MPRAflow Documentation

Release 2.1

Max Schubach, Gracie Gordon

Jul 16, 2020

Installation Getting Started

This pipeline processes sequencing data from Massively Parallel Reporter Assays (MPRA) to create count tables for candidate sequences tested in the experiment.

Installation & Getting Started Instructions for the Installation of the program and some examples to get you started.

MPRAflow Workflows An overview of how MPRAflow works and documentation for the MPRAflow sub workflows.

MPRAflow Examples Muliple examples from the literature are listed for every sub workflow in MPRAflow.

Project Information More information on the project, including the changelog, list of contributing authors, and contribution instructions.

CHAPTER 1

Quick Example

```
nextflow run count.nf --dir "fastQFolder" --experiment-file "experiment.csv" --design
↪"design.fa" --association "bc_map.pickle" --m 1 -resume --mpranalyze --outdir
↪"output"
```

CHAPTER 2

Features

Association This utility takes in library association sequencing data (FASTQ) and a design file (FASTA) to assign barcodes to the corresponding elements tested. Functionality includes filtering for quality and coverage of barcodes. This utility must be run before the COUNT utility.

Count This utility processes sequence data (FASTQ) of barcodes from the DNA and RNA fractions of the MPRA experiment and outputs count tables labeled with the element tested and a label provided in the design file. This utility can process multiple replicates and conditions in a parallelized manner. Based on a user specified flag, the pipeline will either output normalized activity for each tested sequence, or will combine the results into a single count matrix compatible with MPRAnalyze.

Saturation mutagenesis This workflow is about getting single variant effect from a target with multiple mutations generated by error-prone PCR. The workflow takes counts (e.g. from the count workflow), combines them with an association file (variants to barcodes) and uses a generalized linear model to to detect single variant effects.

CHAPTER 3

Feedback

The best place to leave feedback, ask questions, and report bugs is the MPRAflow Issue Tracker.

3.1 Indices and tables

- genindex
- search

3.1.1 Quickstart

1. Create an `experiment.csv` in the format below, including the header. *DNA_R1* or *RNA_R1* is name of the gzipped fastq of the forward read of the DNA or RNA from the defined condition and replicate. *DNA_R2* or *RNA_R2* is the corresponding index read with UMIs (excluding sample barcodes) and *DNA_R3* or *RNA_R3* of the reverse read. If you do not have UMIs remove the columns *DNA_R2* and *RNA_R2* or leave them empty.

```
Condition,Replicate,DNA_R1,DNA_R2,DNA_R3,RNA_R1,RNA_R2,RNA_R3
condition1,1,cond1_rep1_DNA_FWD_reads.fastq.gz,cond1_rep1_DNA_IDX_reads.
↪fastq.gz,cond1_rep1_DNA_REV_reads.fastq.gz,cond1_rep1_RNA_FWD_reads.fastq.
↪gz,cond1_rep1_RNA_IDX_reads.fastq.gz,cond1_rep1_RNA_REV_reads.fastq.gz
condition1,2,cond1_rep2_DNA_FWD_reads.fastq.gz,cond1_rep2_DNA_IDX_reads.
↪fastq.gz,cond1_rep2_DNA_REV_reads.fastq.gz,cond1_rep2_RNA_FWD_reads.fastq.
↪gz,cond1_rep2_RNA_IDX_reads.fastq.gz,cond1_rep2_RNA_REV_reads.fastq.gz
condition2,1,cond2_rep1_DNA_FWD_reads.fastq.gz,cond2_rep1_DNA_IDX_reads.
↪fastq.gz,cond2_rep1_DNA_REV_reads.fastq.gz,cond2_rep1_RNA_FWD_reads.fastq.
↪gz,cond2_rep1_RNA_IDX_reads.fastq.gz,cond2_rep1_RNA_REV_reads.fastq.gz
condition2,2,cond2_rep2_DNA_FWD_reads.fastq.gz,cond2_rep2_DNA_IDX_reads.
↪fastq.gz,cond2_rep2_DNA_REV_reads.fastq.gz,cond2_rep2_RNA_FWD_reads.fastq.
↪gz,cond2_rep2_RNA_IDX_reads.fastq.gz,cond2_rep2_RNA_REV_reads.fastq.gz
```

2. If you would like each insert to be colored based on different user-specified categories, such as “positive control”, “negative control”, “shuffled control”, and “putative enhancer”, to assess the overall quality the user can create a ‘label’ tsv in the format below that maps the name to category:

```
insert1_name insert1_label
insert2_name insert2_label
```

The insert names must exactly match the names in the design FASTA file.

3. Run Association if using a design with randomly paired candidate sequences and barcodes

```
conda activate MPRAflow
nextflow run association.nf --fastq-insert "${fastq_prefix}_R1_001.fastq.gz"␣
↪--design "ordered_candidate_sequences.fa" --fastq-bc "${fastq_prefix}_R2_
↪001.fastq.gz"
```

Note: This will run in local mode, please submit this command to your cluster's queue if you would like to run a parallelized version.

4. Run Count

```
conda activate MPRAflow
nextflow run count.nf --dir "bulk_FASTQ_directory" --e "experiment.csv" --
↪design "ordered_candidate_sequences.fa" --association "dictionary_of_
↪candidate_sequences_to_barcodes.p"
```

Be sure that the `experiment.csv` is correct. All fastq files must be in the same folder given by the `--dir` option. If you do not have UMIs please use the option `--no-umi`. Please specify your barcode length and umi-length with `--bc-length` and `--umi-length`.

5. Run saturation mutagenesis

```
conda activate MPRAflow
nextflow run saturationMutagenesis.nf --dir "directory_of_DNA/RNA_counts" --
↪e "satMutexperiment.csv" --assignment "yourSpecificAssignmentFile.variants.
↪txt.gz"
```

Note: The experiment file is different from the count workflow. It should contain the condition, replicate and filename of the counts, like:

```
Condition,Replicate,COUNTS
condition1,1,cond1_1_counts.tsv.gz
condition1,2,cond1_2_counts.tsv.gz
condition1,3,cond1_3_counts.tsv.gz
condition2,1,cond2_1_counts.tsv.gz
condition2,2,cond2_2_counts.tsv.gz
condition2,3,cond2_3_counts.tsv.gz
```

The count files can be generated by the count workflow, are named: `<condition>_<replicate>_counts.tsv.gz` and can be found in the `outs/<condition>/<replicate>` folder. They have to be copied or linked into the `--dir` folder.

3.1.2 Installation

Installation should take less than 10 minutes

System Requirements

CentOS Linux 7 or above

Required packages

```
conda 4.6 or above
```

Download here: https://docs.conda.io/en/latest/miniconda.html

Clone repository

```
git clone https://github.com/shendurelab/MPRAflow.git
```

Set up conda environment

This pipeline uses python2.7 and python3.6 with additional R scripts. Three *.yml* files are provided to create the appropriate environments and is completely handled by nextflow. The whole pipeline is set up to run on a Linux system. The general environment with nextflow located in the home directory called *environment.yml*.

Install the the conda environment. The general conda environment is called MPRAflow.

```
cd MPRAflow
conda env create -n MPRAflow -f environment.yml
```

If you do not have access to the internet, you have to run the previous command on a node with internet. Afterwards you need to start nextflow too (see Steps to run the pipeline). After creation of the second conda environment by nextflow you can cancel it and start it on your internal node. Be aware that folders must have access on all nodes.

Nextflow has problems using conda 4.7 and highet, because the source activate command is replaced by conda activate. If you get error messages after running you can make a symbolik link of the activate command from you bin folder of the conda or miniconda folder to your MPRAflow environment bin folder. E.g. like:

```
ln -s ~/miniconda3/bin/activate ~/miniconda3/envs/MPRAflow/bin/activate
```

Quick test

```
conda activate MPRAflow
nextflow run count.nf --help
nextflow run association.nf --help
nextflow run saturationMutagenesis.nf --help
```

3.1.3 Running MPRAflow on HPC Cluster

Nextflow gives us the opportunity to run MPRAflow in a cluster environment. Right now we split up processes into two main groups: *longtime* and *shorttime*. We can define different job setting for both groups. As you can imagine from the names *longtime* defines processes that takes a while when running. Sometimes several days. *shortime* defines processes that are quicker and are usually done in several minutes.

To enable the submission to your cluster you have to edit the *conf/cluster.config*, allowing each process to be run as a separate *qsub*, *sbatch* or similar command. The config contains example code for SGE, LSF, and SLURM architectures. Please remove the \ for the architecture you would like to use and place \ in front of any architectures not currently in use. A \ in front of all of them runs the pipeline on your local machine (default). If you run MPRAflow on a cluster system make sure be that you export all environment variables. E.g. this can be done with the *-V* option by SGE.

> **Note:** Please consult your cluster's wiki page for cluster specific commands and change clusterOptions = to reflect these specifications. Additionally, for large libraries, more memory can be specified in this location.

3.1.4 Overview

This pipeline processes sequencing data from Massively Parallel Reporter Assays (MPRA) to create count tables for candidate sequences tested in the experiment.

