com/Mjfreder/RNA_normalize_qc.
Scripts:
filename1.py
count_merger6.py
normalize45.py
convertGCT.py
generate_plot_csv.py
analyze_plot_csv.py
GC_plotsQC.py
Zeroes.py
Requirements: Install integrated development environment or interpreter for Python. We recommend using Visual Studio Code(v1.96.2) with the Python Development Management (PDM) tool installed inside and using Python v 3.11. Specific python codes and the supporting Gene_GC.csv input files are freely available from the Github repository at https://github.com/Mjfreder/RNA_normalize_qc.

Installation steps: open a terminal in VS code and install PDM tool: pip install pdm

2 Verify installation: **pdm –version**

3 Create a folder called “RNA_process_QC”

4 Copy and paste the 8 python programs inside the RNA_process_QC folder or create new python files within the folder and paste text code from below, being careful to name them as above

5 Open a new terminal within the RNA_process_QC folder using VS code

6 Create a virtual environment inside PDM to install Python 3.11 : **pdm venv create -p python3.11**

7 Create a project by typing the command in the terminal: **pdm init**

8 Follow PDM prompts and enter optional information or hit enter for default selections

9 Add the following dependencies one at a time using the command : **pdm add** “fill in name of dependencies” to make sure each is successfully installed:
pdm add pandas
pdm add numpy
pdm add scipy
pdm add statsmodels
pdm add fpdf
pdm add matplotlib

10 Create a subfolder inside “RNA_process_QC” called “Count_inputs”

11 Paste RSEM output files for data to be normalized together as separate files inside the “Count_inputs” subfolder

12 Run the filename1.py code to generate a csv file where all file name prefixes are extracted from the “Count_inputs” folder and outputted to a text file called “file_names.txt”. To run enter: **pdm run filename1.py**

13 Open the file_names.txt file in Excel. Open a new blank worksheet file in Excel and create two column headers by pasting in “File Id” and “Sample ID” being sure to adhere to the exact same font and punctuation shown here.

14 Copy the text with filenames from file_names.txt and paste underneath the column labeled “File Id”.
In the corresponding row of the adjacent column labeled “Sample ID” type in the short name you want to use to identify samples. Save the file with the exact name “File_ID_sample_ID_map” in Excel but use the dropdown menu to save as a csv file. Final name will be File_ID_sample_ID_map.csv if the show extensions option is activated in Windows Explorer.

15 Use excel or a text editor to open an example file of the RSEM outputs (we recommend a program called Delimit on windows OS for conveniently viewing any type of .txt, .csv, or . tsv file). Ensure that the RSEM output has the following typical headers with exact syntax: “gene_id”, “effective length”, expected_count”. Other columns will be ignored by the program. If the count files from samples has already been merged, see information below and skip the appropriate steps.

16 Take note of what kind of gene id the files are using and whether it is an ensemble ID (e.g., ENSG....), an Entrez ID (i.e., 1021, etc..), or an actual gene symbol in the column.

17 Using gene information listed in an Excel table to create a file called "gene_reference.csv". Open a new worksheet file in Excel. In the first row of the first column type exactly "gene_id". Using the Excel table copy and paste the gene IDs that match those used in the count files. If the count file gene_ids are Ensemble IDs with version numbers after a period, the program still works fine with entering them without the version in the "gene_reference.csv" file because the code automatically edits drops the version characters. If the gene IDs in the count files are Entrez IDs they appear as integers and just copy the integer gene IDs from the primary Excel table. Create a second column in the gene_reference.csv file called"Symbol" and paste the corresponding gene symbols from the primary Excel table that align with values in the gene_id column. Use the save as command to save the Excel file as a ".csv" which automatically is added to the file name "gene_reference" which should be used.

18 In the VS code prompt within the RNA_Process_QC folder type : **pdm run count_merger6.py.**
An output called "combined_data.csv" is created which merges expected counts, effective lengths, and includes columns with the short/custom names for samples extracted from the File_ID_sample_ID_map.csv file.

19 If the merged or meta file of count data from individual files already exists then steps #8-16 can be omitted, but the combined_data.csv must be generated by correctly reformatting the metadata so that the first column is a header called "gene_id" which has the gene IDs listed, and subsequent columns should appear in a triplet pattern corresponding to each sample with headers of "Sample_ID", effective_length", and "expected_count". All rows underneath the "Sample_ID" column should contain the desired Sample ID string text repeated all the way down. Values in the effective length column should correspond to the gene sizes in base pairs, and the expected counts column should contain the numerical value for counts. If the RSEM method was not employed, raw counts can be used in place and fixed gene sizes should be used.

