54gene-olink-qc

Release 0.1

Arjun Biddanda, Karen Perez de Arce, Esha Joshi

Dec 02, 2022

CONTENTS

CHAPTER

ONE

OVERVIEW

This workflow was designed by the Genomics & Data Science team (GDS) at 54gene and is used to analyze data from the Olink Explore platform. This pipeline is designed to be deployed per-batch of Olink Explore data. The workflow is designed to support reproducible bioinformatics, and is written in Snakemake to be platform-agnostic. All dependencies are installed by the pipeline as-needed using conda. Development and testing has been predominantly on AWS' ParallelCluster using Amazon Linux using Snakemake version 7.16.0.

Features:

- Filters excessive Assay Warnings
- Applies outlier filtering using PCA, IQR, and Median differentiation
- Integration of metadata to assess correlation of proteome and metadata variables
- Generate interactive HTML QC reports

To install the latest release, type:

```
git clone https://gitlab.com/data-analysis5/proteomics/54gene-olink-qc.git
```

1.1 Inputs

The pipeline requires the following inputs:

- A CSV in "tall" format from Olink.
- A file with the SHA256 hash of the CSV file from Olink.
- Config file with other pipeline parameters configured (see default config provided in `config/config.yaml`).
- A tab-delimited `metadata.tsv` file with sample linked metadata (NOTE: the sampleIDs must match those in the Olink dataset). One column must contain the sample identifiers and noted in the configuration.

1.2 Outputs

Following a pipeline run, you will be able to generate:

- A post-QC set of Olink proteomic and metadata and a *postqc.yaml* file that can be passed along to further analyses.
- Two HTML based reports: - a report describing the structure of the pre-QC data - a report describing the structure of the post-QC data
- Principal components and assorted covariates for carrying forward in downstream analyses.

See the *Installation*, *Execution*, and *Configuration* for details on setting up and running the pipeline.

CHAPTER

TWO

INSTALLATION

This workflow was designed to use conda for dependency management and Snakemake as the workflow management system, to ensure scalable and reproducible analyses.

2.1 Obtain a copy of this workflow

To obtain a copy of this workflow, clone the repository to your local system where you would like to perform the quality control analysis:

```
git clone https://gitlab.com/data-analysis5/proteomics/54gene-olink-qc.git
```

2.2 Install the runtime environment

If you need to install conda, follow this guide to install Miniconda. We also highly recommend using the mamba for a faster environment creation experience.

Once installed, create the run-time environment using the following command:

```
conda env create -f environment.yaml
# or mamba env create -f environment.yaml
```

Then activate the environment as follows:

```
conda activate 54gene-olink-qc
```

CHAPTER

THREE

CONFIGURATION

The workflow needs to be configured to perform the quality control analyses by creating a set of files that are defined in the `config/config.yaml`. Each of the underlying sections below corresponds to specific files and configuration options that should be added in.

- *Configuation file*
- *Metadata file*
- *Sample linker file*

3.1 Configuation file

A configuration file for this pipeline can be found in `config/config.yaml` and is used for generating and specifying the report from the pipeline.

3.2 Config parameters

Below are descriptions and usage options for the various config parameters specified in `config.yaml`.

Parameter	Required	Description
`name`	Y	Name for the project (prepended to results files)
`olink_data/data`	Y	CSV File with Olink proteomics data
`olink_data/checksum`	Y	File with SHA256 checksum for Olink CSV file
`olink_data/ignoreFile`	N	File with sample IDs to ignore from analysis (default: "")
`olink_data/ignoreRegex`	N	File with regex pattern to ignore samples (default: "")
`olink_data/sdIQR`	Y	Standard deviations for IQR-based outlier filter
`olink_data/sdMedian`	Y	Standard deviations for Median based outlier filter
`olink_data/sdPCA`	Y	Standard deviations for PC1 & PC2 based outlier filter
`olink_data/assayWarnProp`	Y	Proportion of assay warnings required to remove assay
`metadata/filename`	Y	Metadata filename (see below)

3.3 Metadata file

The metadata file can specify both quantitative and qualitative covariates to check for downstream association with axes of proteomic variation (e.g. Age, Sex, BMI).

The file can be structured as as TSV or CSV as below:

```
SampleID    COV1    COV2
A1  0.0334699       0.329964
A10 0.690636        0.422487
A100        0.206265        0.250128
A101        0.636559        0.863622
A102        0.301656        0.0249239
A103        0.364993        0.765381
```

For a clearer example of an example dataset, explore our example data here. Note that checks will be performed to ensure that every single individual has metadata accompanying it. If you have no metadata for an individual, make sure to fill it in with empty or NA values.

3.4 Sample linker file

In both processing of metadata and Olink proteomic data there can be potential shifts in sample ID nomenclature that can be difficult for merging and performing analyses. In instances like these you can provide a simple two column file that will perform the sample renaming in downstream files for you, for example:

```
curID newID
A1 B1
A2 B2
```

This file is largely necessary when the IDs sent to Olink are discrepant with your in-house metadata/phenotypes. The column headers are important to retain.

CHAPTER

FOUR

EXECUTION

4.1 Deploying the pipeline

With the `config.yaml` configured to your run-mode of choice with paths to the necessary configuration and input files, the workflow can be executed on any infrastructure using the `snakemake` command, supplied with further Snakemake command-line arguments (e.g. specifying a profile with `--profile` or `--cluster` to submit jobs to an HPC) depending on your environment. In addition to these options, you must also supply the *–configfile* directive on the command line to point to the configuration that you would like to use.