This package contains three utilities:

ASSOCIATION (association.nf)

This utility takes in library association sequencing data (FASTQ) and a design file (FASTA) to assign barcodes to the corresponding elements tested. Functionality includes filtering for quality and coverage of barcodes. This utility must be run before the COUNT utility. See *Association* for more details.

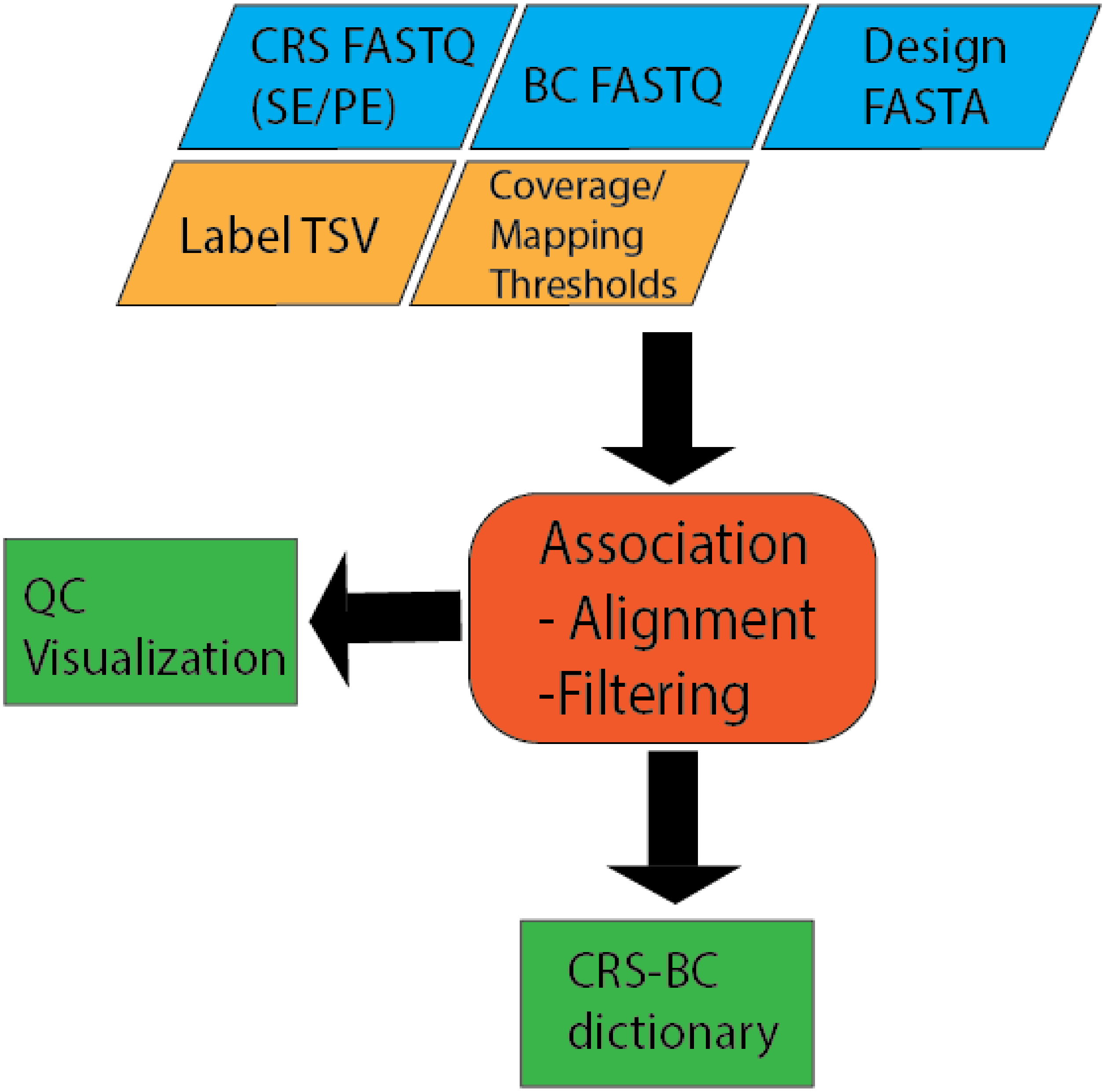
COUNT (count.nf)

This utility processes sequence data (FASTQ) of barcodes from the DNA and RNA fractions of the MPRA experiment and outputs count tables labeled with the element tested and a label provided in the design file. This utility can process multiple replicates and conditions in a parallelized manner. Based on a user specified flag, the pipeline will either output normalized activity for each tested sequence, or will combine the results into a single count matrix compatible with MPRAnalyze. See *Count* for more details.

Saturation mutagenesis (saturationMutagenesis.nf)

This workflow is about getting single variant effect from a target with multiple mutations generated by error-prone PCR. The workflow takes counts (e.g. from the count workflow), combines them with an association file (variants to barcodes) and uses a generalized linear model to to detect single variant effects. See *Saturation mutagenesis* for more details.

3.1.5 Association



Input files

Fastq Files

2-3 Fastq files from library association sequencing –Candidate regulatory sequence (CRS) sequencing, 1 forawrd read and an optional reverse read if paired end sequencing was used –Barcode sequence, 1 read covering the barcode

Design File

Fasta file of of CRS sequences with unique headers describing each tested sequence

Example file:

```
>CRS1
GACGGGAACGTTTGAGCGAGATCGAGGATAGGAGGAGCGGA
>CRS2
GGGCTCTCTTATATTAAGGGGGTGTGTGAACGCTCGCGATT
>CRS3
GGCGCGCTTTTTCGAAGAAACCCGCCGGAGAATATAAGGGA
>CRS4
TTAGACCGCCCTTTACCCCGAGAAAACTCAGCTACACACTC
```

Label File (Optional)

Tab separated file (TSV) of desired labels for each tested sequence

Example file:

```
CRS1	Positive_Control
CRS2	Negative_Control
CRS3	Test
CRS4	Positive_Control
```

association.nf

Options

With `--help` or `--h` you can see the help message.

Mandatory arguments:

--fastq-insert Full path to library association fastq for insert (must be surrounded with quotes)

--fastq-bc Full path to library association fastq for bc (must be surrounded with quotes)

--design Full path to fasta of ordered oligo sequences (must be surrounded with quotes)

--name Name of the association. Files will be named after this.

Optional:

--fastq-insertPE Full path to library association fastq for read2 if the library is paired end (must be surrounded with quotes)

--min-cov minimum coverage of bc to count it (default 3)

--min-frac minimum fraction of bc map to single insert (default 0.5)

--mapq map quality (default 30)

--baseq base quality (default 30)

--cigar require exact match ex: 200M (default none)

--outdir The output directory where the results will be saved and what will be used as a prefix (default outs)

--split Number read entries per fastq chunk for faster processing (default: 2000000)

--labels tsv with the oligo pool fasta and a group label (ex: positive_control) if no labels desired a file will be automatically generated

Processes

Processes run by nextflow in the Association Utility. Some Processes will be run only if certain options used and are marked below.

count_bc or count_bc_nolab (if no label file is provided) Removes any illegal characters (defined by Piccard) in the label file and design file. Counts the number of reads in the fastq file.

create_BWA_ref Creates a BWA reference based on the design file

PE_merge (if paired end fastq files provided) Merges the forward and reverse reads covering the CRS using fastq-join

align_BWA_PE or align_BWA_S (if single end mode) Uses BWA to align the CRS fastq files to the reference created from the Design File. This will be done for each fastq file chunk based on the split option.

collect_chunks merges all bamfiles from each separate alignment

map_element_barcodes Assign barcodes to CRS and filters barcodes by user defined parameters for coverage and mapping percentage

filter_barcodes Visualize results

Output

The output can be found in the folder defined by the option `--outdir`. It is structured in folders of the condition as

