snRNAseq scRNAseq Pipeline

Release 0.1

Prashant N M

Jul 19, 2023

INTRODUCTION

1 TODO: 3

2 Changelog: 7

3 Requirements 9

3.1 Salient Features . . . 9
3.2 Executing the Pipeline . . . 9
3.3 Understanding Snakemake workflows . . . 10
3.4 A basic pipeline . . . 10
3.5 Overview of the Pipeline . . . 12
3.6 Wildcard Processing . . . 12
3.7 Selectable Modules . . . 13
3.8 Sub-Snakemake workflows . . . 13
3.9 Demultiplex pooled snRNA seq datasets . . . 14
3.10 Glossary . . . 18

4 Indices and tables 19

Bibliography 21

Index 23

Warning: This documentation is incomplete and is under heavy development!

CHAPTER

ONE

TODO:

- Miscellaneous:
  - Add PICARD option in new_config file.
  - Write down schemas.
  - Add tutorials.
    - pooled snRNA seq
      - single wildcard
      - multiple wildcards
    - scRNA seq
      - single wildcard
      - multiple wildcards
    - Double HTOs
  - Remove dependency on STARsolo as an aligner.
  - For rules that use **genefull_matrices** make input function that take either *Gene* or *GeneFull* dependent on the project.
  - Combine sub-workflows split_bams and split_bams_gt.
    - Search Ranking of readthedocs (using config file for this too).
  - Might incorporate git submodules for repos on git that I use.
  - Add new Picard metrics.
  - Add options in config file to allow adding extra params for every software:
  - For reruns of vireo, provide a way to retain those information in update logs file.
  - Fix demultiplex_helper_funcs.py for double HTOs in function parse_file.
- analyse_vireo:
  - new_config params:
  - snakemake_rules:
  - scripts:
- calico_solo_demux:
  - new_config params:
  - snakemake_rules:

  - scripts:
- demultiplex:
  - new_config params:
  - snakemake_rules:
  - Employ a strategy for final count matrix dir (file dir in config file) for the cases:
    * when both demultiplex software are run simultaneously.
    * When there's an order (try to name each run separately or at least keep the order somewhere mentioned).
  - scripts:
    * when adding calico_solo or vireo include the demultiplex file (file containing demux stats) as input and append to it.
    * For reruns of vireo, provide a way to retain those information in demultiplex info file.
- helper_functions:
  - new_config params:
  - snakemake_rules:
  - scripts:
- identify_swaps:
  - new_config params:
  - snakemake_rules:
  - scripts:
- input_processing:
  - new_config params:
  - snakemake_rules:
  - scripts:
- kite:
  - new_config params:
  - snakemake_rules:
    * Remove run directive
  - scripts:
- pheno_demux3:
  - new_config params:
  - snakemake_rules:
    * Remove run directive
    * Beautify the function get_filt_barcodes.
  - scripts:
- picard_metrics:
  - new_config params:

    - snakemake_rules:
    - scripts:
- produce_targets:
    - new_config params:
    - snakemake_rules:
        * Simplify target functions.
    - scripts:
- resources:
    - scripts:
    - Add conditions for time and mem
- split_bams_gt*:
    - new_config params:
    - snakemake_rules:
        * Remove run directive
        * Input function that removes low mito cells.
    - scripts:
- split_bams*:
    - new_config params:
    - snakemake_rules:
        * Remove run directive
        * Input function that removes low mito cells.
    - scripts:
- STARsolo:
    - new_config params:
    - snakemake_rules:
        * Remove run directive
        * WASP mode
    - scripts:
        * WASP mode

This pipeline intends to not only make complex *preprocessing* workflows easy (e.g. snRNA seq with pooled samples, double HTOs, etc.) but also to facilitate the use of common workflows used for preprocessing by providing *readymade* different combinations of softwares/tools (see *selectable* modules for more options).

It also supports various software/pipeline for scRNA seq pre-processing.