20 To run the normalization program, make sure the gene_reference.csv file created before is still in the directory and is closed (can be accessed by the program). Type and enter: **pdm run normalize45.py.** Many outputs are generated as explained in the readme information below. The output called "final_adjusted_log2_fpkm_uq.csv" contains log2 transformed upper quartile normalized count data that
will be used as in input for the GCplotQC program. The output "step18_log2_to_linear.csv" contains the upper quartile normalized count data transformed back to linear space after substituting 0 for negative log2 values, and will automatically be read as input for the convert to GC code.

21 To create a .gct file in linear space for gene set enrichment analysis, type and enter: **pdm run convertGCT.py**

22 To run the GC QC steps in the pipeline, it is necessary to supply two additional input files. First, create a csv file with the first header spelled and formatted exactly as “Symbol” and a second column with a
header formatted and spelled as “Reference” which should be populated with the average log2 gene expression values from the the cohort to be used for comparison as a standard (e.g. TCGA data or cohort average). For best usage the average gene expression values from the reference cohort should be derived
using the same normalization protocol here, which globally rescales the data to have a median =7 in log2 space. Alternatively, if no standard reference expression data is available, the experimental cohort average values could be used. The file should be saved as “Reference_ave.csv”. The second additional
input file should be named “Gene_GC.csv” which should have the first column header labeled “Symbol” and the second column header labeled “% GC” formatted exactly as it appears here. Rows in the first column should contain gene symbols spelled and formatted the same in the “Reference_ave.csv” and
“Gene_GC.csv” file and should match the gene symbols present in the normalized output file
“final_adjusted_log2_fpkm_uq.csv” and in the gene_reference.csv file used earlier in the pipeline. The first part of the program is a subroutine called “generate_plot_csv.py” that does not need to be run separately but combines information from the “final_adjusted_log2_fpkm_uq.csv”, “Reference_ave.csv”, and “Gene_GC.csv” files to create the needed input file called “Plot.csv”. The “Gene_GC.csv” file supplied with this manuscript can be used and contains the %GC values derived from canonical transcript IDs of known protein coding genes. If customization is desired, the % GC values can be substituted to correspond to
values calculated from any non-canonical transcripts such as the most abundant transcript in a cohort if that is known.

Once the necessary input files are saved in the main folder “ RNA_Process_QC” the program can be run using the command: **pdm run GC_plotsQC.py .** The program generates the needed input file Plots.csv as well as png files for every smoothed curve plotted along side the smoothed curve derived from whatever is chosen as a reference. The area between the reference curve and point along the sample curve that area beneath the reference curve is calculated for each sample and overlayed on the plots along with their sample names. Additionally, a table called “area_below_reference.csv” is generated that lists all sample IDs alongside the area between the sample and the reference curve, for easy reference. An aggregate PDF file of all the graphs is also generated and outputted as “spline_plots.pdf”.

Note: any gene symbols that do not appear in all 3 input files are automatically filtered out by the program and appear in a separate output as a list in the text file called “dropped_genes.txt”.

23 The program to detect true gene dropout (not to be confused with "dropped_genes" from step 22 above) is called “zeroes.py”. To run this program, only one input file is needed but must be manually created. The required input file is called “zeros.csv” and must contain the following column headers in order “Symbol”, “Observed_zeros”, “Total_samples”, and “Probability”. Rows under “Symbol” should contain the gene symbols and “Total samples” should contain the total number of cohort samples being used to calculate how many samples had an observe zero for a gene, which should exclude outliers. For each gene, manually sum the number of cohort samples for which the gene expression value from the output “step4_fpkm_uq.csv” containing counts was exactly equal to 0, before any 0 values are substituted later for all normalized log2 values (e.g., any file including step15a or later). Therefore it is imperative to use files before step15a when negative log2 values are replaced. The number of samples with “true 0” should be entered in the column labeled “Observed_zeros”. The probability values entered in the rows underneath the column header titled “Probability” must be externally calculated from some reference cohort, such as TCGA data. Here it is okay to use raw TCGA count data to identify true zeros, or again if TCGA is run through the pipeline simply use data in the “step4_fpkm_uq.csv” file for calculating probability. The probability formula is equal to the number of reference samples with true zero divided by the total number of samples considered in the reference cohort and should be a fraction ≤ 1.

Once the needed input files are saved in the main folder “RNA_Process_QC” simply type and enter: **pdm run zeroes.py.** This generates an output called “Zeros_with_P_values” that adds columns with raw P-values and Benjamini-Hochberg adjusted P-values.