Files

count_fastq.txt number of barcode reads

count_merged.txt number of aligned CRS reads

design_rmIllegalChars.fa Design file with illegal characters removed

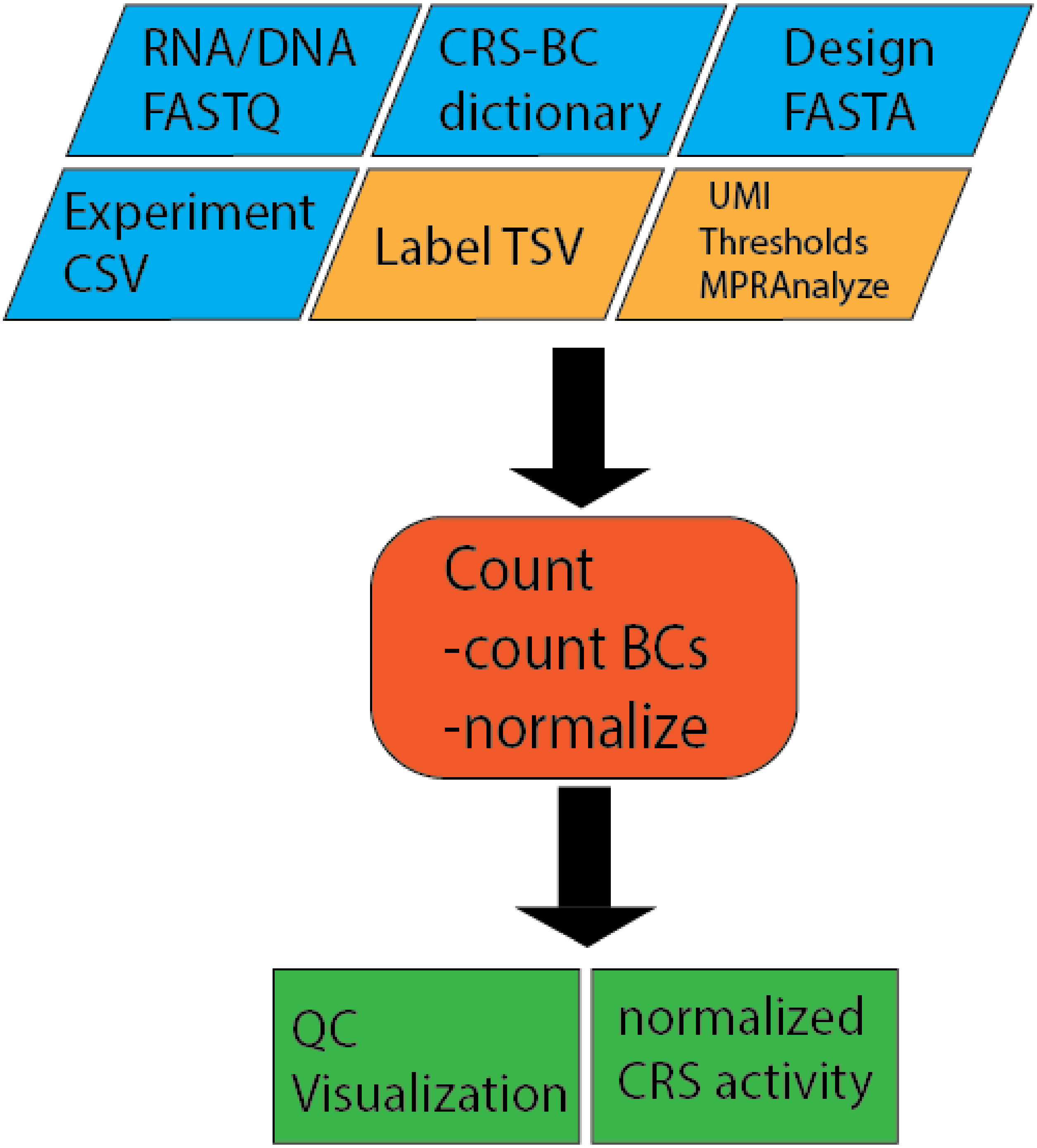
label_rmIllegalChars.txt Label file with illegal characters removed

s_merged.bam sorted bamfile for CRS alignment

${name}_coords_to_barcodes.pickle pickle file containing a python dictionary of CRS/barcode mappings

***.png** Visualization of number of barcodes mapping to enhancers

3.1.6 Count



Input files

Experiment File

Comma separated file (CSV) that assigns all fastq files present in a directory to a condidtion and replicate. Each line represents an experiment, which will all be processed in parallel

```
Condition,Replicate,DNA_BC_F,DNA_UMI,DNA_BC_R,RNA_BC_F,RNA_UMI,RNA_BC_R
Condidtion1,1,C1R1_DNA_barcode_F.fastq.gz,C1R1_DNA_barcode_UMI.fastq.gz,C1R1_DNA_
↪barcode_R.fastq.gz,C1R1_RNA_barcode_F.fastq.gz,C1R1_RNA_barcode_UMI.fastq.gz,C1R1_
↪RNA_barcode_R.fastq.gz
Condidtion1,2,C1R2_DNA_barcode_F.fastq.gz,C1R2_DNA_barcode_UMI.fastq.gz,C1R2_DNA_
↪barcode_R.fastq.gz,C1R2_RNA_barcode_F.fastq.gz,C1R2_RNA_barcode_UMI.fastq.gz,C1R2_
↪RNA_barcode_R.fastq.gz
Condidtion1,3,C1R3_DNA_barcode_F.fastq.gz,C1R3_DNA_barcode_UMI.fastq.gz,C1R3_DNA_
↪barcode_R.fastq.gz,C1R3_RNA_barcode_F.fastq.gz,C1R3_RNA_barcode_UMI.fastq.gz,C1R3_
↪RNA_barcode_R.fastq.gz
Condidtion2,1,C2R1_DNA_barcode_F.fastq.gz,C2R1_DNA_barcode_UMI.fastq.gz,C2R1_DNA_
↪barcode_R.fastq.gz,C2R1_RNA_barcode_F.fastq.gz,C2R1_RNA_barcode_UMI.fastq.gz,C2R1_
↪RNA_barcode_R.fastq.gz
Condidtion2,2,C2R2_DNA_barcode_F.fastq.gz,C2R2_DNA_barcode_UMI.fastq.gz,C2R2_DNA_
↪barcode_R.fastq.gz,C2R2_RNA_barcode_F.fastq.gz,C2R2_RNA_barcode_UMI.fastq.gz,C2R2_
↪RNA_barcode_R.fastq.gz
Condidtion2,3,C2R3_DNA_barcode_F.fastq.gz,C2R3_DNA_barcode_UMI.fastq.gz,C2R3_DNA_
↪barcode_R.fastq.gz,C2R3_RNA_barcode_F.fastq.gz,C2R3_RNA_barcode_UMI.fastq.gz,C2R3_
↪RNA_barcode_R.fastq.gz
```

Design File

Fasta file of of CRS sequences with unique headers describing each tested sequence

Example file:

```
>CRS1
GACGGGAACGTTTGAGCGAGATCGAGGATAGGAGGAGCGGA
>CRS2
GGGCTCTCTTATATTAAGGGGGTGTGTGAACGCTCGCGATT
>CRS3
GGCGCGCTTTTTCGAAGAAACCCGCCGGAGAATATAAGGGA
>CRS4
TTAGACCGCCCTTTACCCCGAGAAAACTCAGCTACACACTC
```

Association Pickle

Python dictionary of CRS to Barcodes

Label File (Optional)

Tab separated file (TSV) of desired labels for each tested sequence

Example file:

```
CRS1	Positive_Control
CRS2	Negative_Control
CRS3	Test
CRS4	Positive_Control
```

Count.nf

Options

With `--help` or `--h` you can see the help message.

Mandatory arguments:

--dir Fasta directory (must be surrounded with quotes)

--e, --experiment-file Experiment csv file

--design Fasta of ordered insert sequences.

--association Pickle dictionary from library association process.

Optional:

--labels tsv with the oligo pool fasta and a group label (ex: positive_control), a single label will be applied if a file is not specified

--outdir The output directory where the results will be saved (default outs)

--bc-length Barcode length (default 15)

--umi-length UMI length (default 10)

--no-umi Use this flag if no UMI is present in the experiment (default with UMI)

--merge_intersect Only retain barcodes in RNA and DNA fraction (TRUE/FALSE, default: FALSE)

--mpranalyze Only generate MPRAnalyze outputs

--thresh minimum number of observed barcodes to retain insert (default 10)

Processes

Processes run by nextflow in the Association Utility. Some Processes will be run only if certain options used and are marked below.

create_BAM or create_BAM_noUMI (if no UMI sequence) creates a bamfile of barcode and UMI sequences

raw_counts creates a table of counts for each barcode (where UMIs, if present, are deduplicated)

filter_counts Remove barcodes that are not the appropriate length

final_counts Record overrepresended UMIs and final count table

dna_rna_merge_counts or dna_rna_mpranalyze_merge Merge RNA/DNA count matrices per barcode

final_merge (MPRAnalyze option only) Merge all DNA/RNA counts into one file

final_label (MPRAnalyze option only) Label the barcodes

generate_mpranalyze_inputs (MPRAnalyze option only) Generate inputs for MPRAnalyze, counts tables and annotation tables for rna/dna

dna_rna_merge Merge each DNA and RNA file label with sequence and insert and normalize

calc_correlations Calculate correlations between Replicates

make_master_tables Create tables of each CRS normalized across replicates

Output

The output can be found in the folder defined by the option `--outdir`. It is structured in folders of the condition as

Files

File tree

```
outdir
  |-Condition
    |-allreps.tsv
    |-average_allreps.tsv
    |-HepG2_1_2_correlation.txt
    |-HepG2_1_2_DNA_pairwise.png
    |-HepG2_1_2_Ratio_pairwise.png
    |-HepG2_1_2_RNA_pairwise.png
    |-HepG2_all_barcodesPerInsert_box.png
    |-HepG2_barcodesPerInsert.png
        |-Reps
            |-HepG2_1_counts.tsv
            |-HepG2_1_counts.tsv.gz
            |-HepG2_1_DNA_counts.tsv
            |-HepG2_1_DNA_raw_counts.tsv.gz
            |-HepG2_1_RNA_filtered_counts.tsv.gz
            |-HepG2_1_DNA_filtered_counts.tsv.gz
            |-HepG2_1_RNA_counts.tsv
            |-HepG2_1_RNA_raw_counts.tsv.gz
```

Files for each Condition

allreps.tsv TSV of normalized DNA and RNA count, ratio, log2ratio, and number of observed barcodes for each condition, replicate, of every CRS

average_allreps.tsv mean ratio, log2 ratio, and observed barcodes per condidition normalized for all replicates

HepG2_1_2_correlation.txt correlation values for a condition and 2 replicates (ie: HepG2 replicate 1 vs replicate 2)

HepG2_1_2_DNA_pairwise.png Correlation plot of DNA counts condition vs two reps (ie: HepG2 replicate 1 vs replicate 2)

HepG2_1_2_Ratio_pairwise.png Correlation plot of normalized log2(RNA/DNA) condition vs two reps (ie: HepG2 replicate 1 vs replicate 2)

HepG2_1_2_RNA_pairwise.png Correlation plot of RNA counts condition vs two reps (ie: HepG2 replicate 1 vs replicate 2)

HepG2_all_barcodesPerInsert_box.png Box plot of each CRS accross replicates for all barcodes in each condition. Colored by the label file.

