Cumulus Documentation

Release 2.0.0

Bo Li, Joshua Gould, and et al.

Mar 15, 2022

Contents

All of our docker images are publicly available on Quay and Docker Hub. Our workflows use Quay as the default Docker registry. Users can use Docker Hub as the Docker registry by entering `cumulusprod` for the workflow input **"docker_registry"**, or enter a custom registry name of their own choice.

If you use Cumulus in your research, please consider citing:

Li, B., Gould, J., Yang, Y. et al. "Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq". *Nat Methods* **17**, 793–798 (2020). https://doi.org/10.1038/s41592-020-0905-x

CHAPTER 1

Release Highlights in Current Stable

1.1 2.0.0 March 14, 2022

Overall:

- Cumulus workflows are now released on Dockstore:
  - Add the tutorial on importing Cumulus workflows to Terra.
  - Archive the legacy versions on Broad Method Registry.
- Add support on multiple platforms via **backend** input: `gcp` for Google Cloud, `aws` for Amazon AWS, `local` for local machine. Enable Google Cloud support by default.
- For Amazon AWS backend, add **awsMaxRetries** input to set the maximum retries allowed for job execution at runtime. By default, use `5`.
- Update the command-line job submission tutorial to work with Altocumulus v2.0.0 or later.
- On Examples:
  - Update gene expression, hashing and CITE-Seq example tutorial.
  - Add tutorial on 10x CellPlex analysis using Cumulus workflows on Cloud.

Workflow-specific:

- Add *STARsolo_create_reference* workflow to build genome references for STARsolo counting. See its documentation for details.
- On *Cellranger* workflow:
  - Add support for 10x Cell Ranger version `6.1.1` and `6.1.2`, and use `6.1.2` by default. See Cell Ranger v6.1 release notes.
  - Add support for 10x Cell Ranger ARC version `2.0.1`, and use it by default. See Cell Ranger ARC v2.0 release notes for the release notes.
  - Upgrade cumulus_feature_barcoding to version `0.7.0` to allow manually set barcode starting position (via input **crispr_barcode_pos**).

- Add support for non 10x CRISPR assays. See the description of `crispr` **DataType** value in this section for details.
- For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per sample listed in the input sample sheet) forms.
- Add **fastq_outputs** to workflow output, which contains *mkfastq* step output folders for samples listed in the input sample sheet.
- Add **count_outputs** to workflow output, which contains *count* step output folderrs for samples listed in the input sample sheet.

• On *Spaceranger* workflow:

- Add support for 10x Space Ranger version `1.3.0` and `1.3.1`, and use `1.3.1` by default. See Space Ranger v1.3 release notes for the release notes.
- For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per library) forms.
- Add output section for the workflow. See here for details.
- Retire old genome references:
  - Keep `GRCh38-2020-A` and `mm10-2020-A`.
  - Retire `GRCh38`, `mm10`, `GRCh38-2020-A-premrna` and `mm10-2020-A-premrna`. Users can still reach out to Cumulus team to ask for URIs to these old references, but they are not provided by default.
- In the description of **ReorientImages** field of input sample sheet, add the information on its valid values.

• On *STARsolo* workflow:

- Add support for STAR version `2.7.9a`, and use it by default. See STAR v2.7.9a release notes for the release notes.
- Reorganize the workflow by exposing more inputs to users.
- Add support on more protocols: 10x multiome, 10x 5' (both SC5P-R2 and SC5P-PE), Slide-Seq and Share-Seq. See *here <./starsolo.html#prepare-a-sample-sheet>* for details.
- Use input **read1_fastq_pattern** and **read2_fastq_pattern** to support fastq files generated by Cell Ranger or SeqWell, as well as Sequence Read Archive (SRA) data.
- For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per library) forms.
- Do not attach filename prefix to output files to avoid the incorrect SJ raw *feature.tsv* symlink error, which would cause the folder delocalization fail. (see discussion with STAR team)
- Add STAR log file to workflow output. This is the *Log.out* file if running STAR locally, which can be used for tracking the process and sharing with STAR team when opening an issue there.
- Retire old genome references:
  - Keep `GRCh38-2020-A`, `mm10-2020-A`, and `GRCh38-and-mm10-2020-A`.
  - Retire old references listed here. Users can still reach out to Cumulus team to ask for URIs to them, but they are not provided by default.

• On *Demultiplexing* workflow:

- Upgrade demuxEM to version `0.1.7` for bug fix.

• On *Cellranger_create_reference* workflow:

  - Add the generated reference file to the workflow output.
  - Bug fix in using input **memory**.
  - Update documentation to suggest only using Cell Ranger version `6.1.1` or later for building reference, as v6.0.1 has issues which leave the job running without terminating.
- On *Cellranger_atac_create_reference* workflow:
  - Add the generated reference file to the workflow output.
- On *Cellranger_vdj_create_reference* workflow:
  - Add the generated reference file to the workflow output.

1.1.1 First Time Running on Terra

Authenticate with Google

If you've done this before you can skip this step - you only need to do this once.

1. Ensure the gcloud CLI is installed on your computer.

   Note: Broad users do not have to install this-they can type:

   ```
   reuse Google-Cloud-SDK
   ```

   to make the Google Cloud tools available.

2. Execute the following command to login to Google Cloud.:

   ```
   gcloud auth login
   ```

3. Copy and paste the link in your unix terminal into your web browser.
4. Enter authorization code in unix terminal.

Create a Terra workspace

1. Create a new Terra workspace by clicking Create New Workspace in Terra

Further reading: Terra tutorials.

1.1.2 Cumulus workflows overview

Cumulus workflows are written in WDL language, and published on Dockstore. Below is an overview of them:

Workflow	First Version	Date Added	Function
Cellranger	0.1.0	2018-07-27	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
Spaceranger	1.2.0	2021-01-19	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
STARsolo	1.2.0	2021-01-19	Run STARsolo to generate gene-count matrices fro FASTQ files.
Demultiplexing	0.3.0	2018-10-24	Run tools (demuxEM, souporcell, or popscle) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
Cumulus	0.1.0	2018-07-27	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
Cellranger_create_reference	0.12.0	2019-12-14	Run Cell Ranger tools to build sc/snRNA-seq references.
Cellranger_atac_create_reference	0.12.0	2019-12-14	Run Cell Ranger tools to build scATAC-seq references.
cellranger_vdj_create_reference	0.12.0	2019-12-14	Run Cell Ranger tools to build single-cell immune profiling references.
STARsolo_create_reference	2.0.0	2022-03-14	Run STAR to build sc/snRNA-seq references for STARsolo count.
Cellranger_atac_aggr	0.13.0	2020-02-07	Run Cell Ranger tools to aggregate scATAC-seq samples.
Smart-Seq2	0.5.0	2018-11-18	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
Smart-Seq2_create_reference	0.12.0	2019-12-14	Generate user-customized genome references for SMART-Seq2 data.

Legacy versions on Broad Method Registry

As Cumulus is now switched to Dockstore for release, we no longer maintain the Cumulus workflows published on Broad Method Registry.

But Terra users can still check out the legacy snapshots listed below for usage.

Stable version - v1.5.1

WDL	Snapshot	Function
cumulus/cellranger_workflow	28	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
cumulus/spaceranger_workflow	3	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
cumulus/star_solo	7	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	18	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	32	Run tools (demuxEM, souporcell, or popscle) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	10	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	5	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	4	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	4	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	10	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
cumulus/smartseq2_create_reference	10	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	43	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.5.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	26	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
cumulus/spaceranger_workflow	3	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
cumulus/star_solo	7	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	18	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	31	Run tools (demuxEM, souporcell, or popscle) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	10	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	5	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	4	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	4	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	10	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
cumulus/smartseq2_create_reference	10	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	43	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.4.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	26	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
cumulus/spaceranger_workflow	3	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
cumulus/star_solo	6	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	18	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	30	Run tools (demuxEM, souporcell, or popscle) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	10	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	5	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	4	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	4	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	10	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
cumulus/smartseq2_create_reference	10	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	41	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.3.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	15	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
cumulus/spaceranger_workflow	1	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
cumulus/star_solo	3	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	18	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	22	Run tools (demuxEM, souporcell, or demuxlet) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	9	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	2	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	2	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	3	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	36	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.2.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	15	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generating count matrix using cellranger count or cellranger-atac count, running cellranger vdj or feature-barcode extraction.
cumulus/spaceranger_workflow	1	Run Space Ranger tools to process spatial transcriptomics data, which includes extracting sequence reads using spaceranger mkfastq, and generating count matrix using spaceranger count.
cumulus/star_solo	3	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	18	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	22	Run tools (demuxEM, souporcell, or demuxlet) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	9	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	2	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	2	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	3	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files.
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	35	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.1.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	14	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/star_solo	3	Run STARsolo to generate gene-count matrices fro FASTQ files.
cumulus/count	16	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	21	Run tools (demuxEM, souporcell, or demuxlet) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	9	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	2	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	2	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	3	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	34	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.

Stable version - v1.0.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	12	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/count	14	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/demultiplexing	20	Run tools (demuxEM, souporcell, or demuxlet) for cell-hashing/nucleus-hashing/genetic-pooling analysis.
cumulus/cellranger_create_reference	9	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	2	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	2	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	3	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	31	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_hashing_cite_seq	10	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.15.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	10	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/count	14	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/cellranger_create_reference	8	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	2	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	2	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	2	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	24	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	16	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	10	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.14.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	8	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/count	11	Run alternative tools (STARsolo, Optimus, Salmon alevin, or Kallisto BUStools) to generate gene-count matrices from FASTQ files.
cumulus/cellranger_create_reference	6	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	1	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	1	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	1	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	7	Run HISAT2/STAR/Bowtie2-RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	8	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	16	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	10	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	8	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.13.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	7	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/cellranger_create_reference	1	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_aggr	1	Run Cell Ranger tools to aggregate scATAC-seq samples.
cumulus/cellranger_atac_create_reference	1	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	1	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	5	Run Bowtie2 and RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	4	Generate user-customized genome references for SMART-Seq2 data.
cumulus/cumulus	14	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	9	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	7	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.12.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	6	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/cellranger_create_reference	1	Run Cell Ranger tools to build sc/snRNA-seq references.
cumulus/cellranger_atac_create_reference	1	Run Cell Ranger tools to build scATAC-seq references.
cumulus/cellranger_vdj_create_reference	1	Run Cell Ranger tools to build single-cell immune profiling references.
cumulus/smartseq2	5	Run Bowtie2 and RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/smartseq2_create_reference	4	Generate user-customized genome references for SMART-Seq2 workflow.
cumulus/cumulus	11	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	8	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	6	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.11.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	4	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/smartseq2	3	Run Bowtie2 and RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/cumulus	8	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	5	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	5	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - v0.10.0

WDL	Snapshot	Function
cumulus/cellranger_workflow	3	Run Cell Ranger tools, which include extracting sequence reads using cellranger mkfastq or cellranger-atac mkfastq, generate count matrix using cellranger count or cellranger-atac count, run cellranger vdj or feature-barcode extraction
cumulus/smartseq2	3	Run Bowtie2 and RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
cumulus/cumulus	7	Run cumulus analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering, visualization, differential expression analysis, cell type annotation, etc.
cumulus/cumulus_subcluster	4	Run subcluster analysis using cumulus
cumulus/cumulus_hashing_cite_seq	4	Run cumulus for cell-hashing/nucleus-hashing/CITE-Seq analysis

Stable version - HTAPP v2

WDL	Snapshot	Function
regev/cellranger_mkfastq_count	45	Run Cell Ranger to extract FASTQ files and generate gene-count matrices for 10x genomics data
scCloud/smartseq2	5	Run Bowtie2 and RSEM to generate gene-count matrices for SMART-Seq2 data from FASTQ files
scCloud/scCloud	14	Run scCloud analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering and more
scCloud/scCloud_subcluster	9	Run subcluster analysis using scCloud
scCloud/scCloud_hashing_cite_seq	9	Run scCloud for cell-hashing/nucleus-hashing/CITE-Seq analysis

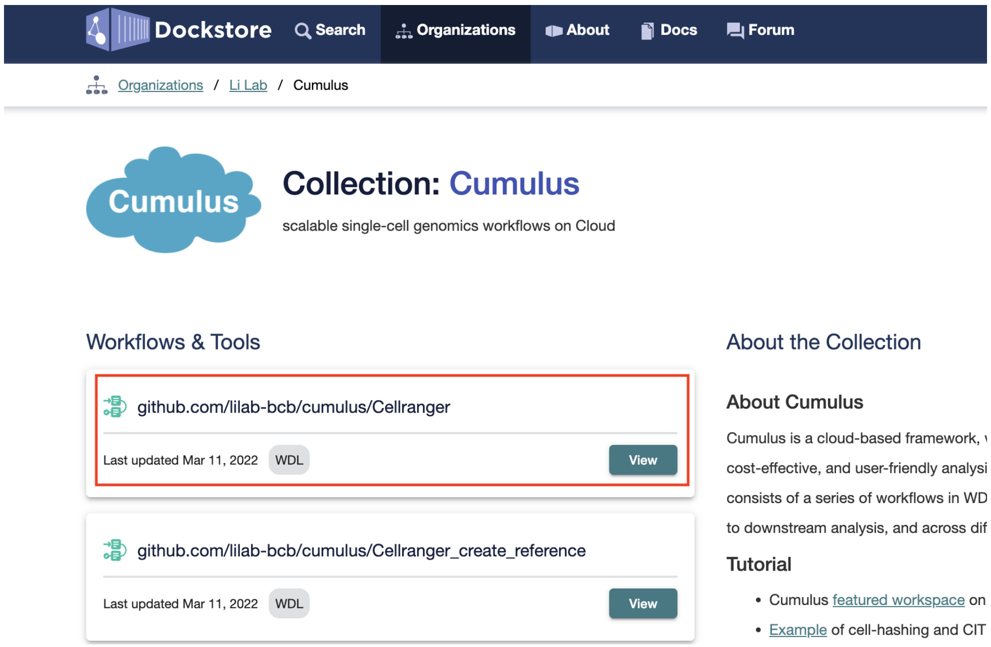
Stable version - HTAPP v1

WDL	Snapshot	Function
regev/cellranger_mkfastq_count	39	Run Cell Ranger to extract FASTQ files and generate gene-count matrices for 10x genomics data
scCloud/scCloud	3	Run scCloud analysis module for variable gene selection, batch correction, PCA, diffusion map, clustering and more

1.1.3 Import workflows to Terra

Cumulus workflows are hosted on Dockstore under the organization of *Li Lab*. For illustration, we'll use *Cellranger* workflow to show how to import Cumulus workflows to your Terra workspace.

1. Select *Cellranger* workflow from Cumulus workflow collection by clicking its "View" button:



Organizations / Li Lab / Cumulus

Collection: Cumulus

scalable single-cell genomics workflows on Cloud

Workflows & Tools

github.com/lilab-bcb/cumulus/Cellranger

Last updated Mar 11, 2022 WDL View

github.com/lilab-bcb/cumulus/Cellranger_create_reference

Last updated Mar 11, 2022 WDL View

About the Collection

About Cumulus

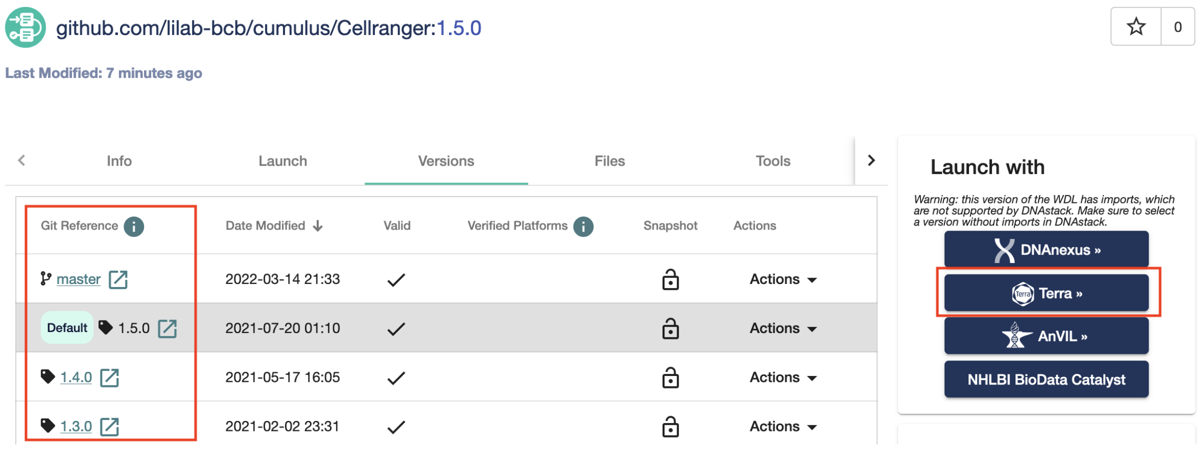
Cumulus is a cloud-based framework, v
cost-effective, and user-friendly analysi
consists of a series of workflows in WD
to downstream analysis, and across dif

Tutorial

- Cumulus featured workspace on
- Example of cell-hashing and CIT

Notice that all Cumulus workflows have `github.com/lilab-bcb/cumulus/` prefix, which indicates they are imported from Cumulus GitHub repo to Dockstore.

2. In the workflow page, by switching to "Versions" tab, you can view all the available versions of *Cellranger* workflow, where the default version is on the top:



github.com/lilab-bcb/cumulus/Cellranger:1.5.0

Last Modified: 7 minutes ago

Git Reference	Date Modified	Valid	Verified Platforms	Snapshot	Actions
master	2022-03-14 21:33	✓		🔓	Actions
Default 1.5.0	2021-07-20 01:10	✓		🔓	Actions
1.4.0	2021-05-17 16:05	✓		🔓	Actions
1.3.0	2021-02-02 23:31	✓		🔓	Actions

Launch with

Warning: this version of the WDL has imports, which are not supported by DNAstack. Make sure to select a version without imports in DNAstack.

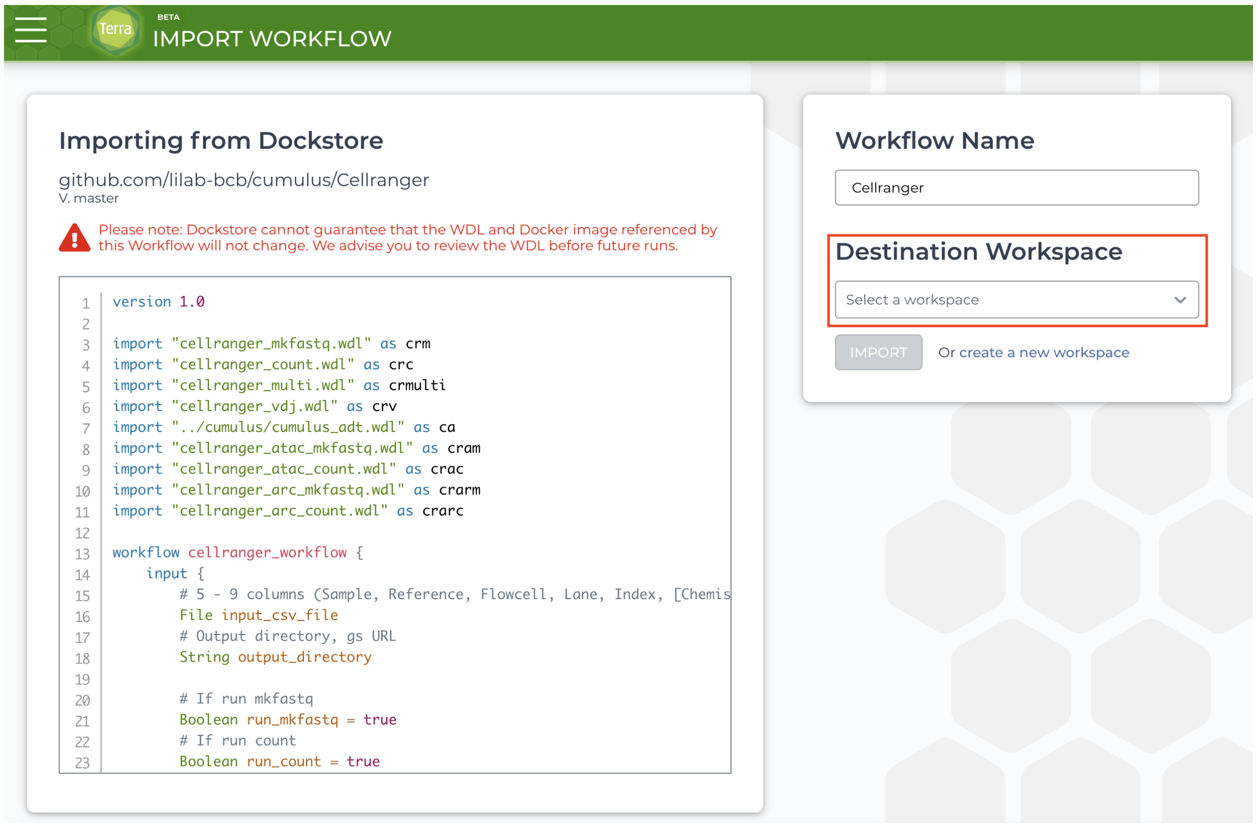
DNAnexus »
Terra »
AnVIL »
NHLBI BioData Catalyst

To change to a non-default version, simply clicking the version name in "Git Reference" column. After that, click "Terra" button on the right panel.