Test your configuration by performing a dry-run:

```
snakemake --use-conda --configfile config/config.yaml -n
```

Execute the workflow locally via:

```
snakemake --use-conda --configfile config/config.yaml --cores $N
```

Execute the workflow on a cluster using something like:

```
snakemake --use-conda --cluster sbatch --configfile config/config.yaml --jobs 10
```

The pipeline will automatically create a subdirectory for logs in `logs/`.

4.2 Logging

All job-specific logs will be directed to a `logs/` subdirectory in the home analysis directory of the pipeline. This directory is automatically created for you upon execution of the pipeline. For example, if you run the pipeline on a Slurm cluster with default parameters, these log files will follow the naming structure of `snakejob.<name_of_rule>.<job_number>`.

CHAPTER

FIVE

REPORTING RESULTS

The primary outputs of this pipeline for a given Olink dataset are:

- A set of interactive HTML reports for pre and post-QC data
- A yaml file detailing post-QC datasets for carrying forward to analysis

5.1 Reports

The HTML reports generated by this pipeline highlight a variety of results:

1. Principal axes of variation in proteomic differences between individuals
2. Reports of assays and individuals removed and corresponding QC-criteria missed
3. ANOVA between axes of proteomic variation and sample-metadata to highlight experimental sources of variation

Here is an example of an interactive report to get a sense of what to expect.

5.2 Post-QC Configuration

The pipeline will also deposit a yaml file in *results/postqc_yaml/<name>.<date>.postqc.yaml*. The entries in this yaml are:

1. data: the post-QCed data as a TSV.
2. metadata: the post-QC metadata as a TSV.
3. pcs: PCs derived from Olink probes.
4. pcs_sd: Eigenvalues of PCs from proteomic data (e.g. variance explained).
5. postqc_report: HTML report of postqc data.
6. preqc_report: HTML report of preqc data.

CHAPTER

SIX

TESTS

6.1 Unit tests

Unit tests for the python and R modules used in this workflow can be found in `tests/` and run using Pytest and testthat, respectively. Both of which are included in the conda run-time environment for this pipeline.

To run all available python unit tests:

```
conda activate 54gene-olink-qc
pytest -s tests/*.py
```

To run all available R unit tests for a particular script:

```
testthat::test_file("tests/testthat/test-<R script name>.R")
```

Or to run all available R unit tests:

```
testthat::test_dir("tests/testthat")
```

6.2 Pipeline/Integration tests

To test the pipeline, we provide a small test dataset and instructions on how to use this dataset available in a repository here.

To summarize, this dataset contains an Olink NPX (Normalized Protein eXpression) dataset containing 23 samples and 2 controls tested for 92 protein biomarkers on a cardiometabolic panel, and the corresponding metadata for each sample in CSV format. The SHA-256 checksum for the dataset that should be normally found is also included.

6.3 CI/CD

The aforementioned python and R unit tests and integration tests are run as part of the Gitlab Continuous Integration (CI) pipeline for this codebase. You can find the status of the CI pipeline on the main repository page.

CHAPTER

SEVEN

FOR DEVELOPERS

7.1 Contribute back

For developers, if you would like to contribute back to this repository, consider forking the original repo and creating a pull request:

1. Fork the original repo to a personal or lab account.
2. Clone the fork to your local system, to a different place than where you ran your analysis.
3. Copy the modified files from your analysis to the clone of your fork, e.g., `cp -r workflow path/to/fork`. (Make sure to not accidentally copy config file contents or sample sheets. Instead, manually update the example config files if necessary)
4. Commit and push your changes to your fork.
5. Create a pull request against the original repository.

7.2 Obtain updates from upstream

Whenever you want to synchronize your workflow copy with new developments from upstream, do the following.

1. Once, register the upstream repository in your local copy: `git@github.com:snakemake-workflows/54gene-olink.git` or `git remote add -f upstream https://github.com/snakemake-workflows/54gene-olink.git` if you do not have setup ssh keys.
2. Update the upstream version: `git fetch upstream`
3. Create a diff with the current version: `git diff HEAD upstream/master workflow > upstream-changes.diff`
4. Investigate the changes: `vim upstream-changes.diff`
5. Apply the modified diff via: `git apply upstream-changes.diff`
6. Carefully check whether you need to update the config files: `git diff HEAD upstream/master config`. If so, do it manually, and only where necessary, since you would otherwise likely overwrite your settings and samples.

CHAPTER

EIGHT

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

8.1 [0.0.1]

CHAPTER

NINE

REFERENCES

The following software packages are used in this pipeline:

Software	Website	Citation
AWS CLI	https://github.com/aws/aws-cli	
conda	https://docs.conda.io/en/latest/	
Olink-Analyze	https://github.com/Olink-Proteomics/OlinkRPackage	
pandas	https://pandas.pydata.org/	
Python	https://www.python.org/	
R	https://www.r-project.org/	R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna
Snakemake	https://github.com/snakemake/snakemake	doi: 10.12688/f1000research.29032.1
UKB-PPP	https://doi.org/10.1101/2022.06.17.496443	doi: 10.12688/f1000research.29032.1

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search