The highlights of the pipeline are:

CHAPTER

TWO

CHANGELOG:

- Changed param name in demultiplex info from *Unique genes* to *gene_ids with an associated gene_name*.
- Added new param in demultiplex info file to add more stats when remove gene IDs without an associated gene name.
- Added an option to run cellSNP without any ref vcfs (1000 Genomes Project vcf is min requirement)
- Now create_wet_lab_info scripts can:
  - Run without a converter file
  - Save donor file along with the wet lab compilation file
  - argparse documented
- Fixed an issue with create_wet_lab_info.py file
- create_wet_lab_info.py file now mirrors actions for donor and multiplex compilations.
- Changed name of the rule demux_samples_calico_solo_STARsolo to demux_samples.
- Changed the *demux_info* parameter to optional (from positional) in demultiplex_no_argp.snkmk's rule that handles adding new demux to a final count matrix.
- Added working argparse to demul_samples_no_argp.py script.
- Changed the name of sub-workflow demultiplex_no_argp.snkmk to demultiplex.snkmk
- Changed the name of sub-workflow demul_samples_no_argp.py to demul_samples.py
- Fix demultiplex_no_argp.snkmk's rule that handles adding new demux to a final count matrix.
- Add an option (in config file) to create h5ads when demultiplexing (demultiplex_no_argp.snkmk) or not (can be used as switch when doing gt checks and finalizing donor assignment).
- Add an option for the rule cellSNP when ref SNPs vcf need not be subsetted further.
- Make the functions similar for demultiplexing with any method.
- Fix issue with reading old wet_lab_info file to update (extension issues).
- Some issue with create_wet_lab_info.py file (it misses to add some lines from certain files - try AMP ones)

CHAPTER

THREE

REQUIREMENTS

This pipeline depends on the following packages/programs:

3.1 Salient Features

3.1.1 Streamlined

The pipeline's heart is the new_config.yaml file, which contains all modifiable properties of the software used in one place. For more info on this file, go *here*.

3.1.2 Maintaining project structure

Another prominent feature of this pipeline is the ability to capture folder structure of the fastqs and provide the ability to organize the outputs of various programs in a similar (with varying structures but using all wildcards) manner. To understand this ability more go *here*.

3.1.3 Easy modification or addition of rules

To modify rules (adding extra params) is easy (add_here the extra params option in STARsolo)

3.1.4 One-select running of popular workflows

Many *pre-selected workflows* are present in the pipeline

3.1.5 Easy resource maintenance

3.1.6 Re-attempts for known issues or errors

3.2 Executing the Pipeline

3.2.1 Installing dependencies

Packages installed through conda

All the packages installed through anaconda3/2018.12 for Python 3.9.5 are described (add_link and file)

Packages installed through pip

All the python packages installed for python version 3.9.5 (and with R version 4.1.0) are described (add_link and file)

3.2.2 Setting up profiles

The info for setting up profiles for different workload managers is mentioned here

3.2.3 Executing Pipeline

This pipeline can be executed by executing (in case of any workflow manager, submitting) the script called `run_snakemake.sh`

```
sh run_snakemake.sh
```

3.3 Understanding Snakemake workflows

To begin with understanding this setup, it is highly advised to go through the basic Snakemake's tutorial.

To surmise the workflow setup, it's easier to follow the bottom-top approach i.e. identify the final files (or a set of) that's required and build the way up to the infput files. This is an example of the *DAG* created by Snakemake.

3.4 A basic pipeline

Let's start to setup the pipeline to run a basic set of softwares required for preprocessing a scRNAseq data i.e. use STARsolo to align our cDNA fastqs and run a set of PICARD tools while retaining all the statistics produced by these tools.

For our example case, we require 2 sets of files from our workflow i.e the outputs created by PICARD programs - CollectGcBiasMetrics and CollectRnaSeqMetrics as they are created independent of each other. Hence, if we were to use only the outputs of one program the other outputs won't be produced.

To finally assimilate all these statistics (alignment statistics and read statistics) we will be runnning another script *run_update_logs.sh* (click here to know how)

3.4.1 Setting up

Firstly, create a list of inputs (check here for different styles of inputs) - we will go with creating text file with the list of fastq files (one line per sample).

How fastq files are arranged

This following pic shows how the fastq files are present in our directory.

Fig. 1: directories

Create fastq_files.txt

This following pic shows the content of the fastq_files.txt.