Note: The **master** version refers to the development branch of Cumulus workflows, which is always under rapid change.

For stable usage, please always refer to a released version.

3. You'll be asked to log in to Terra if not. Then you can see the following page:



Importing from Dockstore

github.com/lilab-bcb/cumulus/Cellranger
V. master

Please note: Dockstore cannot guarantee that the WDL and Docker image referenced by this Workflow will not change. We advise you to review the WDL before future runs.

```
version 1.0

import "cellranger_mkfastq.wdl" as crm
import "cellranger_count.wdl" as crc
import "cellranger_multi.wdl" as crmulti
import "cellranger_vdj.wdl" as crv
import "../cumulus/cumulus_adt.wdl" as ca
import "cellranger_atac_mkfastq.wdl" as cram
import "cellranger_atac_count.wdl" as crac
import "cellranger_arc_mkfastq.wdl" as crarm
import "cellranger_arc_count.wdl" as crarc

workflow cellranger_workflow {
    input {
        # 5 - 9 columns (Sample, Reference, Flowcell, Lane, Index, [Chemis
        File input_csv_file
        # Output directory, gs URL
        String output_directory

        # If run mkfastq
        Boolean run_mkfastq = true
        # If run count
        Boolean run_count = true
```

Workflow Name

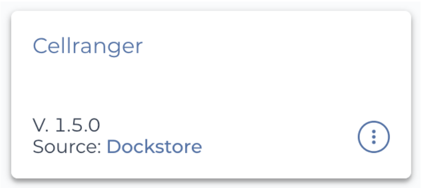
Cellranger

Destination Workspace

Select a workspace

In "Destination Workspace" drop-down menu on the right panel, you can select the target Terra workspace to import *CellRanger* workflow. Optionally, you can even rename the workflow in "Workflow Name" field. When everything is done, click "IMPORT" button below to finish.

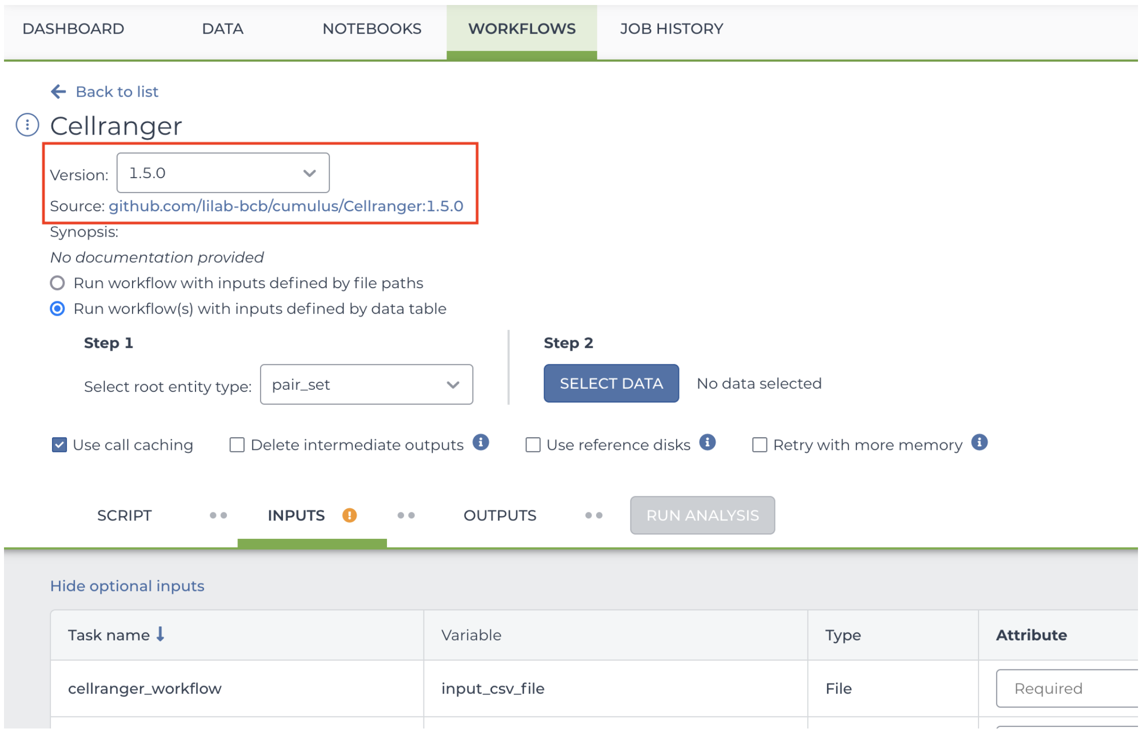
4. When finished, you can see *Cellranger* workflow appearing in "WORKFLOWS" tab of your Terra workspace:



Cellranger

V. 1.5.0
Source: Dockstore

Moreover, in its workflow page (as below)



Cellranger

Version: 1.5.0

Source: github.com/lilab-bcb/cumulus/Cellranger:1.5.0

Synopsis:

No documentation provided

○ Run workflow with inputs defined by file paths

● Run workflow(s) with inputs defined by data table

Step 1 Select root entity type: pair_set

Step 2 No data selected

☑ Use call caching ☐ Delete intermediate outputs ☐ Use reference disks ☐ Retry with more memory

Task name	Variable	Type	Attribute
cellranger_workflow	input_csv_file	File	Required

you can even switch the workflow's version in "Version" drop-down menu, and click the link in "Source" field to view the workflow's WDL source code.

1.1.4 Release notes

Version 2.0

2.0.0 March 14, 2022

Overall:

- Cumulus workflows are now released on Dockstore:
  - Add the tutorial on importing Cumulus workflows to Terra.
  - Archive the legacy versions on Broad Method Registry.
- Add support on multiple platforms via **backend** input: `gcp` for Google Cloud, `aws` for Amazon AWS, `local` for local machine. Enable Google Cloud support by default.
- For Amazon AWS backend, add **awsMaxRetries** input to set the maximum retries allowed for job execution at runtime. By default, use `5`.
- Update the command-line job submission tutorial to work with Altocumulus v2.0.0 or later.
- On Examples:
  - Update gene expression, hashing and CITE-Seq example tutorial.
  - Add tutorial on 10x CellPlex analysis using Cumulus workflows on Cloud.

Workflow-specific:

- Add *STARsolo_create_reference* workflow to build genome references for STARsolo counting. See its documentation for details.

- On *Cellranger* workflow:
  - Add support for 10x Cell Ranger version `6.1.1` and `6.1.2`, and use `6.1.2` by default. See Cell Ranger v6.1 release notes.
  - Add support for 10x Cell Ranger ARC version `2.0.1`, and use it by default. See Cell Ranger ARC v2.0 release notes for the release notes.
  - Upgrade cumulus_feature_barcoding to version `0.7.0` to allow manually set barcode starting position (via input **crispr_barcode_pos**).
  - Add support for non 10x CRISPR assays. See the description of `crispr` **DataType** value in this section for details.
  - For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per sample listed in the input sample sheet) forms.
  - Add **fastq_outputs** to workflow output, which contains *mkfastq* step output folders for samples listed in the input sample sheet.
  - Add **count_outputs** to workflow output, which contains *count* step output folderrs for samples listed in the input sample sheet.
- On *Spaceranger* workflow:
  - Add support for 10x Space Ranger version `1.3.0` and `1.3.1`, and use `1.3.1` by default. See Space Ranger v1.3 release notes for the release notes.
  - For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per library) forms.
  - Add output section for the workflow. See here for details.
  - Retire old genome references:
    - Keep `GRCh38-2020-A` and `mm10-2020-A`.
    - Retire `GRCh38`, `mm10`, `GRCh38-2020-A-premrna` and `mm10-2020-A-premrna`. Users can still reach out to Cumulus team to ask for URIs to these old references, but they are not provided by default.
  - In the description of **ReorientImages** field of input sample sheet, add the information on its valid values.
- On *STARsolo* workflow:
  - Add support for STAR version `2.7.9a`, and use it by default. See STAR v2.7.9a release notes for the release notes.
  - Reorganize the workflow by exposing more inputs to users.
  - Add support on more protocols: 10x multiome, 10x 5' (both SC5P-R2 and SC5P-PE), Slide-Seq and Share-Seq. See *here <./starsolo.html#prepare-a-sample-sheet>* for details.
  - Use input **read1_fastq_pattern** and **read2_fastq_pattern** to support fastq files generated by Cell Ranger or SeqWell, as well as Sequence Read Archive (SRA) data.
  - For input data consisting of fastq files, it's able to handle folder structure of both flat (all fastq files in one folder) and nested (one subfolder per library) forms.
  - Do not attach filename prefix to output files to avoid the incorrect SJ raw *feature.tsv* symlink error, which would cause the folder delocalization fail. (see discussion with STAR team)
  - Add STAR log file to workflow output. This is the *Log.out* file if running STAR locally, which can be used for tracking the process and sharing with STAR team when opening an issue there.
  - Retire old genome references:

    * Keep `GRCh38-2020-A`, `mm10-2020-A`, and `GRCh38-and-mm10-2020-A`.
    * Retire old references listed here. Users can still reach out to Cumulus team to ask for URIs to them, but they are not provided by default.

- On *Demultiplexing* workflow:
  - Upgrade demuxEM to version `0.1.7` for bug fix.
- On *Cellranger_create_reference* workflow:
  - Add the generated reference file to the workflow output.
  - Bug fix in using input **memory**.
  - Update documentation to suggest only using Cell Ranger version `6.1.1` or later for building reference, as v6.0.1 has issues which leave the job running without terminating.
- On *Cellranger_atac_create_reference* workflow:
  - Add the generated reference file to the workflow output.
- On *Cellranger_vdj_create_reference* workflow:
  - Add the generated reference file to the workflow output.

Version 1.x

Version 1.5.1 September 15, 2021

- Fix the issue of WDLs after Terra platform updates the Cromwell engine.

Version 1.5.0 July 20, 2021

- **On *demultiplexing* workflow**
  - Update *demuxEM* to v0.1.6.
- **On *cumulus* workflow**
  - Add Nonnegative Matrix Factorization (NMF) feature: `run_nmf` and `nmf_n` inputs.
  - Add integrative NMF (iNMF) data integration method: `inmf` option in `correction_method` input; the number of expected factors is also specified by `nmf_n` input.
  - When NMF or iNMF is enabled, word cloud plots and gene program UMAP plots of NMF/iNMF results will be generated.
  - Update *Pegasus* to v1.4.2.

Version 1.4.0 May 17, 2021

- **On *cellranger* workflow**
  - Add support for multiomics analysis using linked samples, *cellranger-arc count*, *cellranger multi* and *cellranger count* will be automatically triggered based on the sample sheet
  - Add support for cellranger version 6.0.1 and 6.0.0

  - Add support for cellranger-arc version 2.0.0, 1.0.1, 1.0.0
  - Add support for cellranger-atac version 2.0.0
  - Add support for cumulus_feature_barcoding version 0.6.0, which handles CellPlex CMO tags
  - Add *GRCh38-2020-A_arc_v2.0.0*, *mm10-2020-A_arc_v2.0.0*, *GRCh38-2020-A_arc_v1.0.0* and *mm10-2020-A_arc_v1.0.0* references for *cellranger-arc*.
  - Fixed bugs in cellranger_atac_create_reference
  - Add delete undetermined FASTQs option for mkfastq
- **On *demultiplexing* workflow**
  - Replace *demuxlet* with *popscle*, which includes both *demuxlet* and *freemuxlet*
- **On *cumulus* workflow**
  - Fixed bug that `remap_singlets` and `subset_singlets` don't work when input is in sample sheet format.
- Modified workflows to remove trailing spaces and support spaces within output_directory

Version 1.3.0 February 2, 2021

- **On *cumulus* workflow:**
  - Change `cumulus_version` to `pegasus_version` to avoid confusion.
  - Update to use Pegasus v1.3.0 for analysis.

Version 1.2.0 January 19, 2021

- **Add *spaceranger* workflow:**
  - Wrap up spaceranger version 1.2.1
- **On *cellranger* workflow:**
  - Fix workflow WDL to support both single index and dual index
  - Add support for cellranger version 5.0.1 and 5.0.0
  - Add support for targeted gene expression analysis
  - Add support for `--include-introns` and `--no-bam` options for cellranger count
  - Remove `--force-cells` option for cellranger vdj as noted in cellranger 5.0.0 release note
  - Add *GRCh38_vdj_v5.0.0* and *GRCm38_vdj_v5.0.0* references
- Bug fix on *cumulus* workflow.
- Reorganize the sidebar of Cumulus documentation website.

Version 1.1.0 December 28, 2020

- **On *cumulus* workflow:**
  - Add CITE-Seq data analysis back. (See section Run CITE-Seq analysis for details)

- Add doublet detection. (See `infer_doublets`, `expected_doublet_rate`, and `doublet_cluster_attribute` input fields)
- For tSNE visualization, only support FIt-SNE algorithm. (see `run_tsne` and `plot_tsne` input fields)
- Improve efficiency on log-normalization and DE tests.
- Support multiple marker JSON files used in cell type annotation. (see `organism` input field)
- More preset gene sets provided in gene score calculation. (see `calc_signature_scores` input field)

- **Add *star_solo* workflow (see STARsolo section for details):**
  - Use STARsolo to generate count matrices from FASTQ files.
  - Support chemistry protocols such as 10X-V3, 10X-V2, DropSeq, and SeqWell.
- Update the example of analyzing hashing and CITE-Seq data (see Example section) with the new workflows.
- Bug fix.

Version 1.0.0 September 23, 2020

- Add *demultiplexing* workflow for cell-hashing/nucleus-hashing/genetic-pooling analysis.
- Add support on CellRanger version `4.0.0`.
- **Update *cumulus* workflow with Pegasus version `1.0.0`:**
  - Use `zarr` file format to handle data, which has a better I/O performance in general.
  - Support focus analysis on Unimodal data, and appending other Unimodal data to it. (`focus` and `append` inputs in *cluster* step).
  - Quality-Control: Change `percent_mito` default from `10.0` to `20.0`; by default remove bounds on UMIs (`min_umis` and `max_umis` inputs in *cluster* step).
  - Quality-Control: Automatically figure out name prefix of mitochondrial genes for `GRCh38` and `mm10` genome reference data.
  - Support signature / gene module score calculation. (`calc_signature_scores` input in *cluster* step)
  - Add *Scanorama* method to batch correction. (`correction_method` input in *cluster* step).
  - Calculate UMAP embedding by default, instead of FIt-SNE.
  - Differential Expression (DE) analysis: remove inputs `mwu` and `auc` as they are calculated by default. And cell-type annotation uses MWU test result by default.
- Remove *cumulus_subcluster* workflow.

Version 0.x

Version 0.15.0 May 6, 2020

- Update all workflows to OpenWDL version 1.0.
- Cumulus now supports multi-job execution from Terra data table input.

- Cumulus generates Cirrocumulus input in `.cirro` folder, instead of a huge `.parquet` file.

Version 0.14.0 February 28, 2020

- Added support for gene-count matrices generation using alternative tools (STARsolo, Optimus, Salmon alevin, Kallisto BUStools).
- Cumulus can process demultiplexed data with remapped singlets names and subset of singlets.
- Update VDJ related inputs in Cellranger workflow.
- SMART-Seq2 and Count workflows are in OpenWDL version 1.0.

Version 0.13.0 February 7, 2020

- Added support for aggregating scATAC-seq samples.
- Cumulus now accepts mtx format input.

Version 0.12.0 December 14, 2019

- Added support for building references for sc/snRNA-seq, scATAC-seq, single-cell immune profiling, and SMART-Seq2 data.

Version 0.11.0 December 4, 2019

- Reorganized Cumulus documentation.

Version 0.10.0 October 2, 2019

- scCloud is renamed to Cumulus.
- Cumulus can accept either a sample sheet or a single file.

Version 0.7.0 Feburary 14, 2019

- Added support for 10x genomics scATAC assays.
- scCloud runs FIt-SNE as default.

Version 0.6.0 January 31, 2019

- Added support for 10x genomics V3 chemistry.
- Added support for extracting feature matrix for Perturb-Seq data.
- Added R script to convert output_name.seurat.h5ad to Seurat object. Now the raw.data slot stores filtered raw counts.
- Added min_umis and max_umis to filter cells based on UMI counts.
- Added QC plots and improved filtration spreadsheet.

- Added support for plotting UMAP and FLE.
- Now users can upload their JSON file to annotate cell types.
- Improved documentation.
- Added lightGBM based marker detection.

Version 0.5.0 November 18, 2018

- Added support for plated-based SMART-Seq2 scRNA-Seq data.

Version 0.4.0 October 26, 2018

- Added CITE-Seq module for analyzing CITE-Seq data.

Version 0.3.0 October 24, 2018

- Added the demuxEM module for demultiplexing cell-hashing/nuclei-hashing data.

Version 0.2.0 October 19, 2018

- Added support for V(D)J and CITE-Seq/cell-hashing/nuclei-hashing.

Version 0.1.0 July 27, 2018

- KCO tools released!

1.1.5 Run Cell Ranger tools using cellranger_workflow

cellranger_workflow wraps Cell Ranger to process single-cell/nucleus RNA-seq, single-cell ATAC-seq and single-cell immune profiling data, and supports feature barcoding (cell/nucleus hashing, CITE-seq, Perturb-seq). It also provide routines to build cellranger references.

A general step-by-step instruction

This section mainly considers jobs starting from BCL files. If your job starts with FASTQ files, and only need to run `cellranger count` part, please refer to this subsection.

1. Import `cellranger_workflow`

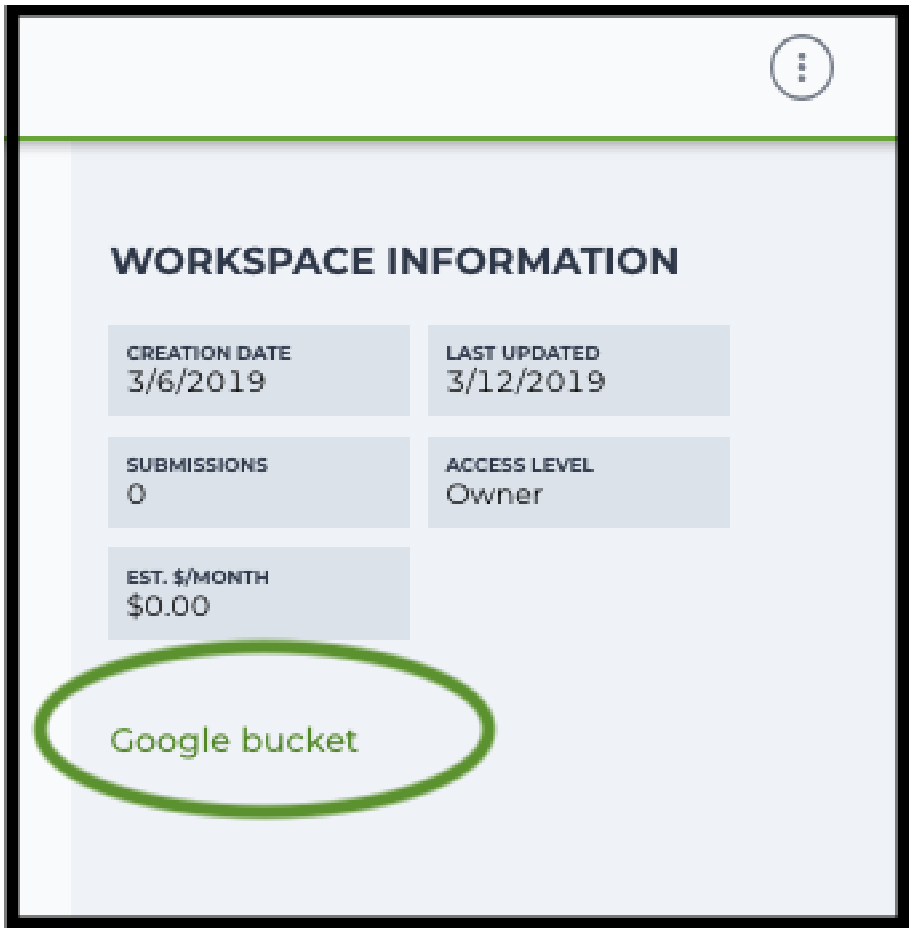
Import *cellranger_workflow* workflow to your workspace by following instructions in Import workflows to Terra. You should choose workflow **github.com/lilab-bcb/cumulus/CellRanger** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *cellranger_workflow* workflow in the drop-down menu.

2. Upload sequencing data to Google bucket

Copy your sequencing output to your workspace bucket using gsutil (you already have it if you've installed Google cloud SDK) in your unix terminal.

You can obtain your bucket URL in the dashboard tab of your Terra workspace under the information panel.



WORKSPACE INFORMATION

CREATION DATE 3/6/2019

LAST UPDATED 3/12/2019

SUBMISSIONS 0

ACCESS LEVEL Owner

EST. $/MONTH $0.00

Google bucket

Use `gsutil cp [OPTION]... src_url dst_url` to copy data to your workspace bucket. For example, the following command copies the directory at /foo/bar/nextseq/Data/VK18WBC6Z4 to a Google bucket:

```
gsutil -m cp -r /foo/bar/nextseq/Data/VK18WBC6Z4 gs://fc-e0000000-0000-0000-
↪0000-000000000000/VK18WBC6Z4
```

`-m` means copy in parallel, `-r` means copy the directory recursively, and `gs://fc-e0000000-0000-0000-0000-000000000000` should be replaced by your own workspace Google bucket URL.