HepG2_barcodesPerInsert.png Histogram of number of barcodes detected per CRS

HepG2_group_barcodesPerInsert_box.png Boxplot of CRS normalized per insert, grouped by labels

Files for each replicate in each condidtion

HepG2_1_counts.tsv mean ratio, log2 ratio, and observed barcodes per condidition for each replicate

HepG2_1_counts.tsv.gz table of barcodes with DNA counts and RNA counts

HepG2_1_DNA_counts.tsv table of barcodes with DNA counts

HepG2_1_DNA_raw_counts.tsv.gz table of barcodes, UMI, and DNA counts raw

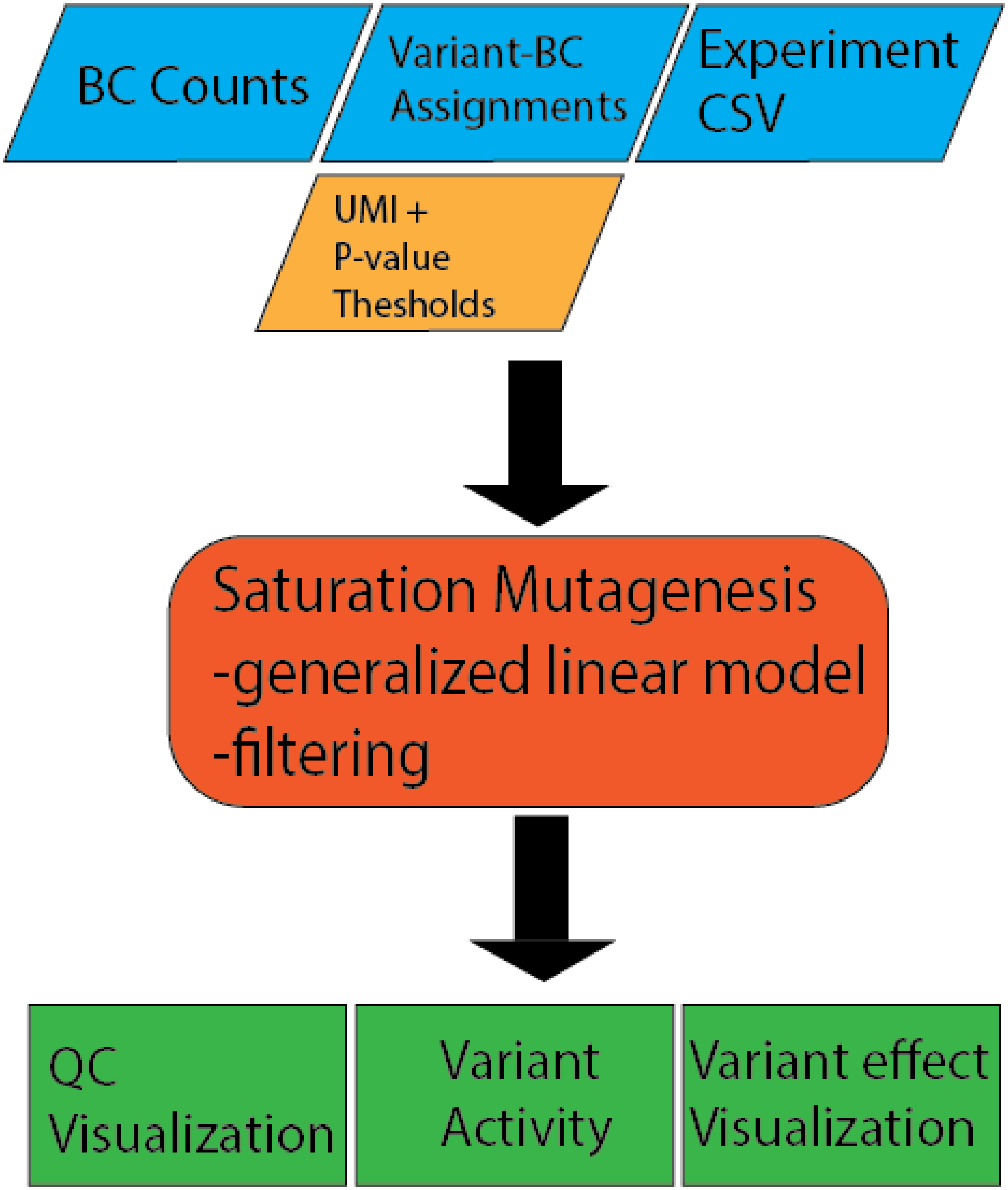
HepG2_1_DNA_filtered_counts.tsv.gz table of barcodes, UMI, and DNA counts raw, filtered for barcodes of correct length

HepG2_1_RNA_counts.tsv table of barcodes with RNA counts

HepG2_1_RNA_raw_counts.tsv.gz table of barcodes, UMI, and RNA counts raw

HepG2_1_RNA_filtered_counts.tsv.gz table of barcodes, UMI, and DNA counts raw, filtered for barcodes of correct length

3.1.7 Saturation mutagenesis



Input files

Variant-BC Assignments

The assignment file contains the variants, assigned to each barcode. It can be gzipped and it is set through the option `--assignment`. Each row starts with one barcode, e.g. `AAAAAACTAATACCA`. The barcode is followed by multiple variants separated by a space. It can also be possible that a barcode has no variant. Then the barcode stands for the complete reference sequence. A variant has the format `identifier:position:ref>alt`, e.g. `TERT:408:G>A`. The *identifier* will be always the same and is use to mark the target of origin. Here the promoter TERT. The *position* is usually the position in the target transcript or any other coordinate system. *ref* is the reference base and *alt* the alternative base to the original target reference. Both can be a single or multiple nucleotide, e.g. `TERT:75:GCCCC>GCCCCC` is a valid variant and refers to an insertion of one C after position 79. For deletions a dot (`.`) can be used in the reference. So `TERT:42:G>.` is the deletion of the base G at position 42. It can also be written as `TERT:41:TG>T`.

Example file:

```
AAAAAACTAATACCA     IRF6:104:A>G IRF6:408:G>A
AAAAAAGCAGGAACA     IRF6:66:A>G IRF6:373:T>A
AAAAAAGCATTCTGT     IRF6:371:G>T IRF6:510:G>A IRF6:560:C>T
AAAAACACTACTGGT     IRF6:326:C>T IRF6:509:T>A
```

Experiment file

Experiment file list the condition(s), replicate(s) and corresponding count files of the analysis. It is comma separated (without spaces in between) and must be set via `--e` or `--experiment-file`.

```
TERT-GBM,1,TERT-GBM_1_counts.tsv.gz
TERT-GBM,2,TERT-GBM_2_counts.tsv.gz
TERT-GBM,3,TERT-GBM_3_counts.tsv.gz
TERT-HEK,1,TERT-HEK_1_counts.tsv.gz
TERT-HEK,2,TERT-HEK_2_counts.tsv.gz
TERT-HEK,3,TERT-HEK_3_counts.tsv.gz
```

DNA/RNA counts per barcode

For each condition and replicate count file (gziped) is needed with the number of DNA and RNA counts per barcode. It is tab separated starting with the barcode, followed by the the DNA and then the RNA counts. Count files can be produced by the count workflow. See *Count*. All count files must be in the `--dir` folder.

Example file:

```
AAAAAAAAAATGATAAGGAA    56      29
AAAAAAAAAATGGGAAGGCG    89      112
AAAAAAAAAATTTGCGTAAA    25      32
AAAAAAAAAATTTGCGTAAT    1       1
AAAAAAAAAATTTGGGGATA    5       26
AAAAAAAAACAATAAGAAAT    21      55
AAAAAAAAACCCAGAATACG    31      39
```

SaturationMutagenesis.nf

Options

With `--help` or `--h` you can see the help message.

Mandatory arguments:

--dir Directory of count files (must be surrounded with quotes).

--assignment Variant assignment file.

--e, --experiment-file Experiment csv file.

Optional:

--outdir The output directory where the results will be saved (default outs).

--thresh Minimum number of observed barcodes to retain variant (default 10).

--pvalue pValue cutoff for significant different variant effects. For variant effect plots only (default 1e-5).