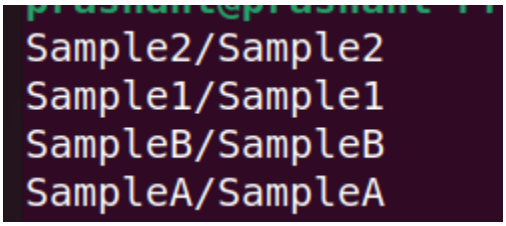


Fig. 2: pools

As one can see it contains one representation for each sample i.e. doesn't separate R1 and R2.

This shows how all the files are present in our directory.

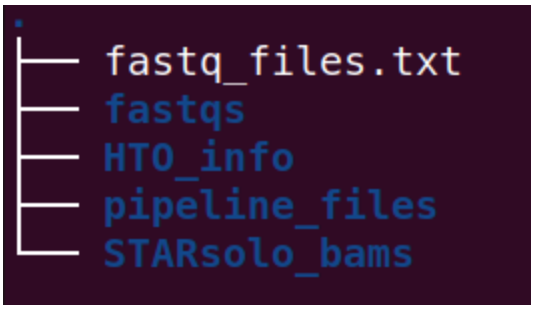


Fig. 3: tree_struct

3.5 Overview of the Pipeline

To set up the pipeline for projects, the following setups need to be performed:

1. Setting up wildcards for the project (see *how*).
2. According to the experiment select a module (module options).
3. As required and wherever possible, setup the folder structures for different programs.
4. Check and modify parameters of the programs to be run (in new_config.yaml file).
    1. (Currently) make sure all required python and R packages are present.
5. Change global variables as required (in Snakefile - check).

3.6 Wildcard Processing

For the purpose of creating wildcards a list of samples to be processed is provided to the pipeline. There are 3 ways to achieve this:

- list of samples/pools (as a folder structure)
- yaml file containing the list of samples/pools
- Directory containing input files

3.6.1 List of samples

This pipeline has many combinations of the aforementioned programs as a built-in set that can be executed using specific keywords.

3.7 Selectable Modules

The following combinations of programs can be run:

where **starsolo** represents STARsolo; **rnaseqmet** and **gcbiasmet** refer to PICARD's CollectRnaSeqMetrics and CollectGcBiasMetrics, respectively while **picard** represents inclusion of both the previously-mentioned programs; **kb_solo** refers to using kallisto, bustools and calico_solo for demultiplexing; **gt_demux** refers to using cellSNP and vireoSNP for genotype based demultiplexing; **split_bams** refers to splitting pooled/multiplexed bams using hashsolo's outputs while **split_bams_gt_demux** refers to splitting pooled/multiplexed bams using vireo's output; **identify_swaps** refers to using qtltools_mbv. The option **multi_vcf** is to provide muiltiple runs (i.e. multiple sets of vcf inputs) for the same sample.

3.7.1 Module description

Table 1: Modules_info

Module Name	Module Info	Sub Worflows Involved
all	module_info (more desc in its own file)	sub_wkfl
all_multi_vcf	module_info (more desc in its own file)	sub_wkfl
starsolo	module_info (more desc in its own file)	sub_wkfl
starsolo_kb_solo	module_info (more desc in its own file)	sub_wkfl
starsolo_gt_demux	module_info (more desc in its own file)	sub_wkfl
starsolo_split_bams	module_info (more desc in its own file)	sub_wkfl
starsolo_split_bams_gt_demux	module_info (more desc in its own file)	sub_wkfl
starsolo_split_bams_gt_demux_multi_vcf	module_info (more desc in its own file)	sub_wkfl
starsolo_gt_demux_multi_vcf	module_info (more desc in its own file)	sub_wkfl
starsolo_cellsnp	module_info (more desc in its own file)	sub_wkfl
starsolo_gt_demux_identify_swaps	module_info (more desc in its own file)	sub_wkfl
starsolo_resolve_swaps_gt_demux	module_info (more desc in its own file)	sub_wkfl

3.8 Sub-Snakemake workflows

This pipeline divides each module into its self-contained individual workflows. These are:

Table 2: Sub_Snakemake_workflows_table

Name of Workflow	Description
resources.snkmk	It contains memory (in MB per thread) and time requirements (in minutes) for each rule.
calico_solo_demux.snkmk	It contains hashsolo rule.
split_bams.snkmk[1]	It contains rules needed to split pooled bams into individual bams dependent on output produced by either hashsolo or vireoSNP using custom scripts.
input_processing.snkmk	It contains rules that collects values for all the wildcards.
STARsolo.snkmk	It contains rules for STARsolo.
produce_targets.snkmk	It contains the **rule all** and the needed functions.
snv_aware_align.snkmk[2]	**This might be removed soon**
kite.snkmk	It contains rules for the kite workflow.
picard_metrics.snkmk	It contains rules for all PICARD metrics (GCBiasMetrics and RNAseqMetrics).
pheno_demux3.snkmk	It contains rules for the cellSNP-vireoSNP pipeline.
split_bams_gt.snkmk	It contains rules needed to split pooled bams into individual bams dependent on output produced by vireo.
demultiplex_no_argp.snkmk	It contains rules for demultiplexing using hashsolo and/or vireoSNP output and create a count matrix file.
identify_swaps.snkmk	It contains rules for identifying swaps using QTLtools-mbv.

`demultiplex_helper_funcs.ret_htos_calico_solo`	Return HTO information and classification for each cell barcode.

Page 14, 1

3.9 Demultiplex pooled snRNA seq datasets

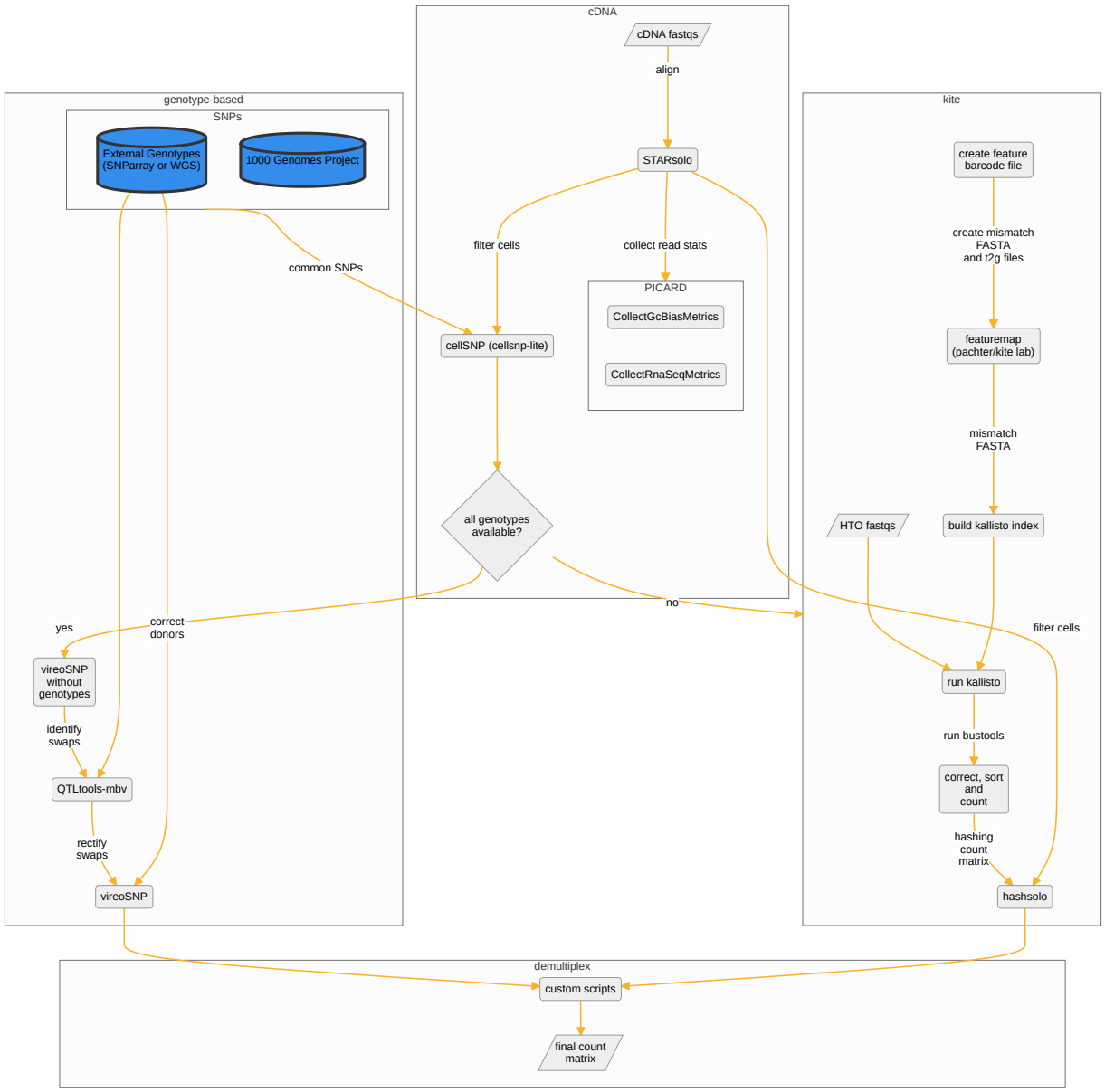
This setup shows one complex workflow that will be simplified and streamlined by this pipeline.