Note: If input is a folder of BCL files, users do not need to upload the whole folder to the Google bucket. Instead, they only need to upload the following files:

```
RunInfo.xml
RTAComplete.txt
```

(continues on next page)

(continued from previous page)

```
runParameters.xml
Data/Intensities/s.locs
Data/Intensities/BaseCalls
```

If data are generated using MiSeq or NextSeq, the location files are inside lane subfloders `L001` under `Data/Intensities/`. In addition, if users' data only come from a subset of lanes (e.g. `L001` and `L002`), users only need to upload lane subfolders from the subset (e.g. `Data/Intensities/BaseCalls/L001, Data/Intensities/BaseCalls/L002` and `Data/Intensities/L001, Data/Intensities/L002` if sequencer is MiSeq or NextSeq).

Alternatively, users can submit jobs through command line interface (CLI) using altocumulus, which will smartly upload BCL folders according to the above rules.

3. Prepare a sample sheet

3.1 Sample sheet format:

Please note that the columns in the CSV can be in any order, but that the column names must match the recognized headings.

The sample sheet describes how to demultiplex flowcells and generate channel-specific count matrices. Note that *Sample*, *Lane*, and *Index* columns are defined exactly the same as in 10x's simple CSV layout file.

A brief description of the sample sheet format is listed below (**required column headers are shown in bold**).

Column	Description
Sample	Contains sample names. Each 10x channel should have a unique sample name. Sample name can only contain characters from [a-zA-Z0-9_-].
Reference	Provides the reference genome used by Cell Ranger for each 10x channel. The elements in the *reference* column can be either Google bucket URLs to reference tarballs or keywords such as *GRCh38-2020-A*. A full list of available keywords is included in each of the following data type sections (e.g. sc/snRNA-seq) below.
Flowcell	Indicates the Google bucket URLs of uploaded BCL folders. If starts with FASTQ files, this should be Google bucket URLs of uploaded FASTQ folders. The FASTQ folders should contain one subfolder for each sample in the flowcell with the sample name as the subfolder name. Each subfolder contains FASTQ files for that sample.
Lane	Tells which lanes the sample was pooled into. Can be either single lane (e.g. 8) or a range (e.g. 7-8) or all (e.g. *).
Index	Sample index (e.g. SI-GA-A12).
Chemistry	Describes the 10x chemistry used for the sample. This column is optional.
DataType	Describes the data type of the sample — *rna*, *vdj*, *citeseq*, *hashing*, *cmo*, *crispr*, *atac*. **rna** refers to gene expression data (*cellranger count*), **vdj** refers to V(D)J data (*cellranger vdj*), **citeseq** refers to CITE-Seq tag data, **hashing** refers to cell-hashing or nucleus-hashing tag data, **adt**, which refers to the case where *hashing* and *citeseq* reads are in a sample library. **cmo** refers to cell multiplexing oligos used in 10x Genomics' CellPlex assay, **crispr** refers to Perturb-seq guide tag data, **atac** refers to scATAC-Seq data (*cellranger-atac count*), This column is optional and the default data type is *rna*.
FeatureBarcodeFile	Google bucket urls pointing to feature barcode files for *rna*, *citeseq*, *hashing*, *cmo* and *crispr* data. Features can be either targeted genes for targeted gene expression analysis, antibody for CITE-Seq, cell-hashing, nucleus-hashing or gRNA for Perburb-seq. If *cmo* data is analyzed separately using *cumulus_feature_barcoding*, file format should follow the guide in Feature barcoding assays section, otherwise follow the guide in Single-cell multiomics section. This column is only required for targeted gene expression analysis (*rna*), CITE-Seq (*citeseq*), cell-hashing or nucleus-hashing (*hashing*), CellPlex (*cmo*) and Perturb-seq (*crispr*).
Link	Designed for Single Cell Multiome ATAC + Gene Expression, Feature Barcoding, or CellPlex. Link multiple modalities together using a single link name. cellranger-arc count, cellranger count, or cellranger multi will be triggered

The sample sheet supports sequencing the same 10x channels across multiple flowcells. If a sample is sequenced across multiple flowcells, simply list it in multiple rows, with one flowcell per row. In the following example, we have 4 samples sequenced in two flowcells.

Example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,1-2,SI-GA-A8,threeprime,rna
sample_2,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,3-4,SI-GA-B8,SC3Pv3,rna
sample_3,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4,
↪5-6,SI-GA-C8,fiveprime,rna
sample_4,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4,
↪7-8,SI-GA-D8,fiveprime,rna
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9Z2,1-2,SI-GA-A8,threeprime,rna
sample_2,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9Z2,3-4,SI-GA-B8,SC3Pv3,rna
sample_3,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2,
↪5-6,SI-GA-C8,fiveprime,rna
sample_4,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2,
↪7-8,SI-GA-D8,fiveprime,rna
```

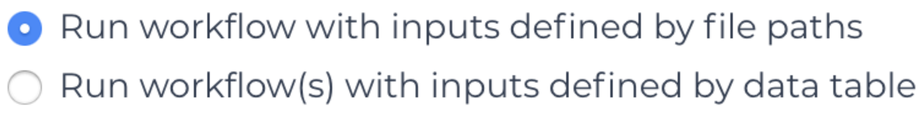
3.2 Upload your sample sheet to the workspace bucket:

Example:

```
gsutil cp /foo/bar/projects/sample_sheet.csv gs://fc-e0000000-0000-
↪0000-0000-000000000000/
```

4. Launch analysis

In your workspace, open `cellranger_workflow` in `WORKFLOWS` tab. Select the desired snapshot version (e.g. latest). Select `Run workflow with inputs defined by file paths` as below


Run workflow with inputs defined by file paths [✓]
Run workflow(s) with inputs defined by data table

and click `SAVE` button. Select `Use call caching` and click `INPUTS`. Then fill in appropriate values in the `Attribute` column. Alternative, you can upload a JSON file to configure input by clicking `Drag or click to upload json`.

Once INPUTS are appropriated filled, click `RUN ANALYSIS` and then click `LAUNCH`.

5. Notice: run `cellranger mkfastq` if you are non Broad Institute users

Non Broad Institute users that wish to run `cellranger mkfastq` must create a custom docker image that contains `bcl2fastq`.

See *bcl2fastq* instructions.

6. Run `cellranger count` only

Sometimes, users might want to perform demultiplexing locally and only run the count part on the cloud. This section describes how to only run the count part via `cellranger_workflow`.

1. Copy your FASTQ files to the workspace using gsutil in your unix terminal. There are two cases:

    - **Case 1**: All the FASTQ files are in one top-level folder. Then you can simply upload this folder to Cloud, and in your sample sheet, make sure **Sample** names are consistent with the filename prefix of their corresponding FASTQ files.
    - **Case 2**: In the top-level folder, each sample has a dedicated subfolder containing its FASTQ files. In this case, you need to upload the whole top-level folder, and in your sample sheet, make sure **Sample** names and their corresponding subfolder names are identical.

    Notice that if your FASTQ files are downloaded from the Sequence Read Archive (SRA) from NCBI, you must rename your FASTQs to follow the bcl2fastq file naming conventions.

    Example:

    ```
    gsutil -m cp -r /foo/bar/fastq_path/K18WBC6Z4 gs://fc-e0000000-
    ↪0000-0000-0000-000000000000/K18WBC6Z4_fastq
    ```

2. Create a sample sheet following the similar structure as above, except the following differences:

    - **Flowcell** column should list Google bucket URLs of the FASTQ folders for flowcells.
    - **Lane** and **Index** columns are NOT required in this case.

    Example:

    ```
    Sample,Reference,Flowcell
    sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-
    ↪000000000000/K18WBC6Z4_fastq
    ```

3. Set optional input `run_mkfastq` to `false`.

7. Workflow outputs

See the table below for workflow level outputs.

Name	Type	Description
fastq_outputs	Array[Array[String]?]	The top-level array contains results (as arrays) for different data modalities. The inner-level array contains cloud locations of FASTQ files, one url per flowcell.
count_outputs	Array[Array[String]?]	The top-level array contains results (as arrays) for different data modalities. The inner-level array contains cloud locations of count matrices, one url per sample.
count_matrix	String	Cloud url for a template count_matrix.csv to run Cumulus. It only contains sc/snRNA-Seq samples.

Single-cell and single-nucleus RNA-seq

To process sc/snRNA-seq data, follow the specific instructions below.

Sample sheet

1. **Reference** column.

   Pre-built scRNA-seq references are summarized below.

Keyword	Description
GRCh38-2020-A	Human GRCh38 (GENCODE v32/Ensembl 98)
mm10-2020-A	Mouse mm10 (GENCODE vM23/Ensembl 98)
GRCh38_and_mm10-2020-A	Human GRCh38 (GENCODE v32/Ensembl 98) and mouse mm10 (GENCODE vM23/Ensembl 98)
GRCh38_v3.0.0	Human GRCh38, cellranger reference 3.0.0, Ensembl v93 gene annotation
hg19_v3.0.0	Human hg19, cellranger reference 3.0.0, Ensembl v87 gene annotation
mm10_v3.0.0	Mouse mm10, cellranger reference 3.0.0, Ensembl v93 gene annotation
GRCh38_and_mm10_v3.1.0	Human (GRCh38) and mouse (mm10), cellranger references 3.1.0, Ensembl v93 gene annotations for both human and mouse
hg19_and_mm10_v3.0.0	Human (hg19) and mouse (mm10), cellranger reference 3.0.0, Ensembl v93 gene annotations for both human and mouse
GRCh38_v1.2.0 or **GRCh38**	Human GRCh38, cellranger reference 1.2.0, Ensembl v84 gene annotation
hg19_v1.2.0 or **hg19**	Human hg19, cellranger reference 1.2.0, Ensembl v82 gene annotation
mm10_v1.2.0 or **mm10**	Mouse mm10, cellranger reference 1.2.0, Ensembl v84 gene annotation
GRCh38_and_mm10_v1.2.0 or **GRCh38_and_mm10**	Human and mouse, built from GRCh38 and mm10 cellranger references, Ensembl v84 gene annotations are used
GRCh38_and_SARSCoV2	Human GRCh38 and SARS-COV-2 RNA genome, cellranger reference 3.0.0, generated by Carly Ziegler. The SARS-COV-2 viral sequence and gtf are as described in [Kim et al. Cell 2020] (https://github.com/hyeshik/sars-cov-2-transcriptome, BetaCov/South Korea/KCDC03/2020 based on NC_045512.2). The GTF was edited to include only CDS regions, and regions were added to describe the 5' UTR ("SARSCoV2_5prime"), the 3' UTR ("SARSCoV2_3prime"), and reads aligning to anywhere within the Negative Strand("SARSCoV2_NegStrand"). Additionally, trailing A's at the 3' end of the virus were excluded from the SARSCoV2 fasta, as these were found to drive spurious viral alignment in pre-COVID19 samples.

   Pre-built snRNA-seq references are summarized below.

Keyword	Description
GRCh38_premrna_v3.0.0	Human, introns included, built from GRCh38 cellranger reference 3.0.0, Ensembl v93 gene annotation, treating annotated transcripts as exons
GRCh38_premrna_v1.2.0 or **GRCh38_premrna**	Human, introns included, built from GRCh38 cellranger reference 1.2.0, Ensembl v84 gene annotation, treating annotated transcripts as exons
mm10_premrna_v1.2.0 or **mm10_premrna**	Mouse, introns included, built from mm10 cellranger reference 1.2.0, Ensembl v84 gene annotation, treating annotated transcripts as exons
GRCh38_premrna_and_mm10_premrna_v1.2.0 or **GRCh38_premrna_and_mm10_premrna**	Human and mouse, introns included, built from GRCh38_premrna_v1.2.0 and mm10_premrna_v1.2.0
GRCh38_premrna_and_SARSCoV2	Human, introns included, built from GRCh38_premrna_v3.0.0, and SARS-COV-2 RNA genome. This reference was generated by Carly Ziegler. The SARS-COV-2 RNA genome is from [Kim et al. Cell 2020] (https://github.com/hyeshik/sars-cov-2-transcriptome, BetaCov/South Korea/KCDC03/2020 based on NC_045512.2). Please see the description of *GRCh38_and_SARSCoV2* above for details.

2. **Index** column.

   Put 10x single cell RNA-seq sample index set names (e.g. SI-GA-A12) here.

3. *Chemistry* column.

   According to *cellranger count*'s documentation, chemistry can be

Chemistry	Explanation
auto	autodetection (default). If the index read has extra bases besides cell barcode and UMI, autodetection might fail. In this case, please specify the chemistry
threeprime	Single Cell 3
fiveprime	Single Cell 5
SC3Pv1	Single Cell 3 v1
SC3Pv2	Single Cell 3 v2
SC3Pv3	Single Cell 3 v3. You should set cellranger version input parameter to >= 3.0.2
SC5P-PE	Single Cell 5 paired-end (both R1 and R2 are used for alignment)
SC5P-R2	Single Cell 5 R2-only (where only R2 is used for alignment)

4. *DataType* column.

   This column is optional with a default **rna**. If you want to put a value, put **rna** here.

5. *FetureBarcodeFile* column.

   Put target panel CSV file here for targeted expressiond data. Note that if a target panel CSV is present, cell ranger version must be >= 4.0.0.

6. Example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType,FeatureBarcodeFile
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4,1-
↪2,SI-GA-A8,threeprime,rna
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2,1-
↪2,SI-GA-A8,threeprime,rna
```

(continues on next page)

(continued from previous page)

```
sample_2,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4,5-6,
↪SI-GA-C8,fiveprime,rna
sample_2,mm10-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2,5-6,
↪SI-GA-C8,fiveprime,rna
sample_3,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4,3,
↪SI-TT-A1,auto,rna,gs://fc-e0000000-0000-0000-0000-000000000000/immunology_v1.0_
↪GRCh38-2020-A.target_panel.csv
```

Workflow input

For sc/snRNA-seq data, `cellranger_workflow` takes Illumina outputs as input and runs `cellranger mkfastq` and `cellranger count`. Revalant workflow inputs are described below, with required inputs highlighted in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and Chemistry, DataType, FeatureBarcodeFile as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_output"	Results are written under directory *output_directory* and will overwrite any existing files at this location.
run_mkfastq	If you want to run `cellranger mkfastq`	true	true
run_count	If you want to run `cellranger count`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
mkfastq_filter_single_index	Only demultiplex samples identified by an i7-only sample index, ignoring dual-indexed samples. Dual-indexed samples will not be demultiplexed	false	false
mkfastq_use_bases_mask	Override the read lengths as specified in *RunInfo.xml*	"Y28n*,I8n*,N10,Y90n*"	
mkfastq_delete_undetermined	Delete undetermined FASTQ files generated by bcl2fastq2	true	false
force_cells	Force pipeline to use this number of cells, bypassing the cell detection algorithm, mutually exclusive with expect_cells	6000	
expect_cells	Expected number of recovered cells. Mutually exclusive with force_cells	3000	
include_introns	Turn this option on to also count reads mapping to intronic regions. With this option, users do not need to use pre-mRNA references. Note that if this option is set, cellranger_version must be >= 5.0.0.	false	false
no_bam	Turn this option on to disable BAM file generation. This option is only available if cellranger_version >= 5.0.0.	false	false
secondary	Perform Cell Ranger secondary analysis (dimensionality reduction, clustering, etc.)	false	false
cellranger_version	cellranger version, could be 6.1.2, 6.1.1, 6.0.2, 6.0.1, 6.0.0, 5.0.1, 5.0.0, 4.0.0, 3.1.0, 3.0.2, or 2.2.0	"6.1.2"	"6.1.2"
config_version	config docker version used for processing sample sheets, could be 0.2, 0.1	"0.2"	"0.2"

Workflow output

See the table below for important sc/snRNA-seq outputs.

Name	Type	Description
cellranger_mkfastq.output_fastqs_directory	Array[String]?	Subworkflow output. A list of cloud urls containing FASTQ files, one url per flowcell.
cellranger_count.output_count_directory	Array[String]?	Subworkflow output. A list of cloud urls containing gene count matrices, one url per sample.
cellranger_count.output_web_summary	Array[File]?	Subworkflow output. A list of htmls visualizing QCs for each sample (cellranger count output).
collect_summaries.metrics_summaries	File?	Task output. A excel spreadsheet containing QCs for each sample.
count_matrix	String	Workflow output. Cloud url for a template count_matrix.csv to run Cumulus.

Feature barcoding assays (cell & nucleus hashing, CITE-seq and Perturb-seq)

`cellranger_workflow` can extract feature-barcode count matrices in CSV format for feature barcoding assays such as *cell and nucleus hashing*, *CellPlex*, *CITE-seq*, and *Perturb-seq*. For cell and nucleus hashing as well as CITE-seq, the feature refers to antibody. For Perturb-seq, the feature refers to guide RNA. Please follow the instructions below to configure `cellranger_workflow`.

Prepare feature barcode files

Prepare a CSV file with the following format: feature_barcode,feature_name. See below for an example:

```
TTCCTGCCATTACTA,sample_1
CCGTACCTCATTGTT,sample_2
GGTAGATGTCCTCAG,sample_3
TGGTGTCATTCTTGA,sample_4
```

The above file describes a cell hashing application with 4 samples.

If cell hashing and CITE-seq data share a same sample index, you should concatenate hashing and CITE-seq barcodes together and add a third column indicating the feature type. See below for an example:

```
TTCCTGCCATTACTA,sample_1,hashing
CCGTACCTCATTGTT,sample_2,hashing
GGTAGATGTCCTCAG,sample_3,hashing
TGGTGTCATTCTTGA,sample_4,hashing
CTCATTGTAACTCCT,CD3,citeseq
GCGCAACTTGATGAT,CD8,citeseq
```

Then upload it to your google bucket:

```
gsutil antibody_index.csv gs://fc-e0000000-0000-0000-0000-000000000000/
↪antibody_index.csv
```

Sample sheet

1. **Reference** column.

   This column is not used for extracting feature-barcode count matrix. To be consistent, please put the reference for the associated scRNA-seq assay here.

2. **Index** column.

   The ADT/HTO index can be either Illumina index primer sequence (e.g. `ATTACTCG`, also known as `D701`), or 10x single cell RNA-seq sample index set names (e.g. SI-GA-A12).

   Note 1: All ADT/HTO index sequences (including 10x's) should have the same length (8 bases). If one index sequence is shorter (e.g. ATCACG), pad it with P7 sequence (e.g. ATCACGAT).

   Note 2: It is users' responsibility to avoid index collision between 10x genomics' RNA indexes (e.g. SI-GA-A8) and Illumina index sequences for used here (e.g. `ATTACTCG`).

   Note 3: For NextSeq runs, please reverse complement the ADT/HTO index primer sequence (e.g. use reverse complement `CGAGTAAT` instead of `ATTACTCG`).

3. *Chemistry* column.

   The following keywords are accepted for *Chemistry* column:

Chemistry	Explanation
auto	Default. This is an alias for Single Cell 3' v3 (SC3Pv3)
threeprime	This is another alias for Single Cell 3' v3
SC3Pv3	Single Cell 3 v3
SC3Pv2	Single Cell 3 v2
fiveprime	Single Cell 5
SC5P-PE	Single Cell 5 paired-end (both R1 and R2 are used for alignment)
SC5P-R2	Single Cell 5 R2-only (where only R2 is used for alignment)

4. *DataType* column.

   The following keywords are accepted for *DataType* column:

DataType	Explanation
citeseq	CITE-seq
hashing	Cell or nucleus hashing
cmo	CellPlex
adt	Hashing and CITE-seq are in the same library
crispr	Perturb-seq/CROP-seq If neither *crispr_barcode_pos* nor *scaffold_sequence* (see Workflow input) is set, **crispr** refers to 10x CRISPR assays. If in addition *Chemistry* is set to be **SC3Pv3** or its aliases, Cumulus automatically complement the middle two bases to convert 10x feature barcoding cell barcodes back to 10x RNA cell barcodes. Otherwise, **crispr** refers to non 10x CRISPR assays, such as CROP-Seq. In this case, we assume feature barcoding cell barcodes are the same as the RNA cell barcodes and no cell barcode convertion will be conducted.

5. *FetureBarcodeFile* column.

Put Google Bucket URL of the feature barcode file here.