Processes

Processes run by nextflow in the saturation mutagenesis utility.

calc_assign_variantMatrix Creates the variant matrix for the linear model using only single base pair substitutions (for each condition and replicate).

calc_assign_variantMatrixWith1bpDel Creates the variant matrix for the linear model using single base pair substitutions and 1 bp deletions (for each condition and replicate).

fitModel Fit the matrix (variantMatrix and variantMatrixWith1bpDel) using a generalized linear model (for each condition and replicate).

summarizeVariantMatrix

Todo: describe summarizeVariantMatrix

statsWithCoefficient Output of the log2 variant effects from the linear model combined with number of barcodes (for each condition and replicate).

plotCorrelation Plots the correlation between replicates of one condition.

plotStatsWithCoefficient Plots the variant effect plot of the target region using all variants larger than the threshold and the significnace level set by `--p-value` (for each condition and replicate).

fitModelCombined Fit the matrix (variantMatrix and variantMatrixWith1bpDel) using a generalized linear model (for each condition the combined model).

combinedStats

Todo: describe combinedStats

statsWithCoefficientCombined Output of the log2 variant effects from the linear model combined with number of barcodes (for each condition the combined model).

plotStatsWithCoefficientCombined Plots the variant effect plot of the target region using all variants larger than the threshold and the significnace level set by `--p-value` (for each condition the combined model).

Output

The output can be found in the folder defined by the option `--outdir`. It is structured in folders of the condition as

Files

Todo: Describe SatMut output files

3.1.8 Basic association workflow

This example runs the association workflow on 5'/5' WT MRPA data in the HEPG2 cell line from Klein J., Agarwal, V., Keith, A., et al. 2019.

Prerequirements

This example depends on the following data and software:

Installation of MPRAflow

Please install conda, the MPRAflow environment and clone the actual MPRAflow master branch. You will find more help under *Installation*.

Meta Data

It is necessary to get the ordered oligo array so that each enhancer sequence can be labeled in the analysis and to trim any adaptors still in the sequence, in this case we trim off 15bp from the end of each sequence

```
mkdir -p Assoc_Basic/data
cd Assoc_Basic/data
wget ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM4237nnn/GSM4237954/suppl/GSM4237954_
↪9MPRA_elements.fa.gz

zcat GSM4237954_9MPRA_elements.fa.gz |awk '{ count+=1; if (count == 1) { print } else
↪{ print substr($1,1,171)}; if (count == 2) { count=0 } }' > design.fa
```

Reads

There is one set of association sequencing for this data, which contains a forward (CRS-forward), reverse (CRS-reverse), and index (barcode) read for DNA and RNA. These data must be downloaded. All data is publically available on the short read archive (SRA). We will use SRA-toolkit to obtain the data.

Note: You need 10 GB disk space to download the data!

```
conda install sra-tools
cd Assoc_Basic/data
fastq-dump --gzip --split-files SRR10800986
cd ..
```

For large files and unstable internet connection we reccommend the comand *prefetch* from SRA tools before running *fastq-dump*. This command is much smarter in warnings when something went wrong.

```
conda install sra-tools
cd Assoc_Basic/data
prefetch SRR10800986
fastq-dump --gzip --split-files SRR10800986
cd ..
```

Note: Please be sure that all files are downloaded completely without errors! Depending on your internet connection this can take a while. If you just want some data to run MPRAflow you can just limit yourself to one condition and/or just one replicate.

With

```
tree data
```

the folder should look like this:

```
data
```

Here is an overview of the files:

Table 1: HEPG2 association data

Condition	GEO Accession	SRA Accession	SRA Runs
HEPG2-association: HEPG2 library association	GSM4237954	SRX7474872	SRR10800986

MPRAflow

Now we are ready to run MPRAflow and create CRS-barcode mappings.

Run nextflow

Now we have everything at hand to run the count MPRAflow pripeline. Therefore we have to be in the cloned MPRAflow folder. But we will change the working and output directory to the `Assoc_Basic` folder. The MPRAflow count command is:

```
cd <path/to/MPRAflow>/MPRAflow
conda activate MPRAflow
nextflow run association.nf -w <path/to/Basic>/Assoc_Basic/work --fastq-insert "<path/
↪to/Basic>/Assoc_Basic/data/SRR10800986_1.fastq.gz" --fastq-insertPE "<path/to/Basic>
↪/Assoc_Basic/data/SRR10800986_3.fastq.gz" --fastq-bc "<path/to/Basic>/Assoc_Basic/
↪data/SRR10800986_2.fastq.gz" --design "<path/to/Basic>/Assoc_Basic/data/design.fa" -
↪-name assoc_basic
```

Note: Please check your `conf/cluster.config` file if it is correctly configured (e.g. with your SGE cluster commands).

If everything works fine the following 7 processes will run: `count_bc_nolab` `create_BWA_ref`, `PE_merge`, `align_BWA_PE`, `collect_chunks`, `map_element_barcodes`, `filter_barcodes`.

Results

All needed output files will be in the `Assoc_Basic/output` folder.

3.1.9 Basic Count workflow

This example runs the count workflow on 5'/5' WT MRPA data in the HEPG2 cell line from Klein J., Agarwal, V., Keith, A., et al. 2019.

Prerequirements

This example depends on the following data and software:

Installation of MPRAflow

Please install conda, the MPRAflow environment and clone the actual MPRAflow master branch. You will find more help under *Installation*.

Producing an association pickle

This workflow requires a python dictionary of candidate regulatory sequence (CRS) mapped to their barcodes in a pickle format. For this example the file can be generated using *Basic association workflow*.

Reads

There is one condition (HEPG2) with three technical replicates. Each replicate contains a forward (barcode-forward), reverse (barcode-reverse), and index (unique molecular identifier) read for DNA and RNA. These data must be downloaded. All data is publically available on the short read archive (SRA). We will use SRA-toolkit to obtain the data.

Note: You need 9 GB disk space to download the data!

```
conda install sra-tools
mkdir -p Count_Basic/data
cd Count_Basic/data
fastq-dump --gzip --split-files SRR10800881 SRR10800882 SRR10800883 SRR10800884␣
↪SRR10800885 SRR10800886
cd ..
```

For large files and unstable internet connection we reccommend the comand *prefetch* from SRA tools before running *fastq-dump*. This command is much smarter in warnings when something went wrong.

conda install sra-tools cd Count_Basic/data prefetch SRR10800881 SRR10800882 SRR10800883 SRR10800884 SRR10800885 SRR10800886 fastq-dump –gzip –split-files SRR10800986 cd ..

Note: Please be sure that all files are downloaded completely without errors! Depending on your internet connection this can take a while. If you just want some data to run MPRAflow you can just limit yourself to one condition and/or just one replicate.

With

```
tree data
```

the folder should look like this:

```
data
```

Here is an overview of the files:

Table 2: HEPG2 data

Condition	GEO Accession	SRA Accession	SRA Runs
HEPG2-DNA-1: HEPG2 DNA replicate 1	GSM4237863	SRX7474781	SRR10800881
HEPG2-RNA-1: HEPG2 RNA replicate 1	GSM4237864	SRX7474782	SRR10800882
HEPG2-DNA-2: HEPG2 DNA replicate 2	GSM4237865	SRX7474783	SRR10800883
HEPG2-RNA-2: HEPG2 RNA replicate 2	GSM4237866	SRX7474784	SRR10800884
HEPG2-DNA-3: HEPG2 DNA replicate 3	GSM4237867	SRX7474785	SRR10800885
HEPG2-RNA-3: HEPG2 RNA replicate 3	GSM4237868	SRX7474786	SRR10800886

MPRAflow

Now we are close to starting MPRAflow and count the number of barcodes. But before we need to generate an environment csv file to tell nextflow the conditions, replicates and the corresponding reads.

Create experiment.csv

Our experiment file looks exactly like this:

```
Condition,Replicate,DNA_BC_F,DNA_UMI,DNA_BC_R,RNA_BC_F,RNA_UMI,RNA_BC_R
HEPG2,1,SRR10800881_1.fastq.gz,SRR10800881_3.fastq.gz,SRR10800881_2.fastq.gz,
↪SRR10800882_1.fastq.gz,SRR10800882_3.fastq.gz,SRR10800882_2.fastq.gz
HEPG2,2,SRR10800883_1.fastq.gz,SRR10800883_3.fastq.gz,SRR10800883_2.fastq.gz,
↪SRR10800884_1.fastq.gz,SRR10800884_3.fastq.gz,SRR10800884_2.fastq.gz
HEPG2,3,SRR10800885_1.fastq.gz,SRR10800885_3.fastq.gz,SRR10800885_2.fastq.gz,
↪SRR10800886_1.fastq.gz,SRR10800886_3.fastq.gz,SRR10800886_2.fastq.gz
```

Save it into the `Count_Basic/data` folder under `experiment.csv`.

Run nextflow

Now we have everything at hand to run the count MPRAflow pripeline. Therefore we have to be in the cloned MPRAflow folder. But we will change the working and output directory to the `Count_Basic` folder. The MPRAflow count command is:

```
cd <path/to/MPRAflow>/MPRAflow
conda activate MPRAflow
nextflow run count.nf -w <path/to/Basic>/Count_Basic/work --experiment-file "<path/to/
↪Basic>/Count_Basic/data/experiment.csv" --dir "<path/to/Basic>/Count_Basic/data" --
↪outdir "<path/to/Basic>/Count_Basic/output" --design "<path/to/design/fasta/design.
↪fa" --association "<path/to/association/pickle/SRR10800986_filtered_coords_to_
↪barcodes.pickle"
```

Note: Please check your `conf/cluster.config` file if it is correctly configured (e.g. with your SGE cluster commands).