To make it more interesting, this tutorial will annotate individual samples through genotype based demultiplexing (using cellSNP-vireoSNP workflow) as well as HTO based demultiplexing (using kite-hashsolo workflow).

3.9.1 Pipeline overwiew

The pipeline can be visualized as:

[1] Consolidating into one
[2] Not yet implemented


cDNA
cDNA fastqs
align
STARsolo
filter cells
collect read stats
PICARD
CollectGcBiasMetrics
CollectRnaSeqMetrics
genotype-based
SNPs
External Genotypes (SNParray or WGS)
1000 Genomes Project
common SNPs
cellSNP (cellsnp-lite)
all genotypes available?
yes
no
correct donors
vireoSNP without genotypes
identify swaps
QTLtools-mbv
rectify swaps
vireoSNP
kite
create feature barcode file
create mismatch FASTA and t2g files
featuremap (pachter/kite lab)
mismatch FASTA
HTO fastqs
build kallisto index
filter cells
run kallisto
run bustools
correct, sort and count
hashing count matrix
hashsolo
demultiplex
custom scripts
final count matrix

3.9.2 Preparing target files

Firstly, we need to create a list of file structure (derived from our fastq files), which will be used by the rule input_processing(add link here) to read in wildcards

Fastq File Structure

asdasd

3.9.3 Configuration File

To begin with, any utilisation of this pipeline starts with setting up the configuration file *new_config.yaml*

This yaml config file (new_config.yaml) has all relevant options for each rule present in this pipeline. Furthermore, this file has been sectioned, through comments, into separate sub-workflow modules in a way containing rule-specific options/parameters (ocurring in the order of their appearance in the sub-workflow scripts). Typically, there are certain parameters that need not be changed irrespective of the project the pipeline is being used for

Common (project-specific) parameters

The following pictures showcase parameters that are only project-specific.

DAG control and project info params

```
You, 2 days ago | 2 authors (pnm27 and others)
# Common Params----------------------------------------------------------------
# Select pipeline module
last_step: STARsolo_kb_solo

# Limit processing to the files present in
select_fastqs: /sc/arion/projects/psychAD/snakemake_pipeline/fastq_files.txt
wildcards_select: null # null when select_fastqs is not a dir otherwise provide wildcard(s) to extract

# Fastqs info, R1 and R2 suffix assumed to be the same for both cDNA and HTO
# Need to update this for multi_modules
HTO_fastqs_dir: /sc/arion/projects/psychAD/fastq_test_run/
cDNA_fastqs_dir: /sc/arion/projects/psychAD/fastq_test_run/
R1_suffix: .R1.fastq.gz      pnm27, 14 months ago • Modified yaml to more structured …
R2_suffix: .R2.fastq.gz

# Wet lab Info file which contains all details like sample name; set name(if applicable); donor names, HTOs, and HTO_barcodes(if applicable)
# ; etc.
wet_lab_info: /sc/arion/projects/psychAD/snakemake_pipeline/wet_lab_info.tsv
gtf_file: /sc/arion/projects/psychAD/ref_GRCh38/Homo_sapiens.GRCh38.104.gtf  # Same gtf used in STARsolo genome generation
whitelist: /sc/arion/projects/psychAD/10x_vs_barcodes/3M-february-2018.txt  # 10x V3 chemistry
genome_fasta: /sc/arion/projects/psychAD/ref_GRCh38/Homo_sapiens.GRCh38.dna.primary_assembly.fa # Same fasta used in STARsolo genome generation
reg_chr_bed: /sc/arion/projects/psychAD/ref_GRCh38/Homo_sapiens_GRCh38_reg_chr.bed
chr_prefix: null
mito: MT-
gene_info_file: /sc/arion/projects/psychAD/pnm/Hs_allchr_MT.txt

# Standard Filters for gene count matrix files
max_mito_percentage: 5
min_genes_per_cell: 1000
min_cells_per_gene: 10