6. Example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType,FeatureBarcodeFile
sample_1_rna,GRCh38_v3.0.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,1-2,SI-GA-A8,threeprime,rna
sample_1_adt,GRCh38_v3.0.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,1-2,ATTACTCG,SC3Pv3,adt,gs://fc-e0000000-0000-0000-0000-000000000000/
↪antibody_index.csv
sample_2_adt,GRCh38_v3.0.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,3-4,TCCGGAGA,SC3Pv3,adt,gs://fc-e0000000-0000-0000-0000-000000000000/
↪antibody_index.csv
sample_3_crispr,GRCh38_v3.0.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,5-6,CGCTCATT,SC3Pv3,crispr,gs://fc-e0000000-0000-0000-0000-
↪000000000000/crispr_index.csv
```

In the sample sheet above, despite the header row,

- First row describes the normal 3' RNA assay;
- Second row describes its associated antibody tag data, which can from either a CITE-seq, cell hashing, or nucleus hashing experiment.
- Third row describes another tag data, which is in 10x genomics' V3 chemistry. For tag and crispr data, it is important to explicitly state the chemistry (e.g. `SC3Pv3`).
- Last row describes one gRNA guide data for Perturb-seq (see `crispr` in *DataType* field).

Workflow input

For feature barcoding data, `cellranger_workflow` takes Illumina outputs as input and runs `cellranger mkfastq` and `cumulus adt`. Revalant workflow inputs are described below, with required inputs highlighted in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and Chemistry, DataType, FeatureBarcodeFile as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_output"	
run_mkfastq	If you want to run `cellranger mkfastq`	true	true
run_count	If you want to run `cumulus adt`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
mkfastq_filter_single_index	Only demultiplex samples identified by an i7-only sample index, ignoring dual-indexed samples. Dual-indexed samples will not be demultiplexed	false	false
mkfastq_use_bases_mask	Override the read lengths as specified in *RunInfo.xml*	"Y28n*,I8n*,N10,Y90n*"	
mkfastq_delete_undetermined	Delete undetermined FASTQ files generated by bcl2fastq2	true	false
crispr_barcode_pos	Barcode start position at Read 2 (0-based coordinate) for CRISPR	19	0
scaffold_sequence	Scaffold sequence in sgRNA for Purturb-seq, only used for crispr data type.	"GTTTAAGAGCTAAGCTGGAA"	""
max_mismatch	Maximum hamming distance in feature barcodes for the adt task	3	3
min_read_ratio	Minimum read count ratio (non-inclusive) to justify a feature given a cell barcode and feature combination, only used for the adt task and crispr data type	0.1	0.1
cellranger_version	cellranger version, could be 6.1.2, 6.1.1, 6.0.2, 6.0.1, 6.0.0, 5.0.1, 5.0.0, 4.0.0, 3.1.0, 3.0.2, 2.2.0	"6.1.2"	"6.1.2"
cumulus_feature_barcoding_version	Cumulus_feature_barcoding version for extracting feature barcode matrix. Version available: 0.7.0, 0.6.0, 0.5.0, 0.4.0, 0.3.0, 0.2.0.	"0.7.0"	"0.7.0"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
mkfastq_docker_registry	Docker registry to use for `cellranger mkfastq`. Default is the registry to which only Broad users have access. See	"gcr.io/broad-cumulus"	"gcr.io/broad-cumulus"

Parameters used for feature count matrix extraction

If the chemistry is V2, 10x genomics v2 cell barcode white list will be used, a hamming distance of 1 is allowed for matching cell barcodes, and the UMI length is 10. If the chemistry is V3, 10x genomics v3 cell barcode white list will be used, a hamming distance of 0 is allowed for matching cell barcodes, and the UMI length is 12.

For Perturb-seq data, a small number of sgRNA protospace sequences will be sequenced ultra-deeply and we may have PCR chimeric reads. Therefore, we generate filtered feature count matrices as well in a data driven manner:

1. First, plot the histogram of UMIs with certain number of read counts. The number of UMIs with `x` supporting reads decreases when `x` increases. We start from `x = 1`, and a valley between two peaks is detected if we find `count[x] < count[x + 1] < count[x + 2]`. We filter out all UMIs with `< x` supporting reads since they are likely formed due to chimeric reads.
2. In addition, we also filter out barcode-feature-UMI combinations that have their read count ratio, which is defined as total reads supporting barcode-feature-UMI over total reads supporting barcode-UMI, no larger than `min_read_ratio` parameter set above.

Workflow outputs

See the table below for important outputs.

Name	Type	Description
cellranger_mkfastq.output_fastqs_directory	Array[String]?	Subworkflow output. A list of cloud urls containing FASTQ files, one url per flowcell.
cumulus_adt.output_count_directory	Array[String]?	Subworkflow output. A list of cloud urls containing feature-barcode count matrices, one url per sample.

In addition, For each antibody tag or crispr tag sample, a folder with the sample ID is generated under `output_directory`. In the folder, two files — `sample_id.csv` and `sample_id.stat.csv.gz` — are generated.

`sample_id.csv` is the feature count matrix. It has the following format. The first line describes the column names: `Antibody/CRISPR,cell_barcode_1,cell_barcode_2,...,cell_barcode_n`. The following lines describe UMI counts for each feature barcode, with the following format: `feature_name,umi_count_1, umi_count_2,...,umi_count_n`.

`sample_id.stat.csv.gz` stores the gzipped sufficient statistics. It has the following format. The first line describes the column names: `Barcode,UMI,Feature,Count`. The following lines describe the read counts for every barcode-umi-feature combination.

If the feature barcode file has a third column, there will be two files for each feature type in the third column. For example, if `hashing` presents, `sample_id.hashing.csv` and `sample_id.hashing.stat.csv.gz` will be generated.

If data type is `crispr`, three additional files, `sample_id.umi_count.pdf`, `sample_id.filt.csv` and `sample_id.filt.stat.csv.gz`, are generated.

`sample_id.umi_count.pdf` plots number of UMIs against UMI with certain number of reads and colors UMIs with high likelihood of being chimeric in blue and other UMIs in red. This plot is generated purely based on number of reads each UMI has. For better visualization, we do not show UMIs with > 50 read counts (rare in data).

`sample_id.filt.csv` is the filtered feature count matrix. It has the same format as `sample_id.csv`.

`sample_id.filt.stat.csv.gz` is the filtered sufficient statistics. It has the same format as `sample_id.stat.csv.gz`.

Single-cell ATAC-seq

To process scATAC-seq data, follow the specific instructions below.

Sample sheet

1. **Reference** column.

   Pre-built scATAC-seq references are summarized below.

Keyword	Description
GRCh38-2020-A_arc_v2.0.0	Human GRCh38, cellranger-arc/atac reference 2.0.0
mm10-2020-A_arc_v2.0.0	Mouse mm10, cellranger-arc/atac reference 2.0.0
GRCh38_atac_v1.2.0	Human GRCh38, cellranger-atac reference 1.2.0
mm10_atac_v1.2.0	Mouse mm10, cellranger-atac reference 1.2.0
hg19_atac_v1.2.0	Human hg19, cellranger-atac reference 1.2.0
b37_atac_v1.2.0	Human b37 build, cellranger-atac reference 1.2.0
GRCh38_and_mm10_atac_v1.2.0	Human GRCh38 and mouse mm10, cellranger-atac reference 1.2.0
hg19_and_mm10_atac_v1.2.0	Human hg19 and mouse mm10, cellranger-atac reference 1.2.0
GRCh38_atac_v1.1.0	Human GRCh38, cellranger-atac reference 1.1.0
mm10_atac_v1.1.0	Mouse mm10, cellranger-atac reference 1.1.0
hg19_atac_v1.1.0	Human hg19, cellranger-atac reference 1.1.0
b37_atac_v1.1.0	Human b37 build, cellranger-atac reference 1.1.0
GRCh38_and_mm10_atac_v1.1.0	Human GRCh38 and mouse mm10, cellranger-atac reference 1.1.0
hg19_and_mm10_atac_v1.1.0	Human hg19 and mouse mm10, cellranger-atac reference 1.1.0

2. **Index** column.

   Put 10x single cell ATAC sample index set names (e.g. SI-NA-B1) here.

3. *Chemistry* column.

   This column is not used for scATAC-seq data. Put **auto** here as a placeholder if you decide to include the Chemistry column.

4. *DataType* column.

   Set it to **atac**.

5. *FetureBarcodeFile* column.

   Leave it blank for scATAC-seq.

6. Example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType
sample_atac,GRCh38_atac_v1.1.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9YB,*,SI-NA-A1,auto,atac
```

Workflow input

`cellranger_workflow` takes Illumina outputs as input and runs `cellranger-atac mkfastq` and `cellranger-atac count`. Please see the description of inputs below. Note that required inputs are shown in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and Chemistry, DataType, FeatureBarcodeFile as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_atac_output"	
run_mkfastq	If you want to run `cellranger-atac mkfastq`	true	true
run_count	If you want to run `cellranger-atac count`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
mkfastq_filter_single_index	Only demultiplex samples identified by an i7-only sample index, ignoring dual-indexed samples. Dual-indexed samples will not be demultiplexed	false	false
mkfastq_use_bases_mask	Override the read lengths as specified in *RunInfo.xml*	"Y28n*,I8n*,N10,Y90n*"	
mkfastq_delete_undetermined	Delete undetermined FASTQ files generated by bcl2fastq2	true	false
force_cells	Force pipeline to use this number of cells, bypassing the cell detection algorithm	6000	
atac_dim_reduce	Choose the algorithm for dimensionality reduction prior to clustering and tsne: "lsa", "plsa", or "pca"	"lsa"	"lsa"
peaks	A 3-column BED file of peaks to override cellranger atac peak caller. Peaks must be sorted by position and not contain overlapping peaks; comment lines beginning with # are allowed	"gs://fc-e0000000-0000-0000-0000-000000000000/common_peaks.bed"	
cellranger_atac_version	cellranger-atac version. Available options: 2.0.0, 1.2.0, 1.1.0	"2.0.0"	"2.0.0"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
mkfastq_docker_registry	Docker registry to use for `cellranger-atac mkfastq`. Default is the registry to which only Broad users have access. See *bcl2fastq* for making your own registry.	"gcr.io/broad-cumulus"	"gcr.io/broad-cumulus"
acronym_file	The link/path of an index file in TSV format for fetching preset genome references, chemistry whitelists, etc. by their names. Set an GS URI if *backend* is `gcp`; an	"s3://xxxx/index.tsv"	"gs://regev-lab/resources/cellranger/index.tsv"

Workflow output

See the table below for important scATAC-seq outputs.

Name	Type	Description
cellranger_atac_mkfastq.output_fastqs_directory	Array[String]	Subworkflow output. A list of cloud urls containing FASTQ files, one url per flowcell.
cellranger_atac_count.output_count_directory	Array[String]	Subworkflow output. A list of cloud urls containing cellranger-atac count outputs, one url per sample.
cellranger_atac_count.output_web_summary	Array[File]	Subworkflow output. A list of htmls visualizing QCs for each sample (cellranger-atac count output).
collect_summaries_atac.metrics_summaries	File	Task output. A excel spreadsheet containing QCs for each sample.

Aggregate scATAC-Seq Samples

To aggregate multiple scATAC-Seq samples, follow the instructions below:

1. Import `cellranger_atac_aggr` workflow. Please see Step 1 here, and the name of workflow is "**cumulus/cellranger_atac_aggr**".
2. Set the inputs of workflow. Please see the description of inputs below. Notice that required inputs are shown in bold:

Name	Description	Example	Default
aggr_id	Aggregate ID.	"aggr_sample"	
input_counts_directories	A string of comma-separated URLs to directories of samples to be aggregated.	"gs://fc-e0000000-0000-0000-0000-000000000000/data/sample1,gs://fc-e0000000-0000-0000-0000-000000000000/data/sample2"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/aggregate_result"	
genome	The reference genome name used by Cell Ranger, can be either a keyword of pre-built genome, or a Google Bucket URL. See this table for the list of keywords of pre-built genomes.	"GRCh38_atac_v1.2.0"	
normalize	Sample normalization mode. Options are: `none`, `depth`, or `signal`.	"none"	"none"
secondary	Perform secondary analysis (dimensionality reduction, clustering and visualization).	false	false
dim_reduce	Choose the algorithm for dimensionality reduction prior to clustering and tsne. Options are: `lsa`, `plsa`, or `pca`.	"lsa"	"lsa"
peaks	A 3-column BED file of peaks to override cellranger atac peak caller. Peaks must be sorted by position and not contain overlapping peaks; comment lines beginning with # are allowed	"gs://fc-e0000000-0000-0000-0000-000000000000/common_peaks.bed"	
cellranger_atac_version	Cell Ranger ATAC version to use. Options: `2.0.0`, `1.2.0`, `1.1.0`.	"2.0.0"	"2.0.0"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-b"
num_cpu	Number of cpus to request for cellranger atac aggr.	64	64
backend	Cloud backend for file transfer. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS; • "local" for local machine.	"gcp"	"gcp"
memory	Memory size string for cellranger atac aggr.	"57.6G"	"57.6G"
disk_space	Disk space in GB needed for cellranger atac aggr.	500	500
preemptible	Number of preemptible tries.	2	2
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"

1. Check out the output in `output_directory/aggr_id` folder, where `output_directory` and `aggr_id` are the inputs you set in Step 2.

Single-cell immune profiling

To process single-cell immune profiling (scIR-seq) data, follow the specific instructions below.

Sample sheet

1. **Reference** column.

   Pre-built scIR-seq references are summarized below.

Keyword	Description
GRCh38_vdj_v5.0.0	Human GRCh38 V(D)J sequences, cellranger reference 5.0.0, annotation built from Ensembl *Homo_sapiens.GRCh38.94.chr_patch_hapl_scaff.gtf*
GRCm38_vdj_v5.0.0	Mouse GRCm38 V(D)J sequences, cellranger reference 5.0.0, annotation built from Ensembl *Mus_musculus.GRCm38.94.gtf*
GRCh38_vdj_v4.0.0	Human GRCh38 V(D)J sequences, cellranger reference 4.0.0, annotation built from Ensembl *Homo_sapiens.GRCh38.94.chr_patch_hapl_scaff.gtf*
GRCm38_vdj_v4.0.0	Mouse GRCm38 V(D)J sequences, cellranger reference 4.0.0, annotation built from Ensembl *Mus_musculus.GRCm38.94.gtf*
GRCh38_vdj_v3.1.0	Human GRCh38 V(D)J sequences, cellranger reference 3.1.0, annotation built from Ensembl *Homo_sapiens.GRCh38.94.chr_patch_hapl_scaff.gtf*
GRCm38_vdj_v3.1.0	Mouse GRCm38 V(D)J sequences, cellranger reference 3.1.0, annotation built from Ensembl *Mus_musculus.GRCm38.94.gtf*
GRCh38_vdj_v2.0.0 or **GRCh38_vdj**	Human GRCh38 V(D)J sequences, cellranger reference 2.0.0, annotation built from Ensembl *Homo_sapiens.GRCh38.87.chr_patch_hapl_scaff.gtf* and *vdj_GRCh38_alts_ensembl_10x_genes-2.0.0.gtf*
GRCm38_vdj_v2.2.0 or **GRCm38_vdj**	Mouse GRCm38 V(D)J sequences, cellranger reference 2.2.0, annotation built from Ensembl *Mus_musculus.GRCm38.90.chr_patch_hapl_scaff.gtf*

2. **Index** column.

   Put 10x single cell V(D)J sample index set names (e.g. SI-GA-A3) here.

3. *Chemistry* column.

   This column is not used for scIR-seq data. Put **fiveprime** here as a placeholder if you decide to include the Chemistry column.

4. *DataType* column.

   Set it to **vdj**.

5. *FetureBarcodeFile* column.

   Leave it blank for scIR-seq.

6. Example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType
sample_vdj,GRCh38_vdj_v3.1.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9ZZ,1,SI-GA-A1,fiveprime,vdj
```

Workflow input

For scIR-seq data, `cellranger_workflow` takes Illumina outputs as input and runs `cellranger mkfastq` and `cellranger vdj`. Revalant workflow inputs are described below, with required inputs highlighted in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and Chemistry, DataType, FeatureBarcodeFile as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_output"	
run_mkfastq	If you want to run `cellranger mkfastq`	true	true
run_count	If you want to run `cellranger vdj`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
mkfastq_filter_single_index	Only demultiplex samples identified by an i7-only sample index, ignoring dual-indexed samples. Dual-indexed samples will not be demultiplexed	false	false
mkfastq_use_bases_mask	Override the read lengths as specified in *RunInfo.xml*	"Y28n*,I8n*,N10,Y90n*"	
mkfastq_delete_undetermined	Delete undetermined FASTQ files generated by bcl2fastq2	true	false
vdj_denovo	Do not align reads to reference V(D)J sequences before de novo assembly	false	false
vdj_chain	Force the analysis to be carried out for a particular chain type. The accepted values are: • "auto" for auto detection based on TR vs IG representation; • "TR" for T cell receptors; • "IG" for B cell receptors.	"auto"	"auto"
cellranger_version	cellranger version, could be 6.1.2, 6.1.1, 6.0.2, 6.0.1, 6.0.0, 5.0.1, 5.0.0, 4.0.0, 3.1.0, 3.0.2, 2.2.0	"6.1.2"	"6.1.2"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
mkfastq_docker_registry	Docker registry to use for `cellranger mkfastq`. Default is the registry to which only Broad users have access. See *bcl2fastq* for making your own registry.	"gcr.io/broad-cumulus"	"gcr.io/broad-cumulus"
acronym_file	The link/path of an index file in TSV format for fetching preset genome references, chemistry whitelists, etc. by their names. Set an GS URI if *backend* is `gcp`; an S3 URI for `aws` backend; an absolute	"s3://xxxx/index.tsv"	"gs://regev-lab/resources/cellranger/index.tsv"

Workflow output

See the table below for important scIR-seq outputs.

Name	Type	Description
cellranger_mkfastq.output_fastqs_directory	Array[String]?	Subworkflow output. A list of cloud urls containing FASTQ files, one url per flowcell.
cellranger_vdj.output_vdj_directory	Array[String]?	Subworkflow output. A list of cloud urls containing vdj results, one url per sample.
cellranger_vdj.output_web_summary	Array[File]?	Subworkflow output. A list of htmls visualizing QCs for each sample (cellranger vdj output).
collect_summaries_vdj.metrics_summaries	File?	Task output. A excel spreadsheet containing QCs for each sample.

Single-cell multiomics

To utilize cellranger arc/cellranger multi/cellranger count for single-cell multiomics, follow the specific instructions below. In particular, we put each single modality in one separate lin in the sample sheet as described above. We then use the *Link* column to link multiple modalities together. Depending on the modalities included, *cellranger arc* (Multiome ATAC + Gene Expression), *cellranger multi* (CellPlex), or *cellranger count* (Feature Barcode) will be triggered. Note that cumulus_feature_barcoding/demuxEM would not be triggered for hashing/citeseq in this setting.

Sample sheet

1. **Reference** column.

   Pre-built Multiome ATAC + Gene Expression references are summarized below. CellPlex and Feature Barcode use the same reference as in Single-cell and single-nucleus RNA-seq.

Keyword	Description
GRCh38-2020-A_arc_v2.0.0	Human GRCh38 sequences (GENCODE v32/Ensembl 98), cellranger arc reference 2.0.0
mm10-2020-A_arc_v2.0.0	Mouse GRCm38 sequences (GENCODE vM23/Ensembl 98), cellranger arc reference 2.0.0
GRCh38-2020-A_arc_v1.0.0	Human GRCh38 sequences (GENCODE v32/Ensembl 98), cellranger arc reference 1.0.0
mm10-2020-A_arc_v1.0.0	Mouse GRCm38 sequences (GENCODE vM23/Ensembl 98), cellranger arc reference 1.0.0

2. *DataType* column.

   For each modality, set it to the corresponding data type.

3. *FetureBarcodeFile* column.

   For RNA-seq modality, only set this if a target panel is provided. For CMO (CellPlex), provide sample name - CMO tag association as follows:

   ```
   sample1,CMO301|CMO302
   sample2,CMO303
   ```

For CITESeq, Perturb-seq and hashing, provide one CSV file as defined in Feature Barcode Reference. Note that one feature barcode reference should be provided for all feature-barcode related modalities (e.g. *citeseq*, *hashing*, *crispr*) and all these modalities should put the same reference file in *FeatureBarcodeFile* column.

4. *Link* column.

   Put a sample unique link name for all modalities that are linked.

5. Example:

```
Sample,Reference,Flowcell,Lane,Index,DataType,FeatureBarcodeFile,Link
sample1_rna,GRCh38-2020-A_arc_v2.0.0,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9ZZ,*,SI-TT-A1,rna,,sample1
sample1_atac,GRCh38-2020-A_arc_v2.0.0,gs://fc-e0000000-0000-0000-0000-
↪000000000000/VK10WBC9ZZ,*,SI-TT-N1,atac,,sample1
sample2_rna,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9ZX,
↪*,SI-TT-A2,rna,,sample2
sample2_cmo,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9ZX,
↪*,SI-TT-N2,cmo,gs://fc-e0000000-0000-0000-0000-000000000000/cmo.csv,sample2
sample3_rna,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9ZY,
↪*,SI-TT-A3,rna,,sample3
sample3_citeseq,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9ZY,*,SI-TT-N3,citeseq,gs://fc-e0000000-0000-0000-0000-000000000000/
↪feature_ref.csv,sample3
```

In the above example, three linked samples are provided. *cellranger arc*, *cellranger multi* and *cellranger count* will be triggered respectively.

Workflow input

For single-cell multiomics data, `cellranger_workflow` takes Illumina outputs as input and runs `cellranger-arc mkfastq/cellranger mkfastq` and `cellranger-arc count/cellranger multi/cellranger count`. Revalant workflow inputs are described below, with required inputs highlighted in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and Chemistry, DataType, FeatureBarcodeFile, Link as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_output"	
run_mkfastq	If you want to run `cellranger-arc mkfastq/ cellranger mkfastq`	true	true
run_count	If you want to run `cellranger-arc count/ cellranger multi/ cellranger count`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false

Continued on next page

Table 1 – continued from previous page

Name	Description	Example	Default
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
mkfastq_filter_single_index	Only demultiplex samples identified by an i7-only sample index, ignoring dual-indexed samples. Dual-indexed samples will not be demultiplexed	false	false
mkfastq_use_bases_mask	Override the read lengths as specified in *RunInfo.xml*	"Y28n*,I8n*,N10,Y90n*"	
mkfastq_delete_undetermined	Delete undetermined FASTQ files generated by bcl2fastq2	true	false
force_cells	Force pipeline to use this number of cells, bypassing the cell detection algorithm, mutually exclusive with expect_cells. This option is used by *cellranger multi* and *cellranger count*.	6000	
expect_cells	Expected number of recovered cells. Mutually exclusive with force_cells. This option is used by *cellranger multi* and *cellranger count*.	3000	
include_introns	Turn this option on to also count reads mapping to intronic regions. With this option, users do not need to use pre-mRNA references. Note that if this option is set, cellranger_version must be >= 5.0.0. This option is used by *cellranger multi* and *cellranger count*.	false	false
arc_gex_exclude_introns	Disable counting of intronic reads. In this mode, only reads that are exonic and compatible with annotated splice junctions in the reference are counted. **Note:** using this mode will reduce the UMI counts in the feature-barcode matrix.	false	false
no_bam	Turn this option on to disable BAM file generation. This option is only available if cellranger_version >= 5.0.0. This option is used by *cellranger-arc count*, *cellranger multi* and *cellranger count*.	false	false

Continued on next page

Table 1 – continued from previous page

Name	Description	Example	Default
arc_min_atac_count	Cell caller override to define the minimum number of ATAC transposition events in peaks (ATAC counts) for a cell barcode. **Note:** this input must be specified in conjunction with `arc_min_gex_count` input. With both inputs set, a barcode is defined as a cell if it contains at least `arc_min_atac_count` ATAC counts AND at least `arc_min_gex_count` GEX UMI counts.	100	
arc_min_gex_count	Cell caller override to define the minimum number of GEX UMI counts for a cell barcode. **Note:** this input must be specified in conjunction with `arc_min_atac_count`. See the description of `arc_min_atac_count` input for details.	200	
peaks	A 3-column BED file of peaks to override cellranger arc peak caller. Peaks must be sorted by position and not contain overlapping peaks; comment lines beginning with # are allowed	"gs://fc-e0000000-0000-0000-0000-000000000000/common_peaks.bed"	
secondary	Perform Cell Ranger secondary analysis (dimensionality reduction, clustering, etc.). This option is used by *cellranger multi* and *cellranger count*.	false	false
cmo_set	CMO set CSV file, delaring CMO constructs and associated barcodes. See CMO reference for details. Used only for *cellranger multi*.	"gs://fc-e0000000-0000-0000-0000-000000000000/cmo_set.csv"	
cellranger_version	cellranger version, could be 6.1.2, 6.1.1, 6.0.2, 6.0.1, 6.0.0, 5.0.1, 5.0.0, 4.0.0, 3.1.0, 3.0.2	"6.1.2"	"6.1.2"
cellranger_arc_version	cellranger arc version, could be 2.0.0, 1.0.1, 1.0.0	"2.0.0"	"2.0.0"

Continued on next page

Table 1 – continued from previous page

Name	Description	Example	Default
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
mkfastq_docker_registry	Docker registry to use for `cellranger-arc mkfastq`/ `cellranger mkfastq`. Default is the registry to which only Broad users have access. See *bcl2fastq* for making your own registry.	"gcr.io/broad-cumulus"	"gcr.io/broad-cumulus"
acronym_file	The link/path of an index file in TSV format for fetching preset genome references, chemistry whitelists, etc. by their names. Set an GS URI if *backend* is `gcp`; an S3 URI for `aws` backend; an absolute file path for `local` backend.	"s3://xxxx/index.tsv"	"gs://regev-lab/resources/cellranger/index.tsv"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of cpus to request for one node for cellranger mkfastq and cellranger vdj	32	32
memory	Memory size string for cellranger/cellranger-arc mkfastq and cellranger vdj	"120G"	"120G"
mkfastq_disk_space	Optional disk space in GB for mkfastq	1500	1500
count_disk_space	Disk space in GB needed for cellranger count	500	500
arc_num_cpu	Number of cpus to request for one node for cellranger-arc count	64	64
arc_memory	Memory size string for cellranger-arc count	"160G"	"160G"
arc_disk_space	Disk space in GB needed for cellranger-arc count	700	700
backend	Cloud backend for file transfer. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS; • "local" for local machine.	"gcp"	"gcp"
preemptible	Number of preemptible tries	2	2

Continued on next page

Table 1 – continued from previous page

Name	Description	Example	Default
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

Workflow output

See the table below for important sc/snRNA-seq outputs.

Name	Type	Description
cellranger_arc_mkfastq.output_fastqs_directory / cellranger_mkfastq.output_fastqs_directory	Array[String]?	Subworkflow output. A list of cloud urls containing FASTQ files, one url per flowcell.
cellranger_arc_count.output_count_directory / cellranger_multi.output_multi_directory / cellranger_count_fbc.output_count_directory	Array[String]?	Subworkflow output. A list of cloud urls containing *cellranger-arc count*, *cellranger multi* or *cellranger count* outputs, one url per sample.
cellranger_arc_count.output_web_summary / cellranger_count_fbc.output_web_summary	Array[File]?	A list of htmls visualizing QCs for each sample (*cellranger-arc count / cellranger count* output).
collect_summaries_arc.metrics_summaries / collect_summaries_fbc.metrics_summaries	File?	A excel spreadsheet containing QCs for each sample.

Build Cell Ranger References

We provide routines wrapping Cell Ranger tools to build references for sc/snRNA-seq, scATAC-seq and single-cell immune profiling data.

Build references for sc/snRNA-seq

We provide a wrapper of `cellranger mkref` to build sc/snRNA-seq references. Please follow the instructions below.

1. Import `cellranger_create_reference`

Import *cellranger_create_reference* workflow to your workspace by following instructions in Import workflows to Terra. You should choose **github.com/kalarman-cell-observatory/cumulus/Cellranger_create_reference** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *cellranger_create_reference* workflow in the drop-down menu.

2. Upload requred data to Google Bucket

Required data may include input sample sheet, genome FASTA files and gene annotation GTF files.

3. Input sample sheet

If multiple species are specified, a sample sheet in CSV format is required. We describe the sample sheet format below, with required columns highlighted in bold:

Column	Description
Genome	Genome name
Fasta	Location to the genome assembly in FASTA/FASTA.gz format
Genes	Location to the gene annotation file in GTF/GTF.gz format
Attributes	Optional, A list of `key:value` pairs separated by `;`. If set, `cellranger mkgtf` will be called to filter the user-provided GTF file. See 10x filter with mkgtf for more details

Please note that the columns in the CSV can be in any order, but that the column names must match the recognized headings.

See below for an example for building Example:

```
Genome,Fasta,Genes,Attributes
GRCh38,gs://fc-e0000000-0000-0000-0000-000000000000/GRCh38.fa.gz,gs://fc-
↪e0000000-0000-0000-0000-000000000000/GRCh38.gtf.gz,gene_biotype:protein_
↪coding;gene_biotype:lincRNA;gene_biotype:antisense
mm10,gs://fc-e0000000-0000-0000-0000-000000000000/mm10.fa.gz,gs://fc-
↪e0000000-0000-0000-0000-000000000000/mm10.gtf.gz
```

If multiple species are specified, the reference will built under **Genome** names concatenated by '_and_'s. In the above example, the reference is stored under 'GRCh38_and_mm10'.

4. Workflow input

Required inputs are highlighted in bold. Note that **input_sample_sheet** and **input_fasta**, **input_gtf** , **genome** and attributes are mutually exclusive.

Name	Description	Example	Default
input_sample_sheet	A sample sheet in CSV format allows users to specify more than 1 genomes to build references (e.g. human and mouse). If a sample sheet is provided, **input_fasta**, **input_gtf**, and attributes will be ignored.	"gs://fc-e0000000-0000-0000-0000-000000000000/input_sample_sheet.csv"	
input_fasta	Input genome reference in either FASTA or FASTA.gz format	"gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.dna.toplevel.fa.gz"	
input_gtf	Input gene annotation file in either GTF or GTF.gz format	"gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.94.chr_patch_hapl_scaff.gtf.gz"	
genome	Genome reference name. New reference will be stored in a folder named **genome**	refdata-cellranger-vdj-GRCh38-alts-ensembl-3.1.0	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_reference"	
attributes	A list of `key:value` pairs separated by `;`. If this option is not None, `cellranger mkgtf` will be called to filter the user-provided GTF file. See 10x filter with mkgtf for more details	"gene_biotype:protein_coding;gene_biotype:lincRNA;gene_biotype:antisense"	
pre_mrna	If we want to build pre-mRNA references, in which we use full length transcripts as exons in the annotation file. We follow 10x build Cell Ranger compatible pre-mRNA Reference Package to build pre-mRNA references	true	false
ref_version	reference version string	Ensembl v94	
cellranger_version	cellranger version, could be: `6.1.2`, `6.1.1`	"6.1.2"	"6.1.2"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of cpus to request for one node for building indices	1	1
memory	Memory size string for cellranger mkref	"32G"	"32G"
disk_space	Optional disk space in GB	100	100
backend	Cloud backend for file transfer. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS;	"gcp"	"gcp"

5. Workflow output

Name	Type	Description
output_reference	File	Gzipped reference folder with name *genome.tar.gz*. We will also store a copy of the gzipped tarball under **output_directory** specified in the input.

Build references for scATAC-seq

We provide a wrapper of `cellranger-atac mkref` to build scATAC-seq references. Please follow the instructions below.

1. Import `cellranger_atac_create_reference`

Import *cellranger_atac_create_reference* workflow to your workspace by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Cellranger_atac_create_reference** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *cellranger_atac_create_reference* workflow in the drop-down menu.

2. Upload required data to Google Bucket

Required data include config JSON file, genome FASTA file, gene annotation file (GTF or GFF3 format) and motif input file (JASPAR format).

3. Workflow input

Required inputs are highlighted in bold.

Name	Description	Example	Default
genome	Genome reference name. New reference will be stored in a folder named **genome**	refdata-cellranger-atac-mm10-1.1.0	
input_fasta	GS URL for input fasta file	"gs://fc-e0000000-0000-0000-0000-000000000000/GRCh38.fa"	
input_gtf	GS URL for input GTF file	"gs://fc-e0000000-0000-0000-0000-000000000000/annotation.gtf"	
organism	Name of the organism	"human"	
non_nuclear_contigs	A comma separated list of names of contigs that are not in nucleus	"chrM"	"chrM"
input_motifs	Optional file containing transcription factor motifs in JASPAR format	"gs://fc-e0000000-0000-0000-0000-000000000000/motifs.pfm"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_atac_reference"	
cellranger_atac_version	cellranger-atac version, could be: 2.0.0, 1.2.0, 1.1.0	"2.0.0"	"2.0.0"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
memory	Memory size string for cellranger-atac mkref	"32G"	"32G"
disk_space	Optional disk space in GB	100	100
backend	Cloud backend for file transfer. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS; • "local" for local machine.	"gcp"	"gcp"
preemptible	Number of preemptible tries	2	2
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

4. Workflow output

Name	Type	Description
output_reference	File	Gzipped reference folder with name *genome.tar.gz*. We will also store a copy of the gzipped tarball under **output_directory** specified in the input.

Build references for single-cell immune profiling data

We provide a wrapper of `cellranger mkvdjref` to build single-cell immune profiling references. Please follow the instructions below.

1. Import `cellranger_vdj_create_reference`

Import *cellranger_vdj_create_reference* workflow to your workspace by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Cellranger_vdj_create_reference** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *cellranger_vdj_create_reference* workflow in the drop-down menu.

2. Upload requred data to Google Bucket

Required data include genome FASTA file and gene annotation file (GTF format).

3. Workflow input

Required inputs are highlighted in bold.

Name	Description	Example	Default
input_fasta	Input genome reference in either FASTA or FASTA.gz format	"gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.dna.toplevel.fa.gz"	
input_gtf	Input gene annotation file in either GTF or GTF.gz format	"gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.94.chr_patch_hapl_scaff.gtf.gz"	
genome	Genome reference name. New reference will be stored in a folder named **genome**	refdata-cellranger-vdj-GRCh38-alts-ensembl-3.1.0	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/cellranger_vdj_reference"	
ref_version	reference version string	Ensembl v94	
cellranger_version	cellranger version, could be: `6.1.2`, `6.1.1`	"6.1.2"	"6.1.2"
docker_registry	Docker registry to use for cellranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
memory	Memory size string for cellranger mkvdjref	"32G"	"32G"
disk_space	Optional disk space in GB	100	100
backend	Cloud backend for file transfer. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS; • "local" for local machine.	"gcp"	"gcp"
preemptible	Number of preemptible tries	2	2
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

4. Workflow output

Name	Type	Description
output_reference	File	Gzipped reference folder with name *genome.tar.gz*. We will also store a copy of the gzipped tarball under **output_directory** specified in the input.

1.1.6 Run Space Ranger tools using spaceranger_workflow

`spaceranger_workflow` wraps Space Ranger to process spatial transcriptomics data.

A general step-by-step instruction

This section mainly considers jobs starting from BCL files. If your job starts with FASTQ files, and only need to run `spaceranger count` part, please refer to this subsection.

1. Import `spaceranger_workflow`

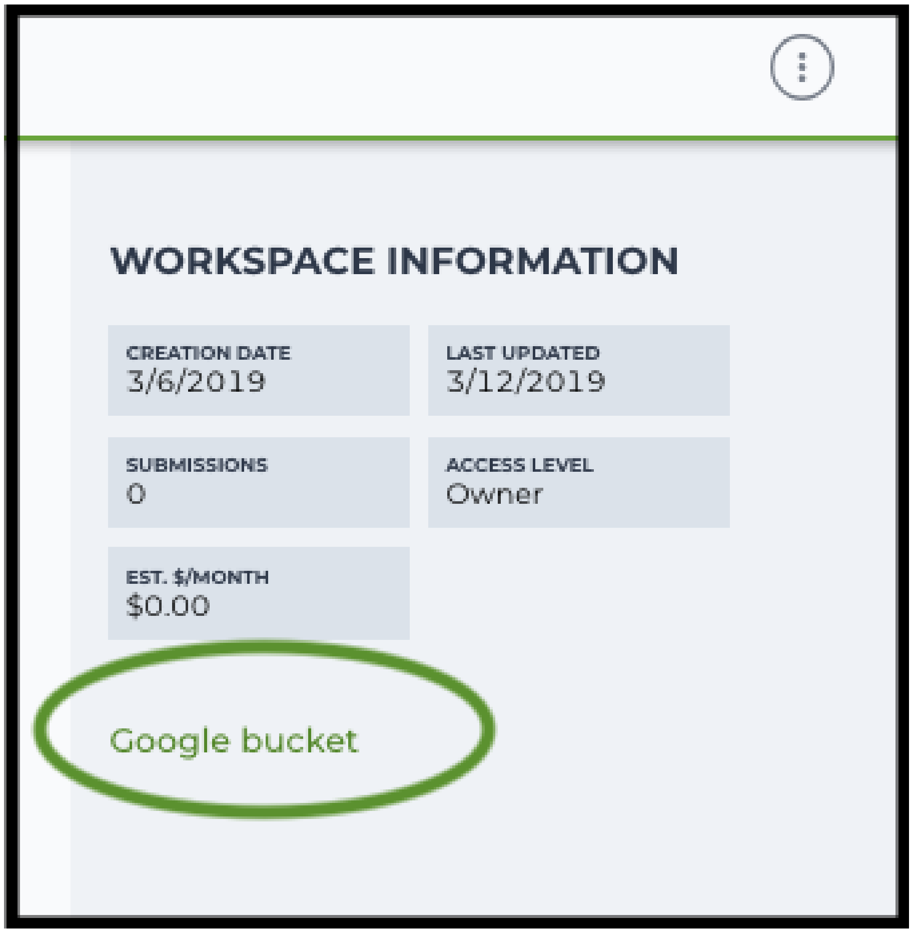
Import *spaceranger_workflow* workflow to your workspace by following instructions in Import workflows to Terra. You should choose workflow **github.com/lilab-bcb/cumulus/Spaceranger** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *spaceranger_workflow* workflow in the drop-down menu.

2. Upload sequencing and image data to Google bucket

Copy your sequencing output to your workspace bucket using gsutil (you already have it if you've installed Google cloud SDK) in your unix terminal.

You can obtain your bucket URL in the dashboard tab of your Terra workspace under the information panel.


WORKSPACE INFORMATION

CREATION DATE
3/6/2019

LAST UPDATED
3/12/2019

SUBMISSIONS
0

ACCESS LEVEL
Owner

EST. $/MONTH
$0.00

Google bucket

Use `gsutil cp [OPTION]... src_url dst_url` to copy data to your workspace bucket. For example, the following command copies the directory at /foo/bar/nextseq/Data/VK18WBC6Z4 to a Google bucket:

```
gsutil -m cp -r /foo/bar/nextseq/Data/VK18WBC6Z4 gs://fc-e0000000-0000-0000-
↪0000-000000000000/VK18WBC6Z4
```

`-m` means copy in parallel, `-r` means copy the directory recursively, and `gs://fc-e0000000-0000-0000-0000-000000000000` should be replaced by your own workspace Google bucket URL.

Similarly, copy all images for spatial data to the same google bucket.

Note: If input is a folder of BCL files, users do not need to upload the whole folder to the Google bucket. Instead, they only need to upload the following files:

```
RunInfo.xml
RTAComplete.txt
runParameters.xml
Data/Intensities/s.locs
Data/Intensities/BaseCalls
```

If data are generated using MiSeq or NextSeq, the location files are inside lane subfloders `L001` under `Data/Intensities/`. In addition, if users' data only come from a subset of lanes (e.g. `L001` and `L002`), users only need to upload lane subfolders from the subset (e.g. `Data/Intensities/BaseCalls/L001, Data/`

`Intensities/BaseCalls/L002` and `Data/Intensities/L001, Data/Intensities/L002` if sequencer is MiSeq or NextSeq).

Alternatively, users can submit jobs through command line interface (CLI) using altocumulus, which will smartly upload BCL folders according to the above rules.

3. Prepare a sample sheet

3.1 Sample sheet format:

Please note that the columns in the CSV can be in any order, but that the column names must match the recognized headings.

For **FFPE** data, **ProbeSet** column is mandatory.

The sample sheet describes how to demultiplex flowcells and generate channel-specific count matrices. Note that *Sample*, *Lane*, and *Index* columns are defined exactly the same as in 10x's simple CSV layout file.

A brief description of the sample sheet format is listed below **(required column headers are shown in bold)**.

Column	Description
Sample	Contains sample names. Each 10x channel should have a unique sample name.
Reference	Provides the reference genome used by Space Ranger for each 10x channel. The elements in the *reference* column can be either Google bucket URLs to reference tarballs or keywords such as *GRCh38-2020-A*. A full list of available keywords is included in each of the following data type sections (e.g. sc/snRNA-seq) below.
Flowcell	Indicates the Google bucket URLs of uploaded BCL folders. If starts with FASTQ files, this should be Google bucket URLs of uploaded FASTQ folders. The FASTQ folders should contain one subfolder for each sample in the flowcell with the sample name as the subfolder name. Each subfolder contains FASTQ files for that sample.
Lane	Tells which lanes the sample was pooled into. Can be either single lane (e.g. 8) or a range (e.g. 7-8) or all (e.g. *).
Index	Sample index (e.g. SI-GA-A12).
ProbeSet	Probe set for FFPE samples. **Choosing** from `human_probe_v1` (10x human probe set) and `mouse_probe_v1` (10x mouse probe set). Alternatively, a CSV file describing the probe set can be directly used. Setting ProbeSet to "" for a sample implies the sample is not FFPE.
Image	Google bucket url for a brightfield tissue H&E image in .jpg or .tiff format. This column is mutually exclusive with DarkImage and ColorizedImage columns.
DarkImage	Google bucket urls for Multi-channel, dark-background fluorescence image as either a single, multi-layer .tiff file, multiple .tiff or .jpg files, or a pre-combined color .tiff or .jpg file. If multiple files are provided, please separate them by ';'. This column is mutually exclusive with Image and ColorizedImage columns.
ColorizedImage	Google bucket url for a color composite of one or more fluorescence image channels saved as a single-page, single-file color .tiff or .jpg. This column is mutually exclusive with Image and DarkImage columns.
Slide	Visium slide serial number. If both Slide and Area are empty, the –unknown-slide option would be set.
Area	Visium capture area identifier. Options for Visium are A1, B1, C1, D1. If both Slide and Area are empty, the –unknown-slide option would be set.
SlideFile	Slide layout file indicating capture spot and fiducial spot positions. Only required if internet access is not available.
ReorientImages	**Valid values**: `true` or `false`. Use with automatic image alignment to specify that images may not be in canonical orientation with the hourglass in the top left corner of the image. The automatic fiducial alignment will attempt to align any rotation or mirroring of the image.
LoupeAlignment	Alignment file produced by the manual Loupe alignment step. Image column must be supplied in this case.
TargetPanel	Google bucket url for a target panel CSV for targeted gene expression analysis.

The sample sheet supports sequencing the same 10x channels across multiple flowcells. If a sample is sequenced across multiple flowcells, simply list it in multiple rows, with one flowcell per row. In the following example, we have 2 samples sequenced in two flowcells.

Example:

```
Sample,Reference,Flowcell,Lane,Index,Image,Slide,Area
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,1-2,SI-GA-A8,gs://image/image1.tif,V19J25-123,A1
sample_2,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK18WBC6Z4,3-4,SI-GA-B8,gs://image/image2.tif,V19J25-123,B1
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9Z2,1-2,SI-GA-A8,gs://image/image1.tif,V19J25-123,A1
sample_2,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪VK10WBC9Z2,3-4,SI-GA-B8,gs://image/image2.tif,V19J25-123,B1
```

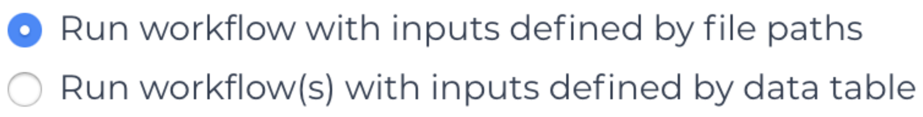
3.2 Upload your sample sheet to the workspace bucket:

Example:

```
gsutil cp /foo/bar/projects/sample_sheet.csv gs://fc-e0000000-0000-
↪0000-0000-000000000000/
```

4. Launch analysis

In your workspace, open `spaceranger_workflow` in `WORKFLOWS` tab. Select the desired snapshot version (e.g. latest). Select `Run workflow with inputs defined by file paths` as below


Run workflow with inputs defined by file paths
Run workflow(s) with inputs defined by data table

and click `SAVE` button. Select `Use call caching` and click `INPUTS`. Then fill in appropriate values in the `Attribute` column. Alternative, you can upload a JSON file to configure input by clicking `Drag or click to upload json`.

Once INPUTS are appropriated filled, click `RUN ANALYSIS` and then click `LAUNCH`.

5. Notice: run `spaceranger mkfastq` if you are non Broad Institute users

Non Broad Institute users that wish to run `spaceranger mkfastq` must create a custom docker image that contains `bcl2fastq`.

See *bcl2fastq* instructions.

6. Run `spaceranger count` only

Sometimes, users might want to perform demultiplexing locally and only run the count part on the cloud. This section describes how to only run the count part via `spaceranger_workflow`.

1. Copy your FASTQ files to the workspace using gsutil in your unix terminal. There are two cases:

- **Case 1**: All the FASTQ files are in one top-level folder. Then you can simply upload this folder to Cloud, and in your sample sheet, make sure **Sample** names are consistent with the filename prefix of their corresponding FASTQ files.
- **Case 2**: In the top-level folder, each sample has a dedicated subfolder containing its FASTQ files. In this case, you need to upload the whole top-level folder, and in your sample sheet, make sure **Sample** names and their corresponding subfolder names are identical.

Notice that if your FASTQ files are downloaded from the Sequence Read Archive (SRA) from NCBI, you must rename your FASTQs to follow the bcl2fastq file naming conventions.

Example:

```
gsutil -m cp -r /foo/bar/fastq_path/K18WBC6Z4 gs://fc-e0000000-0000-0000-
↪0000-000000000000/K18WBC6Z4_fastq
```

2. Create a sample sheet following the similar structure as above, except the following differences:
   - **Flowcell** column should list Google bucket URLs of the FASTQ folders for flowcells.
   - **Lane** and **Index** columns are NOT required in this case.

   Example:

```
Sample,Reference,Flowcell,Image,Slide,Area
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/
↪K18WBC6Z4_fastq,gs://image/image1.tif,V19J25-123,A1
```

3. Set optional input `run_mkfastq` to `false`.

Visium spatial transcriptomics data

To process spatial transcriptomics data, follow the specific instructions below.

Sample sheet

1. **Reference** column.

   Pre-built scRNA-seq references are summarized below.

Keyword	Description
GRCh38-2020-A	Human GRCh38 (GENCODE v32/Ensembl 98)
mm10-2020-A	Mouse mm10 (GENCODE vM23/Ensembl 98)

Workflow input

For spatial data, `spaceranger_workflow` takes Illumina outputs and related images as input and runs `spaceranger mkfastq` and `spaceranger count`. Revalant workflow inputs are described below, with required inputs highlighted in bold.

Name	Description	Example	Default
input_csv_file	Sample Sheet (contains Sample, Reference, Flowcell, Lane, Index as required and ProbeSet, Image, DarkImage, ColorizedImage, Slide, Area, SlideFile, ReorientImages, LoupeAlignment, TargetPanel as optional)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.csv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/spaceranger_output"	Results are written under directory *output_directory* and will overwrite any existing files at this location.
run_mkfastq	If you want to run `spaceranger mkfastq`	true	true
run_count	If you want to run `spaceranger count`	true	true
delete_input_bcl_directory	If delete BCL directories after demux. If false, you should delete this folder yourself so as to not incur storage charges	false	false
mkfastq_barcode_mismatches	Number of mismatches allowed in matching barcode indices (bcl2fastq2 default is 1)	0	
no_bam	Turn this option on to disable BAM file generation.	false	false
secondary	Perform Space Ranger secondary analysis (dimensionality reduction, clustering, etc.)	false	false
spaceranger_version	spaceranger version, could be: `1.3.1`, `1.3.0`	"1.3.1"	"1.3.1"
config_version	config docker version used for processing sample sheets, could be 0.2, 0.1	"0.2"	"0.2"
docker_registry	Docker registry to use for spaceranger_workflow. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
spaceranger_mkfastq_docker_registry	Docker registry to use for `spaceranger mkfastq`. Default is the registry to which only Broad users have access. See *bcl2fastq* for making your own registry.	"gcr.io/broad-cumulus"	"gcr.io/broad-cumulus"
zones	Google cloud zones	"us-central1-a us-west1-a"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of cpus to request for one node for spaceranger mkfastq and spaceranger count	32	32
memory	Memory size string for spac-	"120G"	"120G"

Workflow output

See the table below for important sc/snRNA-seq outputs.

Name	Type	Description
fastq_outputs	Array[String]?	A list of cloud urls containing FASTQ files, one url per flowcell.
count_outputs	Array[String]?	A list of cloud urls containing spaceranger count outputs, one url per sample.
metrics_summaries	File?	A excel spreadsheet containing QCs for each sample.
spaceranger_count.output_web_summary	Array[File]?	A list of htmls visualizing QCs for each sample (spaceranger count output).

Build Space Ranger References

Reference built by Cell Ranger for sc/snRNA-seq should be compatible with Space Ranger. For more details on building references using Cell Ranger, please refer to here.

1.1.7 Run STARsolo to generate gene-count matrices from FASTQ files

This `starsolo_workflow` workflow generates gene-count matrices from FASTQ data using STARsolo.

Prepare input data and import workflow

1. Run `cellranger_workflow` to generate FASTQ data

You can skip this step if your data are already in FASTQ format.

Otherwise, for 10X data, you need to first run *cellranger_workflow* to generate FASTQ files from BCL raw data for each sample. Please follow cellranger_workflow manual.

Notice that you should set **run_mkfastq** to `true` to get FASTQ output. You can also set **run_count** to `false` to skip Cell Ranger count step.

For Non-Broad users, you'll need to build your own docker for *bcl2fastq* step. Instructions are here.

2. Import `starsolo_workflow`

Import *starsolo_workflow* workflow to your workspace by following instructions in Import workflows to Terra. You should choose workflow **github.com/lilab-bcb/cumulus/STARsolo** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *starsolo_workflow* in the drop-down menu.

3. Prepare a sample sheet

3.1 Sample sheet format:

Please note that the columns in the CSV can be in any order, but that the column names must match the recognized headings.

The sample sheet describes how to identify flowcells and generate sample/channel-specific count matrices.

A brief description of the sample sheet format is listed below **(required column headers are shown in bold)**.

Column	Description
Sample	Contains the sample name. Each sample should have a unique sample name.
Reference	Provides the reference genome used by STARSolo for each sample. The elements in this column can be either Cloud bucket URIs to reference tarballs or keywords such as *GRCh38-2020-A*. A full list of available keywords is included in genome reference section below.
Location	Indicates the Cloud bucket URI of the folder holding FASTQ files of each sample.
Assay	Indicates the assay type of each sample. Available options: • `tenX_v3` for 10x 3’ v3 • `tenX_multiome` for 10x multiome • `tenX_v2` for 10x 3’ v2 • `tenX_5p` for 10x 5’ (only use R2 for alignment; equivalent to 10x chemistry *SC5P-R2*) • `tenX_5p_pe` for 10x 5’ (use both R1 and R2 for alignment, and R1 has length longer than 39 nt; equivalent to 10x chemistry *SC5P-PE*) • `DropSeq` • `SeqWell` • `SlideSeq` • `ShareSeq` • `None` If not specified, use the default `tenX_v3`.

3.2 Assay-specific preset STARsolo options

If **tenX_v3**, The following STARsolo options would be applied (could be overwritten by user-specified options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 16 --soloUMIstart 17 --
↪soloUMIlen 12 --soloCBmatchWLtype 1MM_multi_Nbase_pseudocounts --soloUMIfiltering␣
↪MultiGeneUMI_CR --soloUMIdedup 1MM_CR --clipAdapterType CellRanger4 --
↪outFilterScoreMin 30 --outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR␣
↪CY UY CB UB
```

If **tenX_multiome**, use the same STARsolo options as for *tenX_v3* assay, but with the 10X ARC Multiome Gene Expression whitelist.

If **tenX_v2**, the following STARsolo options would be applied (could be overwritten by user-specified options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 16 --soloUMIstart 17 --
↪soloUMIlen 10 --soloCBmatchWLtype 1MM_multi_Nbase_pseudocounts --soloUMIfiltering␣
↪MultiGeneUMI_CR --soloUMIdedup 1MM_CR --clipAdapterType CellRanger4 --
↪outFilterScoreMin 30 --outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR␣
↪CY UY CB UB
```

If **tenX_5p**, the following STARsolo options would be applied (could be overwritten by user-specified options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 16 --soloUMIstart 17 --
↪soloUMIlen 10 --soloCBmatchWLtype 1MM_multi_Nbase_pseudocounts --soloUMIfiltering␣
↪MultiGeneUMI_CR --soloStrand Reverse --soloUMIdedup 1MM_CR --outFilterScoreMin 30 --
↪outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR CY UY CB UB
```

If **tenX_5p_pe**, the following STARsolo options would be applied (could be overwritten by user-specified options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 16 --soloUMIstart 17 --
↪soloUMIlen 10 --soloCBmatchWLtype 1MM_multi_Nbase_pseudocounts --soloUMIfiltering␣
↪MultiGeneUMI_CR --soloBarcodeMate 1 --clip5pNbases 39 0 --soloUMIdedup 1MM_CR --
↪outFilterScoreMin 30 --outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR␣
↪CY UY CB UB
```

If **ShareSeq**, the following STARsolo options would be applied (could be overwritten by user-specific options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 24 --soloUMIstart 25 --
↪soloUMIlen 10 --soloCBmatchWLtype 1MM_multi_Nbase_pseudocounts --soloUMIfiltering␣
↪MultiGeneUMI_CR --soloUMIdedup 1MM_CR --clipAdapterType CellRanger4 --
↪outFilterScoreMin 30 --outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR␣
↪CY UY CB UB
```

If **SeqWell** or **DropSeq**, the following STARsolo options would be applied (could be overwritten by user-specified options):

```
--soloType CB_UMI_Simple --soloCBstart 1 --soloCBlen 12 --soloUMIstart 13 --
↪soloUMIlen 8 --outSAMtype BAM SortedByCoordinate --outSAMattributes CR UR CY UY CB␣
↪UB
```

If **None**, no preset option would be applied.

The sample sheet supports sequencing the same sample across multiple flowcells. In case of multiple flowcells, you should specify one line for each flowcell using the same sample name. In the following example, we have 2 samples and `sample_1` is sequenced in two flowcells.

Example:

```
Sample,Reference,Location,Assay
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4/sample_
↪1_fastqs,tenX_v3
sample_1,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2/sample_
↪1_fastqs,tenX_v3
sample_2,GRCh38-2020-A,gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4/sample_
↪2_fastqs,tenX_v2
```

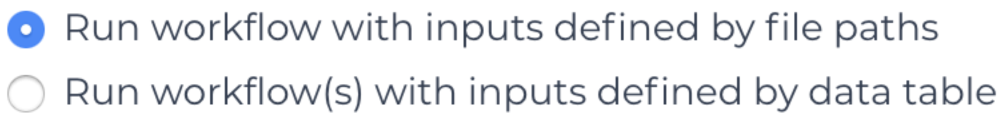
3.2 Upload your sample sheet to the workspace bucket:

Example:

```
gsutil cp /foo/bar/projects/sample_sheet.csv gs://fc-e0000000-0000-0000-0000-
↪000000000000/
```

1. Launch analysis

In your workspace, open `starsolo_workflow` in `WORKFLOWS` tab. Select the desired snapshot version (e.g. latest). Select `Process single workflow from files` as below



Run workflow with inputs defined by file paths [✓]

Run workflow(s) with inputs defined by data table

and click `SAVE` button. Select `Use call caching` and click `INPUTS`. Then fill in appropriate values in the `Attribute` column. Alternative, you can upload a JSON file to configure input by clicking `Drag or click to upload json`.

Once INPUTS are appropriated filled, click `RUN ANALYSIS` and then click `LAUNCH`.

Workflow inputs

Below are inputs for *count* workflow. Notice that required inputs are in bold.

Name	Description	Example	Default
input_csv_file	Input CSV sample sheet describing metadata of each sample.	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.tsv"	
output_directory	Cloud bucket URI of output directory.	"gs://fc-e0000000-0000-0000-0000-000000000000/count_result"	
read1_fastq_pattern	Filename suffix pattern in wildcards for Read 1. This is used for looking for Read 1 fastq files. If fastq files are generated by CellRanger count, use `_S*_L*_R1_001.fastq.gz`, which means Read 1 files must have names such as "<Sample>_S1_L1_R1_001.fastq.gz", where *<Sample>* is specified in **input_csv_file**. If fastq files are Sequence Read Archive (SRA) data, use something like `_1.fastq.gz`, where `_1` refers to the first reads, so that Read 1 files must have names such as "<Sample>_1.fastq.gz" where *<Sample>* is specified in **input_csv_file**. If fastq files are not zipped, substitute `.fastq` for `.fastq.gz` in the corresponding pattern above.	"_S*_L*_R1_001.fastq.gz"	"_S*_L*_R1_001.fastq.gz"

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
read2_fastq_pattern	Filename suffix pattern in wildcards for Read 2. This is used for looking for Read 2 fastq files. If fastq files are generated by CellRanger count, use `_S*_L*_R2_001.fastq.gz`, which means Read 2 files must have names such as "<Sample>_S1_L1_R2_001.fastq.gz", where *<Sample>* is specified in **input_csv_file**. If fastq files are Sequence Read Archive (SRA) data, use something like `_2.fastq.gz`, where `_2` refers to the second reads, so that Read 2 files must have names such as "<Sample>_2.fastq.gz" where *<Sample>* is specified in **input_csv_file**. If fastq files are not zipped, substitute `.fastq` for `.fastq.gz` in the corresponding pattern above.	"_S*_L*_R2_001.fastq.gz"	"_S*_L*_R2_001.fastq.gz"
barcode_read	Specify which read contains cell barcodes and UMIs: either `read1` or `read2`. This only applies to samples with *Assay* `None` in **input_csv_file**. Otherwise, samples with *Assay* type `ShareSeq` automatically specify `read2` for cell barcodes and UMIs, while `read1` for cDNAs; samples of all the other know *Assay* types automatically specify `read1` for cell barcodes and UMIs, while `read2` for cDNAs.	"read1"	"read1"
soloType	[STARsolo option] Type of single-cell RNA-seq, choosing from *CB_UMI_Simple*, *CB_UMI_Complex*, *CB_samTagOut*, *SmartSeq*.	"CB_UMI_Simple"	None
soloCBwhitelist	[STARsolo option] Cell barcode white list in either plain text or gzipped format. **Notice:** If specified, it will overwrite the white lists for **ALL** the samples in your sample sheet.	gs://my_bucket/my_white_list.txt	None
soloFeatures	[STARsolo option] Genomic features for which the UMI counts per Cell Barcode are collected (can choose multiple items): • *Gene*: reads match the gene transcript • *SJ*: splice junctions reported in SJ.out.tab • *GeneFull*: count all reads overlapping genes' exons and introns • *Velocyto*: calculate Spliced, Unspliced, and Ambiguous counts per cell per gene similar to the velocyto.py tool developed by LaManno et al. Note that *Velocyto* requires *Gene*.	"Gene GeneFull SJ Velocyto"	"Gene"

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
soloMultiMappers	[STARsolo option] Counting method for reads mapping to multiple genes (can choose multiple items): • *Unique*: count only reads that map to unique genes • *Uniform*: uniformly distribute multi-genic UMIs to all genes • *Rescue*: distribute UMIs proportionally to unique+uniform counts (first iteartion of EM) • *PropUnique*: distribute UMIs proportionally to unique mappers, if present, and uniformly if not • *EM*: use Maximum Likelihood Estimation (MLE) to distribute multi-gene UMIs among their genes	"Unique"	"Unique"
soloCBstart	[STARsolo option] Cell barcode start position (1-based coordinate).	1	1
soloCBlen	[STARsolo option] Cell barcode length.	16	16
soloUMIstart	[STARsolo option] UMI start position (1-based coordinate).	17	17
soloUMIlen	[STARsolo option] UMI length.	10	10
soloBarcodeReadLength	[STARsolo option] Length of the barcode read - 1: equals to sum of *soloCBlen* and *soloUMIlen*. - 0: not defined, do not check. **Notice:** `0` is set to be default, which is different from STAR. This is in case users have barcode read sequenced of length 28 nt (standard for 10x 3'), but assay is 5' (CB+UMI length is 26 nt).	0	0
soloBarcodeMate	[STARsolo option] Identifies which read mate contains the barcode (CB+UMI) sequence: • 0: barcode sequence is on separate read, which should always be the last file in the input Read1 file list • 1: barcode sequence is a part of mate 1 • 2: barcode sequence is a part of mate 2	0	0

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
soloCBposition	[STARsolo option] Position of Cell Barcode(s) on the barcode read. Presently only works when *solo_type* is `CB_UMI_Complex`, and barcodes are assumed to be on Read2. Format for each barcode: "startAnchor_startPosition_endAnchor_endPosition" start(end)Anchor defines the Anchor Base for the CB: 0: read start; 1: read end; 2: adapter start; 3: adapter end start(end)Position is the 0-based position with of the CB start(end) with respect to the Anchor Base String for different barcodes are separated by space.	"0_0_2_-1 3_1_3_8"	
soloUMIposition	[STARsolo option] Position of the UMI on the barcode read, same as soloCBposition	"3_9_3_14"	
soloAdapterSequence	[STARsolo option] Adapter sequence to anchor barcodes.		
soloAdapterMismatchesNmax	[STARsolo option] Maximum number of mismatches allowed in adapter sequence.	1	1
soloCBmatchWLtype	[STARsolo option] Matching the Cell Barcodes to the WhiteList, choosing from • *Exact*: only exact matches allowed • *1MM*: only one match in whitelist with 1 mismatched base allowed. Allowed CBs have to have at least one read with exact match • *1MM_multi*: multiple matches in whitelist with 1 mismatched base allowed, posterior probability calculation is used choose one of the matches. Allowed CBs have to have at least one read with exact match. This option matches best with CellRanger 2.2.0 • *1MM_multi_pseudocounts*: same as *1MM_multi*, but pseudocounts of 1 are added to all whitelist barcodes • *1MM_multi_Nbase_pseudocounts*: same as *1MM_multi_pseudocounts*, multimatching to WL is allowed for CBs with N-bases. This option matches best with CellRanger >= 3.0.0	"1MM_multi"	"1MM_multi"
soloInputSAMattrBarcodeSeq	[STARsolo option] When inputting reads from a SAM file (`--readsFileType SAM SE/PE`), these SAM attributes mark the barcode qualities (in proper order). For instance, for 10X CellRanger or STARsolo BAMs, use `--soloInputSAMattrBarcodeSeq CR UR`. This parameter is required when running STARsolo with input from SAM.	"CR UR"	