If everything works fine the following 5 processes will run: `create_BAM (make idx)` `raw_counts`, `filter_counts`, `final_counts`, `dna_rna_merge_counts`, `calc_correlations`, `make_master_tables`.

Results

All output files will be in the `Count_Basic/output` folder.

We expect the program to output the following status when complete:

```
start analysis
executor >  sge (32)
[23/09474b] process > create_BAM (make idx)    [100%] 6 of 6 ✓
[0f/4ee034] process > raw_counts (6)           [100%] 6 of 6 ✓
[01/6ac02f] process > filter_counts (6)        [100%] 6 of 6 ✓
[4f/b23748] process > final_counts (6)         [100%] 6 of 6 ✓
[86/4ded79] process > dna_rna_merge_counts (3) [100%] 3 of 3 ✓
[29/0813f8] process > dna_rna_merge (3)        [100%] 3 of 3 ✓
[1d/4e7d56] process > calc_correlations (1)    [100%] 1 of 1 ✓
[9c/4714cb] process > make_master_tables (1)   [100%] 1 of 1 ✓
Completed at: 07-Jan-2020 04:29:07
Duration    : 11h 28m 5s
CPU hours   : 41.5
Succeeded   : 32
```

3.1.10 Count for Saturation Mutagenesis of the TERT promoter

This example runs the count workflow on saturation mutagenesis data of the TERT promoter from Kircher et al. 2019. It will only do the count part. For unraveling single varaint effects please go to the example: *Saturation mutagenesis of the TERT promoter*. The same saturation mutagenesis library was used in four different experiments. We will focus only on the experiments in HEK293T and in glioblastoma SF7996 (GBM) cells (two different conditions).

Prerequirements

This example depends on the following data and software:

Installation of MPRAflow

Please install conda, the MPRAflow environment and clone the actual MPRAflow master branch. You will find more help under *Installation*.

Reads

We have two conditions (HEK293T and GBM cells). Each of them has three technical replicates and one replicate exists of forward, reverse and index reads for DNA and RNA. These data has to be downloaded. All data is public available on the short read archive (SRA). We will use the SRA-tools to download the reads.

Note: You need 16 GB disk space to download the data!

```
conda install sra-tools
mkdir -p Count_TERT/data
cd Count_TERT/data
fastq-dump --gzip --split-files SRR8647059 SRR8647060 SRR8647061 SRR8647062␣
↪SRR8647063 SRR8647064 SRR8647119 SRR8647120 SRR8647121 SRR8647122 SRR8647123␣
↪SRR8647124 SRR8647125 SRR8647126 SRR8647127 SRR8647128 SRR8647129 SRR8647130
cd ..
```

Note: Please be sure that all files are downloaded completely without errors! Depending on your internet connection this can take a while. If you just want some data to run MPRAflow you can just limit yourself to one condition and/or just one replicate.

With

```
tree data
```

the folder should look like this:

```
data
├── SRR8647059_1.fastq.gz
├── SRR8647059_2.fastq.gz
├── SRR8647059_3.fastq.gz
├── SRR8647060_1.fastq.gz
├── SRR8647060_2.fastq.gz
├── SRR8647060_3.fastq.gz
├── SRR8647061_1.fastq.gz
├── SRR8647061_2.fastq.gz
├── SRR8647061_3.fastq.gz
├── SRR8647062_1.fastq.gz
├── SRR8647062_2.fastq.gz
├── SRR8647062_3.fastq.gz
├── SRR8647063_1.fastq.gz
├── SRR8647063_2.fastq.gz
├── SRR8647063_3.fastq.gz
├── SRR8647064_1.fastq.gz
├── SRR8647064_2.fastq.gz
├── SRR8647064_3.fastq.gz
├── SRR8647119_1.fastq.gz
├── SRR8647119_2.fastq.gz
```

(continues on next page)

(continued from previous page)

```
├── SRR8647119_3.fastq.gz
├── SRR8647120_1.fastq.gz
├── SRR8647120_2.fastq.gz
├── SRR8647120_3.fastq.gz
├── SRR8647120_4.fastq.gz
├── SRR8647121_1.fastq.gz
├── SRR8647121_2.fastq.gz
├── SRR8647121_3.fastq.gz
├── SRR8647122_1.fastq.gz
├── SRR8647122_2.fastq.gz
├── SRR8647122_3.fastq.gz
├── SRR8647122_4.fastq.gz
├── SRR8647123_1.fastq.gz
├── SRR8647123_2.fastq.gz
├── SRR8647123_3.fastq.gz
├── SRR8647124_1.fastq.gz
├── SRR8647124_2.fastq.gz
├── SRR8647124_3.fastq.gz
├── SRR8647124_4.fastq.gz
├── SRR8647125_1.fastq.gz
├── SRR8647125_2.fastq.gz
├── SRR8647125_3.fastq.gz
├── SRR8647126_1.fastq.gz
├── SRR8647126_2.fastq.gz
├── SRR8647126_3.fastq.gz
├── SRR8647126_4.fastq.gz
├── SRR8647127_1.fastq.gz
├── SRR8647127_2.fastq.gz
├── SRR8647127_3.fastq.gz
├── SRR8647128_1.fastq.gz
├── SRR8647128_2.fastq.gz
├── SRR8647128_3.fastq.gz
├── SRR8647128_4.fastq.gz
├── SRR8647129_1.fastq.gz
├── SRR8647129_2.fastq.gz
├── SRR8647129_3.fastq.gz
├── SRR8647130_1.fastq.gz
├── SRR8647130_2.fastq.gz
├── SRR8647130_3.fastq.gz
└── SRR8647130_4.fastq.gz

0 directories, 60 files
```

Here is an overview of the files:

Table 3: TERT data

Condition	GEO Accession	SRA Accession	SRA Runs
TERT-GBM-DNA-1: TERT-GBM transfection DNA replicate 1	GSM3604284	SRX5444854	SRR8647059
TERT-GBM-DNA-2: TERT-GBM transfection DNA replicate 2	GSM3604285	SRX5444855	SRR8647060
TERT-GBM-DNA-3: TERT-GBM transfection DNA replicate 3	GSM3604286	SRX5444856	SRR8647061
TERT-GBM-RNA-1: TERT-GBM transfection RNA replicate 1	GSM3604287	SRX5444857	SRR8647062
TERT-GBM-RNA-2: TERT-GBM transfection RNA replicate 2	GSM3604288	SRX5444858	SRR8647063
TERT-GBM-RNA-3: TERT-GBM transfection RNA replicate 3	GSM3604289	SRX5444859	SRR8647064
TERT-HEK-DNA-1: TERT-HEK transfection DNA replicate 1	GSM3604302	SRX5444888	SRR8647119, SRR8647120
TERT-HEK-DNA-2: TERT-HEK transfection DNA replicate 2	GSM3604303	SRX5444889	SRR8647121, SRR8647122
TERT-HEK-DNA-3: TERT-HEK transfection DNA replicate 3	GSM3604304	SRX5444890	SRR8647123, SRR8647124
TERT-HEK-RNA-1: TERT-HEK transfection RNA replicate 1	GSM3604305	SRX5444891	SRR8647125, SRR8647126
TERT-HEK-RNA-2: TERT-HEK transfection RNA replicate 2	GSM3604306	SRX5444892	SRR8647127, SRR8647128
TERT-HEK-RNA-3: TERT-HEK transfection RNA replicate 3	GSM3604307	SRX5444893	SRR8647129, SRR8647130

Note: The runs SRR8647120, SRR8647122, SRR8647124, SRR8647126, SRR8647128, and SRR8647130 have two index reads (forward and reverse). In the publication by Kircher et al. 2019 merging and trimming is used to combine them. For simplification we will discard the reverse index reads: `rm data/*_4.fastq.gz`

Also two different sequencing runs where made in condition TERT-HEK using the same library. Therefore, we have to process both runs together. So we will combine the reads. But in this csase bot sequencing runs have different read lengths (20 and 50 bp). Different read length will make false assumption when merging paired-end reads. Therefore we cut them down to 20 bp.

```
for i in 1 2 3; do
   zcat data/{SRR8647119,SRR8647120}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647119_SRR8647120_$i.fastq.gz;
   zcat data/{SRR8647121,SRR8647122}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647121_SRR8647122_$i.fastq.gz;
   zcat data/{SRR8647123,SRR8647124}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647123_SRR8647124_$i.fastq.gz;
   zcat data/{SRR8647125,SRR8647126}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647125_SRR8647126_$i.fastq.gz;
   zcat data/{SRR8647127,SRR8647128}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647127_SRR8647128_$i.fastq.gz;
   zcat data/{SRR8647129,SRR8647130}_$i.fastq.gz | cut -c 1-20 | gzip -c > data/
↪SRR8647129_SRR8647130_$i.fastq.gz;
```

(continues on next page)

(continued from previous page)

```
done
```

Note: If you combine multiple sequence runs (e.g. you need more reads) you have to combine the reads before. Otherwise barcodes with the same UMI can be count twice. But it is important that all read lengths are the same. The easiest workaround is to cut them down to the minimum length. If you have a different library (but with the same barcodes) you should run the count utility with both runs separately using different conditions. Later you have to combine the final counts of both conditions.