```

Fig. 4: new_config.yaml (Part 1)

Folder structures

```
31
32  # ENCLOSE WILDCARDS IN {}
33  # HERE, 'id1' IS THE WILDCARD.
34  # Folder structure and prefix for the fastqs (will be mirrored by STARsolo)
35  # presence of HTO confuses snakemake to determine individual or pooled samples
36  # Hence, keep cDNA in all names
37  fold_struct: "Sample_{id1}-cDNA/{id1}-cDNA"
38  # Folder structure for the kallisto-bustools pipeline
39  fold_struct_kb: "Sample_{id1}-HTO/"
40  # Folder structure for demux through the kallisto-bustools pipeline
41  fold_struct_demux: "{id1}/{id1}-HTO"
42  # Folder structure for demux through the cellSNP-vireoSNP pipeline ('vcf_type' wildcard for multi-vcfs input)
43  fold_struct_gt_demux: "Sample_{id1}-cDNA/{vcf_type}/"
44  # Folder structure for adding phe demux when calico_solo run exists ('vcf_type' wildcard for multi-vcfs input)
45  fold_struct_gt_demux2: "Sample-{id1}-cDNA_{vcf_type}"
46  # Folder structure to get cells for cellSNP from
47  fold_struct_filt_bc: "filt_bc_vireo_Sample_{id1}-cDNA"
48  # Folder structures for split bams pipeline
49  fold_struct_bam_split1: "Sample-{id1}-cDNA" # Used for bc_donor hash file, reads_barcodes file, and the split_bams script
50  fold_struct_bam_split2: "Sample_{id1}-cDNA/" # Used for per-barcode and per-donor bam files
51  # Folder structure for per-donor vcf files for STARsolo SNV-Aware pipeline
52  fold_struct_vcf: "{id1}-cDNA/"
53  # Folder structure for per-donor SNV-aware STARsolo bams
54  # fold_struct_snv_aware: "{id1}/"
55
56  fold_struct_gt_demux_redo: "{donor}" # for donor bams and h5ads
57  # fold_struct_swaps_check: "{id1}_{donor}"
58
```

Fig. 5: new_config.yaml (Part 2)

Extra Info (can be removed soon!)

```
58
59  STAR_version: star/2.7.9a
60  featuremap_script: /sc/arion/projects/psychAD/pnm/kite/featuremap/featuremap.py
61  STAR_snv_aware_version: star/2.7.9a
62
63  # If running hashsolo and vireoSNP simultaneously then store in this path
64  demux_count_matrix_dir: /sc/arion/projects/psychAD/final_count_matrix/ # Can be null or a path
65
```

Fig. 6: new_config.yaml (Part 3)

Module selector

last_step: This is the key which needs to be fed one of the *pre-selected modules*

3.9.4 Project-specific changes to rules

3.9.5 Changes to executor script

Finally we have to setup the 2 executor scripts:

..Snakefile:

3.10 Glossary

pooled sample
: A sample using cell hashing, where oligo-tagged antibodies against ubiquitously expressed surface proteins uniquely label cells from distinct samples, which can be subsequently pooled[SZHL+18] and be run as a single experiment.

preprocessing
: To produce gene count matrix file, which can be directly used for an analysis pipeline. For example, in the case of pooled samples this means to retain either one gene count matrix file with each cell attributed to the *donor of origin* or to produce individual gene count matrix file for each donor

DAG
: Directed Acyclic Graph. Click *here* for an example

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[SZHL+18] Marlon Stoeckius, Shiwei Zheng, Brian Houck-Loomis, Stephanie Hao, Bertrand Z Yeung, William M Mauck, 3rd, Peter Smibert, and Rahul Satija. Cell hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol.*, 19(1):224, December 2018.

INDEX

D

DAG, 18

P

pooled sample, 18
preprocessing, 18