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
soloInputSAMattrBarcodeQual	[STARsolo option] When inputting reads from a SAM file (`--readsFileType SAM SE/PE`), these SAM attributes mark the barcode sequence (in proper order). For instance, for 10X CellRanger or STARsolo BAMs, use `--soloInputSAMattrBarcodeQual CY UY`. If this parameter is `-` (default), the quality 'H' will be assigned to all bases.	"CY UY"	
soloStrand	[STARsolo option] Strandedness of the solo libraries: • *Unstranded*: no strand information • *Forward*: read strand same as the original RNA molecule • *Reverse*: read strand opposite to the original RNA molecule	"Forward"	"Forward"
soloUMIdedup	[STARsolo option] Type of UMI deduplication (collapsing) algorithm: • *1MM_All*: all UMIs with 1 mismatch distance to each other are collapsed (i.e. counted once) • *1MM Directional UMItools*: follows the "directional" method from the UMI-tools by Smith, Heger and Sudbery (Genome Research 2017) • *1MM Directional*: same as 1MM Directional UMItools, but with more stringent criteria for duplicate UMIs • *Exact*: only exactly matching UMIs are collapsed • *NoDedup*: no deduplication of UMIs, count all reads • *1MM CR*: CellRanger2-4 algorithm for 1MM UMI collapsing	"1MM_All"	"1MM_All"
soloUMIfiltering	[STARsolo option] Type of UMI filtering (for reads uniquely mapping to genes): • -: basic filtering: remove UMIs with N and homopolymers (similar to CellRanger 2.2.0) • *MultiGeneUMI*: basic + remove lower-count UMIs that map to more than one gene • *MultiGeneUMI_All*: basic + remove all UMIs that map to more than one gene • *MultiGeneUMI_CR*: basic + remove lower-count UMIs that map to more than one gene, matching CellRanger > 3.0.0. Only works with `--soloUMIdedup 1MM CR`	"MultiGeneUMI"	"-"

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
soloCellFilter	[STARsolo option] Cell filtering type and parameters: • *None*: do not output filtered cells • *TopCells*: only report top cells by UMI count, followed by the exact number of cells • *CellRanger2.2*: simple filtering of CellRanger 2.2. Can be followed by numbers: number of expected cells, robust maximum percentile for UMI count, maximum to minimum ratio for UMI count. The harcoded values are from CellRanger: nExpectedCells=3000; maxPercentile=0.99; maxMinRatio=10 • *EmptyDrops_CR*: EmptyDrops filtering in CellRanger flavor. Please cite the original EmptyDrops paper: A.T.L Lun et al, Genome Biology, 20, 63 (2019): https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1662-y. Can be followed by 10 numeric parameters: nExpectedCells maxPercentile maxMinRatio indMin indMax umiMin umiMinFracMedian candMaxN FDR simN. The harcoded values are from CellRanger: 3000 0.99 10 45000 90000 500 0.01 20000 0.01 10000	"CellRanger2.2 3000 0.99 10"	"CellRanger2.2 3000 0.99 10"
soloOutFormatFeaturesGeneField3	[STARsolo option] Field 3 in the Gene features.tsv file. If "-", then no 3rd field is output.	"Gene Expression"	"Gene Expression"
outSAMtype	[STAR option] Type of SAM/BAM output.	"BAM SortedByCoordinate"	"BAM SortedByCoordinate" for *tenX_v3*, *tenX_v2*, *SeqWell* and *DropSeq* assay types, "BAM Unsorted" otherwise.
star_version	STAR version to use. Currently support: `2.7.9a`.	"2.7.9a"	"2.7.9a"
docker_registry	Docker registry to use: • `quay.io/cumulus` for images on Red Hat registry; • `cumulusprod` for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"

Continued on next page

Table 2 – continued from previous page

Name	Description	Example	Default
zones	Google cloud zones to consider for execution.	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of CPUs to request for count per sample.	32	32
memory	Memory size string for count per sample.	"120G"	"120G"
disk_space	Disk space in GB needed for count per sample.	500	500
backend	Cloud infrastructure backend to use. Available options: • `gcp` for Google Cloud; • `aws` for Amazon AWS; • `local` for local machine.	"gcp"	"gcp"
preemptible	Number of maximum preemptible tries allowed. This works only when *backend* is `gcp`.	2	2
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

Workflow outputs

See the table below for *star_solo* workflow outputs.

Name	Type	Description
output_folder	String	Google Bucket URI of output directory. Within it, each folder is for one sample in the input sample sheet.
starsoloLogs	Array[File]	Google Bucket URIs of STAR logs for each sample, respectively. This is the `Log.out` if running STAR locally, which is important for debugging.

Prebuilt genome references

We've built the following scRNA-seq references for users' convenience:

Keyword	Description
GRCh38-2020-A	Human GRCh38, comparable to cellranger reference 2020-A (GENCODE v32/Ensembl 98)
mm10-2020-A	Mouse mm10, comparable to cellranger reference 2020-A (GENCODE vM23/Ensembl 98)
GRCh38-and-mm10-2020-A	Human GRCh38 (GENCODE v32/Ensembl 98) and mouse mm10 (GENCODE vM23/Ensembl 98)

Build STARSolo References

We provide a wrapper of STAR to build sc/snRNA-seq references. Please follow the instructions below.

1. Import `starsolo_create_reference`

Import *starsolo_create_reference* workflow to your workspace by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/STARsolo_create_reference** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *starsolo_create_reference* workflow in the drop-down menu.

2. Upload required data to Cloud bucket

Required data include the genome FASTA file and gene annotation GTF file of the target genome reference.

3. Workflow input

Required inputs are highlighted **in bold**.

Name	Description	Example	Default
input_fasta	Input genome reference in FASTA format.	"gs://fc-e0000000-0000-0000-0000-000000000000/mm-10/genome.fa"	
input_gtf	Input gene annotation file in GTF format.	"gs://fc-e0000000-0000-0000-0000-000000000000/mm-10/genes.gtf"	
genome	Genome reference name. This is used for specifying the name of the genome index generated.	"mm-10"	
output_directory	Cloud bucket URI of the output directory.	"gs://fc-e0000000-0000-0000-0000-000000000000/starsolo-reference"	
docker_registry	Docker registry to use: • `quay.io/cumulus` for images on Red Hat registry; • `cumulusprod` for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
star_version	STAR version to use. Currently support: `2.7.9a`.	"2.7.9a"	"2.7.9a"
num_cpu	Number of CPUs to request for count per sample.	32	32
memory	Memory size string for count per sample.	"80G"	"80G"
disk_space	Disk space in GB needed for count per sample.	100	100
zones	Google cloud zones to consider for execution.	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
backend	Cloud infrastructure backend to use. Available options: • `gcp` for Google Cloud; • `aws` for Amazon AWS; • `local` for local machine.	"gcp"	"gcp"
preemptible	Number of maximum preemptible tries allowed. This works only when *backend* is `gcp`.	2	2
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

4. Workflow Output

Name	Type	Description
output_reference	File	Gzipped reference folder with name **"<genome>-starsolo.tar.gz"**, where *<genome>* is specified by workflow input **genome** above. The workflow will save a copy of it under **output_directory** specified in workflow input above.

1.1.8 Demultiplex genetic-pooling/cell-hashing/nucleus-hashing sc/snRNA-Seq data

This `demultiplexing` workflow generates gene-count matrices from cell-hashing/nucleus-hashing/genetic-pooling data by demultiplexing.

In the workflow, `demuxEM` is used for analyzing cell-hashing/nucleus-hashing data, while `souporcell` and `popscle` (including *demuxlet* and *freemuxlet*) are for genetic-pooling data.

Prepare input data and import workflow

1. Run `cellranger_workflow`

To demultiplex, you'll need raw gene count and hashtag matrices for cell-hashing/nucleus-hashing data, or raw gene count matrices and genome BAM files for genetic-pooling data. You can generate these data by running the `cellranger_workflow`.

Please refer to the cellranger_workflow tutorial for details.

When finished, you should be able to find the raw gene count matrix (e.g. `raw_gene_bc_matrices_h5.h5`), hashtag matrix (e.g. `sample_1_ADT.csv`) / genome BAM file (e.g. `possorted_genome_bam.bam`) for each sample.

2. Import `demultiplexing`

Import *demultiplexing* workflow to your workspace by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Demultiplexing** to import.

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *demultiplexing* workflow in the drop-down menu.

3. Prepare a sample sheet

3.1 Sample sheet format:

Create a sample sheet, **sample_sheet_demux.csv**, which describes the metadata for each pair of RNA and hashtag data. A brief description of the sample sheet format is listed below **(required column headers are shown in bold)**.

Column	Description
OUTNAME	Output name for one pair of RNA and hashtag data. Must be unique per pair.
RNA	Google bucket url to the raw gene count matrix generated in Step 1.
TagFile/ADT	Google bucket url to the hashtag file generated in Step 1. The column name can be either *TagFile* or *ADT*, where *ADT* is for backward compatibility with older snapshots.
TYPE	Assay type, which can be `cell-hashing`, `nucleus-hashing`, or `genetic-pooling`.
Genotype	Google bucket url to the reference genotypes in `vcf.gz` format. This column is **required** in the following cases: • Run `genetic-pooling` assay with `souporcell` algorithm (i.e. *TYPE* is `genetic-pooling`, *demultiplexing_algorithm* input is `souporcell`): – Run with reference genotypes, i.e. *souporcell_de_novo_mode* is `false`. – Run in *de novo* mode (i.e. *souporcell_de_novo_mode* is `true`), but need to match the resulting cluster names by information from reference genotypes (see description of *souporcell_rename_donors* input below). • Run `genetic-pooling` assay with `popscle` algorithm (i.e. *TYPE* is `genetic-pooling`, *demultiplexing_algorithm* input is `popscle`): – *popscle_num_samples* input is `0`. In this case, *demuxlet* will be run with reference genotypes. – *popscle_num_samples* input is larger than `0`. In this case, reference genotypes will be only used to generate pileups, then *freemuxlet* will be used for demultiplexing **without** reference genotypes.

Example:

```
OUTNAME,RNA,TagFile,TYPE,Genotype
sample_1,gs://exp/data_1/raw_gene_bc_matrices_h5.h5,gs://exp/data_1/sample_1_
↪ADT.csv,cell-hashing
sample_2,gs://exp/data_2/raw_gene_bc_matrices_h5.h5,gs://exp/data_2/sample_2_
↪ADT.csv,nucleus-hashing
sample_3,gs://exp/data_3/raw_gene_bc_matrices_h5.h5,gs://exp/data_3/
↪possorted_genome_bam.bam,genetic-pooling
sample_4,gs://exp/data_4/raw_gene_bc_matrices_h5.h5,gs://exp/data_4/
↪possorted_genome_bam.bam,genetic-pooling,gs://exp/variants/ref_genotypes.
↪vcf.gz
```

3.2 Upload your sample sheet to the workspace bucket:

Use gsutil (you already have it if you've installed gcloud CLI) in your unix terminal to upload your sample sheet to workspace bucket.

Example:

```
gsutil cp /foo/bar/projects/sample_sheet_demux.csv gs://fc-e0000000-0000-
↪0000-0000-000000000000/
```

Workflow inputs

Below are inputs for *demultiplexing* workflow. We'll first introduce global inputs, and then inputs for each of the demultiplexing tools. Notice that required inputs are in bold.

global inputs

Name	Description	Example	Default
input_sample_sheet	Input CSV file describing metadata of RNA and hashtag data pairing.	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet_demux.csv"	
output_directory	This is the output directory (gs url + path) for all results. There will be one folder per RNA-hashtag data pair under this directory.	"gs://fc-e0000000-0000-0000-0000-000000000000/demux_output"	
genome	Reference genome name. Its usage depends on the assay type: • For *cell-hashing* or *nucleus-hashing*, only write this name as an annotation into the resulting count matrix file. • For *genetic-pooling*, if *demultiplexing_algorithm* input is `souporcell`, you should choose one name from this genome reference list. • For *genetic-pooling*, if *demultiplexing_algorithm* input is `popscle`, reference genome name is not needed.	"GRCh38"	
demultiplexing_algorithm	Demultiplexing algorithm to use for *genetic-pooling* data. Options: • "souporcell": Use souporcell, a reference-genotypes-free algorithm for demultiplexing droplet scRNA-Seq data. • "popscle": Use popscle, a canonical algorithm for demultiplexing droplet scRNA-Seq data, including *demuxlet* (with reference genotypes) and *freemuxlet* (reference-genotype-free) components.	"souporcell"	"souporcell"
min_num_genes	Only demultiplex cells/nuclei with at least <min_num_genes> expressed genes	100	100
zones	Google cloud zones to consider for execution.	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
docker_registry	Docker registry to use. • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
config_version	Version of config docker image to use. This docker is used for parsing the input sample sheet for downstream execution. Available options: `0.2`, `0.1`.	"0.2"	"0.2"
backend	Cloud infrastructure backend to use. Available options:	"gcp"	"gcp"

demuxEM inputs

Name	Description	Example	Default
demuxEM_alpha_on_samples	demuxEM parameter. The Dirichlet prior concentration parameter (alpha) on samples. An alpha value < 1.0 will make the prior sparse.	0.0	0.0
demuxEM_min_num_umis	demuxEM parameter. Only demultiplex cells/nuclei with at least <demuxEM_min_num_umis> of UMIs.	100	100
demuxEM_min_signal_hashtag	demuxEM parameter. Any cell/nucleus with less than <demuxEM_min_signal_hashtag> hashtags from the signal will be marked as unknown.	10.0	10.0
demuxEM_random_state	demuxEM parameter. The random seed used in the KMeans algorithm to separate empty ADT droplets from others.	0	0
demuxEM_generate_diagnostic_plots	demuxEM parameter. If generate a series of diagnostic plots, including the background/signal between HTO counts, estimated background probabilities, HTO distributions of cells and non-cells, etc.	true	true
demuxEM_generate_gender_plot	demuxEM parameter. If generate violin plots using gender-specific genes (e.g. Xist). <demuxEM_generate_gender_plot> is a comma-separated list of gene names	"XIST"	
demuxEM_version	demuxEM version to use. Choose from "0.1.7", "0.1.6" and "0.1.5".	"0.1.7"	"0.1.7"
demuxEM_num_cpu	demuxEM parameter. Number of CPUs to request for demuxEM per pair.	8	8
demuxEM_memory	demuxEM parameter. Memory size string for demuxEM per pair.	"10G"	"10G"
demuxEM_disk_space	demuxEM parameter. Disk space (integer) in GB needed for demuxEM per pair.	20	20

souporcell inputs

Name	Description	Example	Default
souporcell_version	souporcell version to use. Available versions: • `2021.03`: Based on commitment 1bd9f1 on 2021/03/07. • `2020.07`: Based on commitment 0d09fb on 2020/07/27. • `2020.03`: Based on commitment eeddcd on 2020/03/31.	"2021.03"	"2021.03"
souporcell_num_clusters	souporcell parameter. Number of expected clusters when doing clustering. **This needs to be set when running souporcell.**	8	1
souporcell_de_novo_mode	souporcell parameter. • If `true`, run souporcell in de novo mode without reference genotypes: – If input *souporcell_common_variants* is further provided, use this common variants list instead of calling SNPs de novo. – If a reference genotype vcf file is provided in the sample sheet, use it **only** for matching the cluster labels computed by souporcell. • If `false`, run souporcell with `--known_genotypes` option using the reference genotype vcf file specified in sample sheet.	true	true
souporcell_num_clusters	souporcell parameter. Number of expected clusters when doing clustering. **This needs to be set when running souporcell.**	8	1
souporcell_common_variants	souporcell parameter. Users can provide a common variants list in VCF format for Souporcell to use, instead of calling SNPs de novo. **Notice:** This input is enabled only when *souporcell_de_novo_mode* is `false`.	"1000genome.common.variants.vcf.gz"	
souporcell_skip_remap	souporcell parameter. Skip remap step. Only recommended in non denovo mode or common variants are provided.	true	false
souporcell_rename_donors	souporcell parameter. A comma-separated list of donor names for matching clusters achieved by souporcell. Must be consistent with *souporcell_num_clusters* input. • If this input is empty, use cluster labels from the reference genotype vcf file if provided in the sample sheet; if this vcf file is not provided, simply name clusters as *Donor1*, *Donor2*, . . . • If this input is not empty, and a reference genotype vcf file is provided in the sample sheet, first match the cluster labels using those from this vcf file, then rename to donor names specified in this input.	"CB1,CB2,CB3,CB4"	

Popscle inputs

Name	Description	Example	Default
popscle_num_samples	popscle parameter. Number of samples to be multiplexed together: • If `0`, run with *demuxlet* using reference genotypes. • Otherwise, run with *freemuxlet* in de novo mode without reference genotypes.	4	0
popscle_min_MQ	popscle parameter. Minimum mapping quality to consider (lower MQ will be ignored).	20	20
popscle_min_TD	popscle parameter. Minimum distance to the tail (lower will be ignored).	0	0
popscle_tag_group	popscle parameter. Tag representing readgroup or cell barcodes, in the case to partition the BAM file into multiple groups. For 10x genomics, use `CB`.	“CB”	“CB”
popscle_tag_UMI	popscle parameter. Tag representing UMIs. For 10x genomics, use `UB`.	“UB”	“UB”
popscle_field	popscle parameter. FORMAT field to extract from: genotype (`GT`), genotype likelihood (`GL`), or posterior probability (`GP`).	“GT”	“GT”
popscle_alpha	popscle parameter. Grid of alpha to search for, in a comma separated list format of all alpha values to be considered.	“0.1,0.2,0.3,0.4,0.5”	“0.1,0.2,0.3,0.4,0.5”
popscle_rename_donors	popscle parameter. A comma-separated list of donor names for renaming clusters achieved by popscle. Must be consistent with *popscle_num_samples* input. By default, the resulting donors are *Donor1*, *Donor2*, . . .	“CB1,CB2,CB3,CB4”	
popscle_version	popscle parameter. popscle version to use. Available options: • `2021.05`: Based on commitment da70fc7 on 2021/05/05. • `0.1b`: Based on version 0.1-beta released on 2019/10/03.	“2021.05”	“2021.05”
popscle_num_cpu	popscle parameter. Number of CPU used by popscle per pair.	1	1
popscle_memory	popscle parameter. Memory size string per pair.	“120G”	“120G”
popscle_extra_disk_space	popscle parameter. Extra disk space size (integer) in GB needed for popscle per pair, besides the disk size required to hold input files specified in the sample sheet.	100	100

Workflow outputs

See the table below for *demultiplexing* workflow outputs.

Name	Type	Description
output_folders	Array[String]	A list of Google Bucket URLs of the output folders. Each folder is associated with one RNA-hashtag pair in the given sample sheet.
output_zarr_files	Array[File]	A list of demultiplexed RNA count matrices in zarr format. Each zarr file is associated with one RNA-hashtag pair in the given sample sheet. Please refere to section load demultiplexing results into Python and R for its structure.

In the output subfolder of each cell-hashing/nuclei-hashing RNA-hashtag data pair, you can find the following files:

Name	Description
output_name_demux.zarr.zip	Demultiplexed RNA raw count matrix in zarr format. Please refer to section load demultiplexing results into Python and R for its structure.
output_name.out.demuxEM.zarr.zip	This file contains intermediate results for both RNA and hashing count matrices. To load this file into Python, you need to first install Pegasusio on your local machine. Then use `import pegasusio as io; data = io.read_input("output_name.out.demuxEM.zarr.zip")` in Python environment. It contains 2 UnimodalData objects: one with key name suffix `-hashing` is the hashtag count matrix, the other one with key name suffix `-rna` is the demultiplexed RNA count matrix. To load the hashtag count matrix, type `hash_data = data.get_data('<genome>-hashing')`, where `<genome>` is the genome name of the data. The count matrix is `hash_data.X`; cell barcode attributes are stored in `hash_data.obs`; sample names are in `hash_data.var_names`. Moreover, the estimated background probability regarding hashtags is in `hash_data.uns['background_probs']`. To load the RNA matrix, type `rna_data = data.get_data('<genome>-rna')`, where `<genome>` is the genome name of the data. It only contains cells which have estimated sample assignments. The count matrix is `rna_data.X`. Cell barcode attributes are stored in `rna_data.obs`: `rna_data.obs['demux_type']` stores the estimated droplet types (singlet/doublet/unknown) of cells; `rna_data.obs['assignment']` stores the estimated hashtag(s) that each cell belongs to. Moreover, for cell-hashing/nucleus-hashing data, you can find estimated sample fractions (sample1, sample2, . . . , samplen, background) for each droplet in `rna_data.obsm['raw_probs']`.
output_name.ambient_hashtag.hist.png	Optional output. A histogram plot depicting hashtag distributions of empty droplets and non-empty droplets.
output_name.background_probabilities.bar.png	Optional output. A bar plot visualizing the estimated hashtag background probability distribution.
output_name.real_content.hist.png	Optional output. A histogram plot depicting hashtag distributions of not-real-cells and real-cells as defined by total number of expressed genes in the RNA assay.
output_name.rna_demux.hist.png	Optional output. A histogram plot depicting RNA UMI distribution for singlets, doublets and unknown cells.
output_name.gene_name.violin.png	Optional outputs. Violin plots depicting gender-specific gene expression across samples. We can have multiple plots if a gene list is provided in `demuxEM_generate_gender_plot` field of cumulus_hashing_cite_seq inputs.

In the output subfolder of each genetic-pooling RNA-hashtag data pair generated by *souporcell*, you can find the following files:

Name	Description
output_name_demux.zarr.zip	Demultiplexed RNA count matrix in zarr format. Please refer to section load demultiplexing results into Python and R for its structure.
clusters.tsv	Inferred droplet type and cluster assignment for each cell barcode.
cluster_genotypes.vcf	Inferred genotypes for each cluster.
match_donors.log	Log of matching donors step, with information of donor matching included.