MPRAflow

Now we are close to start MPRAflow and count the number of barcodes. But before we need to generate an `environment.csv` file to tell nextflow the conditions, replicates and the corresponding reads.

Create experiment.csv

Our experiment file looks exactly like this:

```
Condition,Replicate,DNA_BC_F,DNA_UMI,DNA_BC_R,RNA_BC_F,RNA_UMI,RNA_BC_R
TERT-GBM,1,SRR8647059_1.fastq.gz,SRR8647059_3.fastq.gz,SRR8647059_2.fastq.gz,
↪SRR8647062_1.fastq.gz,SRR8647062_3.fastq.gz,SRR8647062_2.fastq.gz
TERT-GBM,2,SRR8647060_1.fastq.gz,SRR8647060_3.fastq.gz,SRR8647060_2.fastq.gz,
↪SRR8647063_1.fastq.gz,SRR8647063_3.fastq.gz,SRR8647063_2.fastq.gz
TERT-GBM,3,SRR8647061_1.fastq.gz,SRR8647061_3.fastq.gz,SRR8647061_2.fastq.gz,
↪SRR8647064_1.fastq.gz,SRR8647064_3.fastq.gz,SRR8647064_2.fastq.gz
TERT-HEK,1,SRR8647119_SRR8647120_1.fastq.gz,SRR8647119_SRR8647120_3.fastq.gz,
↪SRR8647119_SRR8647120_2.fastq.gz,SRR8647125_SRR8647126_1.fastq.gz,SRR8647125_
↪SRR8647126_3.fastq.gz,SRR8647125_SRR8647126_2.fastq.gz
TERT-HEK,2,SRR8647121_SRR8647122_1.fastq.gz,SRR8647121_SRR8647122_3.fastq.gz,
↪SRR8647121_SRR8647122_2.fastq.gz,SRR8647127_SRR8647128_1.fastq.gz,SRR8647127_
↪SRR8647128_3.fastq.gz,SRR8647127_SRR8647128_2.fastq.gz
TERT-HEK,3,SRR8647123_SRR8647124_1.fastq.gz,SRR8647123_SRR8647124_3.fastq.gz,
↪SRR8647123_SRR8647124_2.fastq.gz,SRR8647129_SRR8647130_1.fastq.gz,SRR8647129_
↪SRR8647130_3.fastq.gz,SRR8647129_SRR8647130_2.fastq.gz
```

Save it into the `Count_TERT/data` folder under `experiment.csv`.

Run nextflow

Now we have everything at hand to run the count MPRAflow pipeline. Therefore we have to be in the cloned MPRAflow folder. But we will change the working and output directory to the `Count_TERT` folder. For the TERT example the barcode length is 20 bp and the UMI length 10 bp. The MPRAflow count command is:

```
cd <path/to/MPRAflow>/MPRAflow
conda activate MPRAflow
nextflow run -resume -w <path/to/TERT>/Count_TERT/work  count.nf --experiment-file "
↪<path/to/TERT>/Count_TERT/data/experiment.csv" --dir "<path/to/TERT>/Count_TERT/data
↪" --outdir "<path/to/TERT>/Count_TERT/output" --bc-length 20 --umi-length 10
```

Note: Please check your `conf/cluster.config` file if it is correctly configured (e.g. with your SGE cluster commands).

If everything works fine the following 5 processes will run: `create_BAM (make idx)` `raw_counts`, `filter_counts`, `final_counts`, `dna_rna_merge_counts`.

```
[fe/d8ac14] process > create_BAM (make idx)    [100%] 12 of 12 ✓
[7d/b56129] process > raw_counts (12)          [100%] 12 of 12 ✓
[06/2c938d] process > filter_counts (12)       [100%] 12 of 12 ✓
[2d/ce1afe] process > final_counts (12)        [100%] 12 of 12 ✓
[68/df8db0] process > dna_rna_merge_counts (6) [100%] 6 of 6 ✓
Completed at: 09-Jan-2020 15:38:32
Duration    : 3h 45m 17s
CPU hours   : 21.8
Succeeded   : 54
```

Results

All needed output files will be in the `Count_TERT/output` folder. In this tutorial we are only interested in the counts per barcode, because we can use these outputs in the *Saturation mutagenesis of the TERT promoter* tutorial.

```
cd <path/to/TERT>/Count_TERT
tree output -P "*[123]_counts.tsv.gz"
```

```
output
├── TERT-GBM
│   ├── 1
│   │   └── TERT-GBM_1_counts.tsv.gz
│   ├── 2
│   │   └── TERT-GBM_2_counts.tsv.gz
│   └── 3
│       └── TERT-GBM_3_counts.tsv.gz
└── TERT-HEK
    ├── 1
    │   └── TERT-HEK_1_counts.tsv.gz
    ├── 2
    │   └── TERT-HEK_2_counts.tsv.gz
    └── 3
        └── TERT-HEK_3_counts.tsv.gz

8 directories, 6 files
```

The count files are tab separated and contain the barcode, the number number of unique UMI DNA counts and the umber of unique RNA counts. E.g. this is an example count file:

```
zcat output/TERT-GBM/1/TERT-GBM_1_counts.tsv.gz | head
```

```
AAAAAAAAAAAAAAA 2       76
AAAAAAAAAAAAAAC 1       2
AAAAAAAAAAAAAAT 1       7
AAAAAAAAAAAAAGA 1       5
AAAAAAAAAAAAATA 2       7
AAAAAAAAAAAAATG 2       6
```

(continues on next page)

(continued from previous page)

```
AAAAAAAAAAAAATT 1       2
AAAAAAAAAAAAATGA 3      1
AAAAAAAAAAAAATGG 1      3
AAAAAAAAAAAAATTA 1      2
```

3.1.11 Saturation mutagenesis of the TERT promoter

This example runs the saturation mutagenesis workflow on saturation mutagenesis data of the TERT promoter from Kircher et al. 2019. The same saturation mutagenesis library was used in four different experiments. We will use the experiments in HEK293T and in glioblastoma SF7996 (GBM) cells in this workflow to see differences between the two cell lines (conditions).

Prerequirements

This example depends on the following data and software:

Installation of MPRAflow

Please install conda, the MPRAflow environment and clone the actual MPRAflow master branch. You will find more help under *Installation*.

Assignment file

This file is a tab separated files that assigns variants to barcodes. We will create a new working folder and download the file into it

```
mkdir -p SatMut_TERT/data
cd SatMut_TERT/data
wget http
cd ..
```

Count tables

We need the count tables of the count workflow. Please go to the *Count for Saturation Mutagenesis of the TERT promoter* and run it first. Afterwards copy the count tables into the data folder or use symbolic links:

```
ln -s ../Count_TERT/output/TERT-GBM/*/TERT-GBM_{1,2,3}_counts.tsv.gz data/
ln -s ../Count_TERT/output/TERT-HEK/*/TERT-HEK_{1,2,3}_counts.tsv.gz data/
```

Now the data folder should have the following files:

```
tree data
```

```
data
├── TERT-GBM_1_counts.tsv.gz
├── TERT-GBM_2_counts.tsv.gz
├── TERT-GBM_3_counts.tsv.gz
├── TERT-HEK_1_counts.tsv.gz
```

(continues on next page)

(continued from previous page)

```
├── TERT-HEK_2_counts.tsv.gz
├── TERT-HEK_3_counts.tsv.gz
└── TERT.variants.txt.gz

0 directories, 7 files
```

MPRAflow

Now we are close to start MPRAflow and find out individual variant effects. But before we need to generate an `environment.csv` file to tell nextflow the conditions, replicates and the count files.

Create environment.csv

Our experiment file looks exactly like this:

```
Condition,Replicate,COUNTS
TERT-GBM,1,TERT-GBM_1_counts.tsv.gz
TERT-GBM,2,TERT-GBM_2_counts.tsv.gz
TERT-GBM,3,TERT-GBM_3_counts.tsv.gz
TERT-HEK,1,TERT-HEK_1_counts.tsv.gz
TERT-HEK,2,TERT-HEK_2_counts.tsv.gz
TERT-HEK,3,TERT-HEK_3_counts.tsv.gz
```

Save it into the `SatMut_TERT/data` folder under `experiment.csv`.

Run nextflow

Now we have everything at hand to run the saturation mutagenesis MPRAflow pipeline. Therefore we have to be in the cloned MPRAflow folder. But we will change the working and output directory to the `SatMut_TERT` folder. The MPRAflow saturation mutagenesis command is:

```
cd <path/to/MPRAflow>/MPRAflow
conda activate MPRAflow
nextflow run -resume -w <path/to/TERT>/SatMut_TERT/work  saturationMutagenesis.nf --
↪experiment-file "<path/to/TERT>/SatMut_TERT/data/experiment.csv" --assignment "
↪<path/to/TERT>/SatMut_TERT/data/TERT.variants.txt.gz" --dir "<path/to/TERT>/SatMut_
↪TERT/data" --outdir "<path/to/TERT>/SatMut_TERT/output"
```

Note: Please check your `conf/cluster.config` file if it is correctly configured (e.g. with your SGE cluster commands).

If everything works fine the following 11 processes will run: `calc_assign_variantMatrix` `calc_assign_variantMatrixWith1bpDel`, `fitModel`, `summarizeVariantMatrix`, `statsWithCoefficient`, `plotCorrelation`, `plotStatsWithCoefficient`, `fitModelCombined`, `combinedStats`, `statsWithCoefficientCombined`, and `plotStatsWithCoefficientCombined`.

```
[3c/835d00] process > calc_assign_variantMatrix (1)          [100%] 6 of 6 ✓
[7a/887135] process > calc_assign_variantMatrixWith1bpDel (1) [100%] 6 of 6 ✓
[ca/a90b00] process > fitModel (8)                           [100%] 12 of 12 ✓
[67/3a3e8a] process > summarizeVariantMatrix (12)            [100%] 12 of 12 ✓
```

(continues on next page)

(continued from previous page)

```
[56/846670] process > statsWithCoefficient (12)            [100%] 12 of 12 ✓
[74/466bfb] process > plotCorrelation (1)                  [100%] 12 of 12 ✓
[a5/baf1ef] process > plotStatsWithCoefficient (12)        [100%] 12 of 12 ✓
[ac/d38378] process > fitModelCombined (3)                 [100%] 4 of 4 ✓
[0b/600d8b] process > combinedStats (2)                    [100%] 4 of 4 ✓
[32/80f6a6] process > statsWithCoefficientCombined (2)     [100%] 4 of 4 ✓
[2f/817e76] process > plotStatsWithCoefficientCombined (1) [100%] 4 of 4 ✓
Completed at: 07-Jan-2020 11:31:00
Duration    : 22m 41s
CPU hours   : 1.0
Succeeded   : 88
```

Results

Todo: Add sat mut example

3.1.12 Frequently Asked Questions

If you have more question please write us a ticket on github.

MPRAflow is not able to create a Conda environment If you get a message like:

```
Caused by: java.lang.IllegalStateException: Failed to create Conda environment
command: conda env create --prefix /home/user/MPRAflow/work/conda/mpraflow_py27-
↪a6601743cee3b1029d4f3c810b7ebf02 --file /home/user/MPRAflow/conf/mpraflow_py27.
↪yml`
```

Try to run conda separately using:

```
conda env create --prefix /home/user/MPRAflow/work/conda/mpraflow_py27-
↪a6601743cee3b1029d4f3c810b7ebf02 --file /home/user/MPRAflow/conf/mpraflow_py27.
↪yml
```

Afterwards try MPRAflow again. Please be sure that you are connected to the internet!

Can I use STARR-seq with MPRAflow? No. For more details have a look at this comment.

3.1.13 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/shendurelab/MPRAflow/issues

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the Github issues for bugs. If you want to start working on a bug then please write short message on the issue tracker to prevent duplicate work.

Implement Features

Look through the Github issues for features. If you want to start working on an issue then please write short message on the issue tracker to prevent duplicate work.

Write Documentation

MPRAflow could always use more documentation, even on the web in blog posts, articles, and such.

MPRAflow uses Sphinx for the user documentation (that you are currently reading). See *doc_guidelines* on how the documentation reStructuredText is used. See *doc_setup* on creating a local setup for building the documentation.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/shendurelab/MPRAflow/issues

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Documentation Guidelines

For the documentation, please adhere to the following guidelines:

- Put each sentence on its own line, this makes tracking changes through Git SCM easier.
- Provide hyperlink targets, at least for the first two section levels.
- Use the section structure from below.

```
.. heading_1:

=========
Heading 1
=========



.. heading_2:
```

(continues on next page)

(continued from previous page)

```
---------
Heading 2
---------


.. heading_3:

Heading 3
=========


.. heading_4:

Heading 4
---------


.. heading_5:

Heading 5
~~~~~~~~~


.. heading_6:

Heading 6
:::::::::
```

Documentation Setup

For building the documentation, you have to install the Python program Sphinx. This is best done in a virtual environment. We created a conda environment to work with the actual documentation.

Use the following steps for installing Sphinx and the dependencies for building the MPRAflow documentation:

```
cd MPRAflow/docs
conda env create -f environment.yml -n sphinx
conda activate sphinx
```

Use the following for building the documentation. The first two lines is only required for loading the virtualenv. Afterwards, you can always use `make html` for building.

```
cd MPRAflow/docs
conda activate sphinx
make html  # rebuild for changed files only
make clean && make html  # force rebuild
```

Get Started!

Ready to contribute? First, create your Documentation development setup.

1. Fork the *MPRAflow* repo on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/MPRAflow.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making your changes, make sure that nextflow runs properly For nextflow:

```
nextflow run <your_nextflow_pipeline>
```

For documentation:

```
cd docs
make clean && make html
```

6. Commit your changes and push your branch to GitHub:

```
git add <your_new_file> # or git stage <your_edited_file>
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated.

3.1.14 Authors

in alphabetical order

- Vikram Agarwal
- Tal Ashuach
- Gracie Gordon
- Martin Kircher
- Max Schubach
- Sean Whalen

Contributors

- You name here

3.1.15 History

```
# MPRAflow Changelog

## development

## v2.2

No workflow changes. Only a few fixes and some restructuring of configs. Using␣
↪nextflow version 20.01 now!

### global changes

* nextflow version 20.01 is needed because of multiMap() function
* introducing new config file `conf/global.config` with global variables like the min.
↪ required nextflow version and the actual MPRAflow version.
* moving cluster config to a sepArate file: `conf/cluster.config`. Try to adapt the␣
↪times to the sort and longtime labels. Modify SLURM queue to SLUM not SGE options.
* improved documentation

### saturationMutagenesis.nf

* Bugfix of default out dir. It was not set to `params.outdir = "outs"` so it tries␣
↪to create a folder `null`. Now in `params.outdir`
* removing default `params.version` and `params.nf_required_version`. Now in `conf/
↪global.config`
* Catching cases when barcode/p-value filtering produces 0 variants
* Change update depricated fork method. Now works with nextflow 20.01

### count.nf

* removing default `params.version`, `params.nf_required_version` and `params.outdir`.
↪ Now in `conf/global.config`.

### association.nf

* removing default `params.version`, `params.nf_required_version` and `params.outdir`.
↪ Now in `conf/global.config`.


## v2.1

Initial MPRAflow version for publication.
```

3.1.16 MPRAflow License

MPRAflow is licensed under the Apache 2.0 License:

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.
```

(continues on next page)

(continued from previous page)

```
      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.
```

(continues on next page)

(continued from previous page)

```
   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      (except as stated in this section) patent license to make, have made,
      use, offer to sell, sell, import, and otherwise transfer the Work,
      where such license applies only to those patent claims licensable
      by such Contributor that are necessarily infringed by their
      Contribution(s) alone or by combination of their Contribution(s)
      with the Work to which such Contribution(s) was submitted. If You
      institute patent litigation against any entity (including a
      cross-claim or counterclaim in a lawsuit) alleging that the Work
      or a Contribution incorporated within the Work constitutes direct
      or contributory patent infringement, then any patent licenses
      granted to You under this License for that Work shall terminate
      as of the date such litigation is filed.

   4. Redistribution. You may reproduce and distribute copies of the
      Work or Derivative Works thereof in any medium, with or without
      modifications, and in Source or Object form, provided that You
      meet the following conditions:

      (a) You must give any other recipients of the Work or
          Derivative Works a copy of this License; and

      (b) You must cause any modified files to carry prominent notices
          stating that You changed the files; and

      (c) You must retain, in the Source form of any Derivative Works
          that You distribute, all copyright, patent, trademark, and
          attribution notices from the Source form of the Work,
          excluding those notices that do not pertain to any part of
          the Derivative Works; and

      (d) If the Work includes a "NOTICE" text file as part of its
          distribution, then any Derivative Works that You distribute must
          include a readable copy of the attribution notices contained
          within such NOTICE file, excluding those notices that do not
          pertain to any part of the Derivative Works, in at least one
          of the following places: within a NOTICE text file distributed
          as part of the Derivative Works; within the Source form or
          documentation, if provided along with the Derivative Works; or,
          within a display generated by the Derivative Works, if and
          wherever such third-party notices normally appear. The contents
          of the NOTICE file are for informational purposes only and
          do not modify the License. You may add Your own attribution
          notices within Derivative Works that You distribute, alongside
          or as an addendum to the NOTICE text from the Work, provided
          that such additional attribution notices cannot be construed
          as modifying the License.
```

(continues on next page)

(continued from previous page)

```
   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.
```

(continues on next page)

(continued from previous page)

```
      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "[]"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright [yyyy] [name of copyright owner]

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

3.1.17 Documentation TODO-list

Todo: describe summarizeVariantMatrix

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/mpraflow/checkouts/v2.2/docs/saturation_mutagenes line 95.)

Todo: describe combinedStats

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/mpraflow/checkouts/v2.2/docs/saturation_mutagenes line 105.)

Todo: Describe SatMut output files

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/mpraflow/checkouts/v2.2/docs/saturation_mutagenes line 120.)

Todo: Add sat mut example

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/mpraflow/checkouts/v2.2/docs/saturation_mutagenes line 130.)