In the output subfolder of each genetic-pooling RNA-hashtag data pair generated by *demuxlet*, you can find the following files:

Name	Description
output_name_demux.zarr.zip	Demultiplexed RNA count matrix in zarr format. Please refer to section load demultiplexing results into Python and R for its structure.
output_name.best (demuxlet) or output_name.clust1.samples.gz (freemuxlet)	Inferred droplet type and cluster assignment for each cell barcode.

Load demultiplexing results into Python and R

To load demultiplexed RNA count matrix into Python, you need to install Python package pegasusio first. Then follow the codes below:

```
import pegasusio as io
data = io.read_input('output_name_demux.zarr.zip')
```

Once you load the data object, you can find estimated droplet types (singlet/doublet/unknown) in `data.obs['demux_type']`. Notices that there are cell barcodes with no sample associated, and therefore have no droplet type.

You can also find estimated sample assignments in `data.obs['assignment']`.

For cell-hashing/nucleus-hashing data, if one sample name can correspond to multiple feature barcodes, each feature barcode is assigned to a unique sample name, and this deduplicated sample assignment results are in `data.obs['assignment.dedup']`.

To load the results into R, you need to install R package `reticulate` in addition to Python package `pegasusio`. Then follow the codes below:

```
library(reticulate)
ad <- import("pegasusio", convert = FALSE)
data <- ad$read_input("output_name_demux.zarr.zip")
```

Results are in `data$obs['demux_type']`, `data$obs['assignment']`, and similarly as above, for cell-hashing/nucleus-hashing data, you'll find an additional field `data$obs['assignment.dedup']` for deduplicated sample assignment in the case that one sample name can correspond to multiple feature barcodes.

1.1.9 Run Cumulus for sc/snRNA-Seq data analysis

Run Cumulus analysis

Prepare Input Data

Case One: Sample Sheet

Follow the steps below to run **cumulus** on Terra.

1. Create a sample sheet, **count_matrix.csv**, which describes the metadata for each sample count matrix. The sample sheet should at least contain 2 columns — *Sample* and *Location*. *Sample* refers to sample names and *Location* refers to the location of the channel-specific count matrix in either of

- 10x format with v2 chemistry. For example, `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/sample_1/raw_gene_bc_matrices_h5.h5`.
- 10x format with v3 chemistry. For example, `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/sample_1/raw_feature_bc_matrices.h5`.
- Drop-seq format. For example, `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/sample_2/sample_2.umi.dge.txt.gz`.
- Matrix Market format (mtx). If the input is mtx format, location should point to a mtx file with a file suffix of either '.mtx' or '.mtx.gz'. In addition, the associated barcode and gene tsv/txt files should be located in the same folder as the mtx file. For example, if we generate mtx file using BUStools, we set *Location* to `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/mm10/cells_x_genes.mtx`. We expect to see `cells_x_genes.barcodes.txt` and `cellx_x_genes.genes.txt` under folder `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/mm10/`. We support loading mtx files in HCA DCP, 10x Genomics V2/V3, SCUMI, dropEST and BUStools format. Users can also set *Location* to a folder `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/hg19_and_mm10/`. This folder must be generated by Cell Ranger for multi-species samples and we expect one subfolder per species (e.g. 'hg19') and each subfolder should contain mtx file, barcode file, gene name file as generated by Cell Ranger.
- csv format. If it is HCA DCP csv format, we expect the expression file has the name of `expression.csv`. In addition, we expect that `cells.csv` and `genes.csv` files are located under the same folder as the `expression.csv`. For example, `gs://fc-e0000000-0000-0000-0000-000000000000/my_dir/sample_3/`.
- tsv or loom format.

An optional Reference column can be used to select samples generated from a same reference (e.g. `mm10`). If the count matrix is in either DGE, mtx, csv, tsv, or loom format, the value in this column will be used as the reference since the count matrix file does not contain reference name information. The only exception is mtx format. If users do not provide a Reference column, we will use the basename of the folder containing the mtx file as its reference. In addition, the Reference column can be used to aggregate count matrices generated from different genome versions or gene annotations together under a unified reference. For example, if we have one matrix generated from `mm9` and the other one generated from `mm10`, we can write `mm9_10` for these two matrices in their Reference column. Pegasus will change their references to `mm9_10` and use the union of gene symbols from the two matrices as the gene symbols of the aggregated matrix. For HDF5 files (e.g. 10x v2/v3), the reference name contained in the file does not need to match the value in this column. In fact, we use this column to rename references in HDF5 files. For example, if we have two HDF files, one generated from `mm9` and the other generated from `mm10`. We can set these two files' Reference column value to `mm9_10`, which will rename their reference names into `mm9_10` and the aggregated matrix will contain all genes from either `mm9` or `mm10`. This renaming feature does not work if one HDF5 file contain multiple references (e.g. `mm10` and `GRCh38`).

The sample sheet can optionally contain two columns - nUMI and nGene. These two columns define minimum number of UMIs and genes for cell selection for each sample in the sample sheet. nGene column overwrites `minimum_number_of_genes` parameter.

You are free to add any other columns and these columns will be used in selecting channels for futher analysis. In the example below, we have *Source*, which refers to the tissue of origin, *Platform*, which refers to the sequencing platform, *Donor*, which refers to the donor ID, and *Reference*, which refers to the reference genome.

Example:

```
Sample,Source,Platform,Donor,Reference,Location
sample_1,bone_marrow,NextSeq,1,GRCh38,gs://fc-e0000000-0000-0000-0000-
↪000000000000/my_dir/sample_1/raw_gene_bc_matrices_h5.h5
sample_2,bone_marrow,NextSeq,2,GRCh38,gs://fc-e0000000-0000-0000-0000-
↪000000000000/my_dir/sample_2/raw_gene_bc_matrices_h5.h5
sample_3,pbmc,NextSeq,1,GRCh38,gs://fc-e0000000-0000-0000-0000-000000000000/
↪my_dir/sample_3/raw_feature_bc_matrices.h5
sample_4,pbmc,NextSeq,2,GRCh38,gs://fc-e0000000-0000-0000-0000-000000000000/
↪my_dir/sample_4/raw_feature_bc_matrices.h5
```

If you ran **cellranger_workflow** previously, you should already have a template **count_matrix.csv** file that you can modify from **generate_count_config**'s outputs.

1. Upload your sample sheet to the workspace.

   Example:

   ```
   gsutil cp /foo/bar/projects/my_count_matrix.csv gs://fc-e0000000-0000-
   ↪0000-0000-000000000000/
   ```

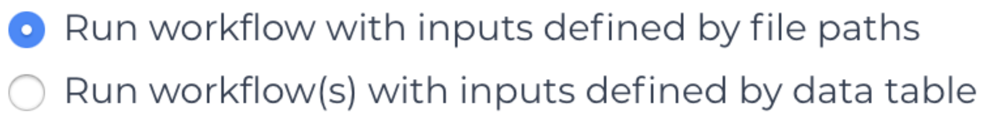
   where `/foo/bar/projects/my_count_matrix.csv` is the path to your sample sheet in local machine, and `gs://fc-e0000000-0000-0000-0000-000000000000/` is the location on Google bucket to hold it.

2. Import *cumulus* workflow to your workspace.

   Import by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Cumulus** for import.

   Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *cumulus* workflow in the drop-down menu.

3. In your workspace, open `cumulus` in `WORKFLOWS` tab. Select `Run workflow with inputs defined by file paths` as below


Run workflow with inputs defined by file paths
Run workflow(s) with inputs defined by data table

   and click the `SAVE` button.

Case Two: Single File

Alternatively, if you only have one single count matrix for analysis, you can go without sample sheets. **Cumulus** currently supports the following formats:

- 10x genomics v2/v3 format (hdf5);
- Drop-seq dge format;
- csv (no HCA DCP format), tsv or loom formats.

Simply upload your data to the Google Bucket of your workspace, and specify its URL in `input_file` field of Cumulus' global inputs (see below). For hdf5 files, there is no need to specify genome names. For other formats, you can specify genome name in `considered_refs` field in cluster inputs; otherwise, default name `''` will be used.

In this case, the **aggregate_matrices** step will be skipped.

Case Three: Multiple samples without aggregation

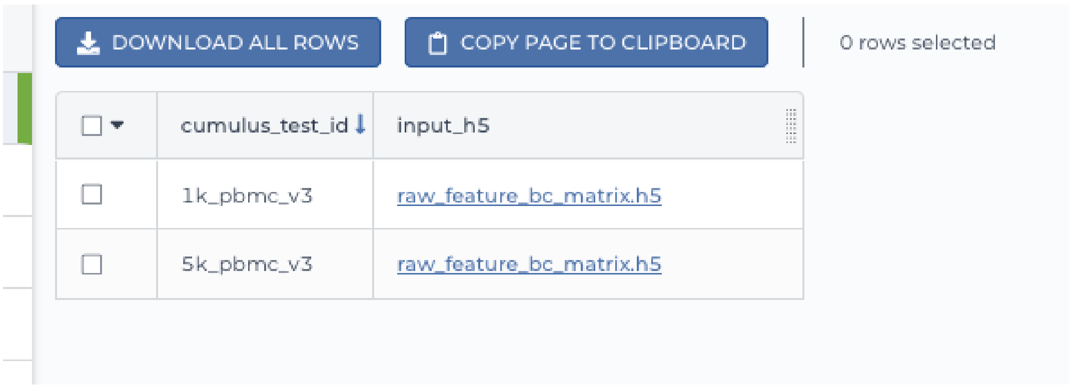
Sometimes, you may want to run Cumulus on multiple samples simultaneously. This is different from Case one, because samples are analyzed separately without aggregation.

1. To do it, you need to first create a data table on Terra. An example TSV file is the following:

```
entity:cumulus_test_id  input_h5
5k_pbmc_v3  gs://fc-e0000000-0000-0000-0000-000000000000/5k_pbmc_v3/raw_feature_
↪bc_matrix.h5
1k_pbmc_v3  gs://fc-e0000000-0000-0000-0000-000000000000/1k_pbmc_v3/raw_feature_
↪bc_matrix.h5
```

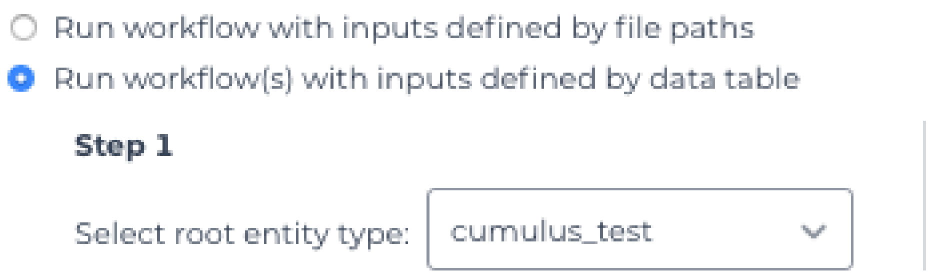
You are free to add more columns, but sample ids and URLs to RNA count matrix files are required. I'll use this example TSV file for the rest of steps in this case.

1. Upload your TSV file to your workspace. Open the `DATA` tab on your workspace. Then click the upload button on left `TABLE` panel, and select the TSV file above. When uploading is done, you'll see a new data table with name "cumulus_test":



cumulus_test_id	input_h5
1k_pbmc_v3	raw_feature_bc_matrix.h5
5k_pbmc_v3	raw_feature_bc_matrix.h5

2. Import *cumulus* workflow to your workspace as in Case one. Then open `cumulus` in `WORKFLOW` tab. Select `Run workflow(s) with inputs defined by data table`, and choose *cumulus_test* from the drop-down menu.



Run workflow with inputs defined by file paths [✓ Run workflow(s) with inputs defined by data table]

Step 1

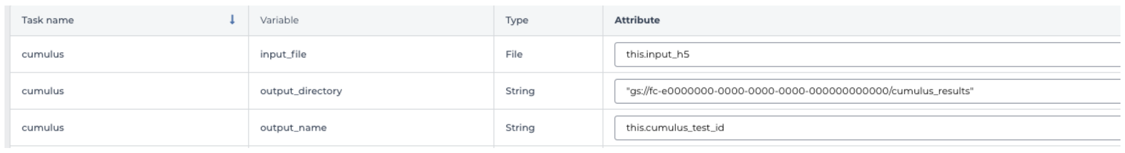
Select root entity type: cumulus_test

3. In the input field, specify:

- `input_file`: Type `this.input_h5`, where `this` refers to the data table selected, and `input_h5` is the column name in this data table for RNA count matrices.

- `output_directory`: Type Google bucket URL for the main output folder. For example, `gs://fc-e0000000-0000-0000-0000-000000000000/cumulus_results`.
- `output_name`: Type `this.cumulus_test_id`, where `cumulus_test_id` is the column name in data table for sample ids.

An example is in the screen shot below:



Task name	Variable	Type	Attribute
cumulus	input_file	File	this.input_h5
cumulus	output_directory	String	"gs://fc-e0000000-0000-0000-0000-000000000000/cumulus_results"
cumulus	output_name	String	this.cumulus_test_id

Then finish setting up other inputs following the description in sections below. When you are done, click `SAVE`, and then `RUN ANALYSIS`.

When all the jobs are done, you'll find output for the 2 samples in subfolders `gs://fc-e0000000-0000-0000-0000-000000000000/cumulus_results/5k_pbmc_v3` and `gs://fc-e0000000-0000-0000-0000-000000000000/cumulus_results/1k_pbmc_v3`, respectively.

Cumulus steps:

Cumulus processes single cell data in the following steps:

1. **aggregate_matrices** (optional). When given a CSV format sample sheet, this step aggregates channel-specific count matrices into one big count matrix. Users can specify which channels they want to analyze and which sample attributes they want to import to the count matrix in this step. Otherwise, if a single count matrix file is given, skip this step.
2. **cluster**. This is the main analysis step. In this step, **Cumulus** performs low quality cell filtration, highly variable gene selection, batch correction, dimension reduction, diffusion map calculation, graph-based clustering and 2D visualization calculation (e.g. t-SNE/UMAP/FLE).
3. **de_analysis**. This step is optional. In this step, **Cumulus** can calculate potential markers for each cluster by performing a variety of differential expression (DE) analysis. The available DE tests include Welch's t test, Fisher's exact test, and Mann-Whitney U test. **Cumulus** can also calculate the area under ROC (AUROC) curve values for putative markers. If `find_markers_lightgbm` is on, **Cumulus** will try to identify cluster-specific markers by training a LightGBM classifier. If the samples are human or mouse immune cells, **Cumulus** can also optionally annotate putative cell types for each cluster based on known markers.
4. **plot**. This step is optional. In this step, **Cumulus** can generate 6 types of figures based on the **cluster** step results:
    - **composition** plots which are bar plots showing the cell compositions (from different conditions) for each cluster. This type of plots is useful to fast assess library quality and batch effects.
    - **umap** and **net_umap**: UMAP like plots based on different algorithms, respectively. Users can specify cell attributes (e.g. cluster labels, conditions) for coloring side-by-side.
    - **tsne**: FIt-SNE plots. Users can specify cell attributes (e.g. cluster labels, conditions) for coloring side-by-side.
    - **fle** and **net_fle**: FLE (Force-directed Layout Embedding) like plots based on different algorithms, respectively. Users can specify cell attributes (e.g. cluster labels, conditions) for coloring side-by-side.
    - If input is CITE-Seq data, there will be **citeseq_umap** plots which are UMAP plots based on epitope expression.

5. **cirro_output**. This step is optional. Generate Cirrocumulus inputs for visualization using Cirrocumulus .
6. **scp_output**. This step is optional. Generate analysis result in Single Cell Portal (SCP) compatible format.

In the following sections, we will first introduce global inputs and then introduce the WDL inputs and outputs for each step separately. But please note that you need to set inputs from all steps simultaneously in the Terra WDL.

Note that we will make the required inputs/outputs bold and all other inputs/outputs are optional.

---

global inputs

Name	Description	Example	Default
input_file	Input CSV sample sheet describing metadata of each 10x channel, or a single input count matrix file	"gs://fc-e0000000-0000-0000-0000-000000000000/my_count_matrix.csv"	
output_directory	Google bucket URL of the output directory.	"gs://fc-e0000000-0000-0000-0000-000000000000/my_results_dir"	
output_name	This is the name of subdirectory for the current sample; and all output files within the subdirectory will have this string as the common filename prefix.	"my_sample"	
default_reference	If sample count matrix is in either DGE, mtx, csv, tsv or loom format and there is no Reference column in the csv_file, use default_reference as the reference string.	"GRCh38"	
pegasus_version	Pegasus version to use for analysis. Versions available: `1.4.3`, `1.4.2`, `1.4.0`, `1.3.0`.	"1.4.3"	"1.4.3"
docker_registry	Docker registry to use. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones to consider for execution.	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of CPUs per Cumulus job	32	64
memory	Memory size string	"200G"	"200G"
disk_space	Total disk space in GB	100	100
backend	Cloud infrastructure backend to use. Available options: • "gcp" for Google Cloud; • "aws" for Amazon AWS; • "local" for local machine.	"gcp"	"gcp"
preemptible	Number of preemptible tries. This works only when *backend* is `gcp`.	2	2
awsMaxRetries	Number of maximum retries when running on AWS. This works only when *backend* is `aws`.	5	5

aggregate_matrices

aggregate_matrices inputs

Name	Description	Example	Default
restrictions	Select channels that satisfy all restrictions. Each restriction takes the format of name:value,…,value. Multiple restrictions are separated by ';'	"Source:bone_marrow;Platform:NextSeq"	
attributes	Specify a comma-separated list of outputted attributes. These attributes should be column names in the count_matrix.csv file	"Source,Platform,Donor"	
select_only_singlets	If we have demultiplexed data, turning on this option will make cumulus only include barcodes that are predicted as singlets.	true	false
remap_singlets	For demultiplexed data, user can remap singlet names using assignment in String in this input. This string assignment takes the format "new_name_i:old_name_1,old_name_2;new_name_ii:old_name_3;…". For example, if we hashed 5 libraries from 3 samples: sample1_lib1, sample1_lib2; sample2_lib1, sample2_lib2; sample3, we can remap them to 3 samples using this string: `"sample1:sample1_lib1,sample1_lib2;sample2:sample2_lib1,sample2_lib2"`. In this way, the new singlet names will be in metadata field with key `assignment`, while the old names are kept in metadata with key `assignment.orig`. **Notice:** This input is enabled only when *select_only_singlets* input is `true`.	"Group1:CB1,CB2;Group2:CB3,CB4,CB5"	
subset_singlets	For demultiplexed data, user can use this input to choose a subset of singlets based on their names. This string takes the format "name1,name2,…". Note that if *remap_singlets* input is specified, subsetting happens after remapping, i.e. you should use the new singlet names for choosing subset. **Notice:** This input is enabled only when *select_only_singlets* input is `true`.	"Group2,CB6,CB7"	
minimum_number_of_genes	Only keep barcodes with at least this number of expressed genes	100	100
is_dropseq	If inputs are DropSeq data.	false	false

aggregate_matrices output

Name	Type	Description
output_aggr_zarr	File	Aggregated count matrix in Zarr format

---

cluster

cluster inputs

Name	Description	Example	Default
focus	Focus analysis on Unimodal data with <keys>. <keys> is a comma-separated list of keys. If None, the `self._selected` will be the focused one. Focus key consists of two parts: reference genome name, and data type, connected with a hyphen marker "–". Reference genome name depends on the reference you used when running Cellranger workflow. See details in reference list.	"GRCh38-rna"	
append	Append Unimodal data <key> to any <keys> in *focus*. Similarly as focus keys, append key also consists of two parts: reference genome name, and data type, connected with a hyphen marker "–". See reference list for details.	"SARSCoV2-rna"	
channel	Specify the cell barcode attribute to represent different samples.	"Donor"	
black_list	Cell barcode attributes in black list will be poped out. Format is "attr1,attr2,. . . ,attrn".	"attr1,attr2,attr3""	
min_genes_before_filtration	If raw data matrix is input, empty barcodes will dominate pre-filtration statistics. To avoid this, for raw data matrix, only consider barcodes with at least <min_genes_before_filtration> genes for pre-filtration condition.	100	100
select_only_singlets	If we have demultiplexed data, turning on this option will make cumulus only include barcodes that are predicted as singlets	false	false

Continued on next page

Table 3 – continued from previous page

Name	Description	Example	Default
remap_singlets	For demultiplexed data, user can remap singlet names using assignment in String in this input. This string assignment takes the format "new_name_i:old_name_1,old_name_2;new_name_ii:old_name_3;…". For example, if we hashed 5 libraries from 3 samples: sample1_lib1, sample1_lib2; sample2_lib1, sample2_lib2; sample3, we can remap them to 3 samples using this string: `"sample1:sample1_lib1,sample1_lib2;sample2:sample2_lib1,sample2_lib2"`. In this way, the new singlet names will be in metadata field with key `assignment`, while the old names are kept in metadata with key `assignment.orig`. **Notice:** This input is enabled only when *select_only_singlets* input is `true`.	"Group1:CB1,CB2;Group2:CB3,CB4,CB5"	
subset_singlets	For demultiplexed data, user can use this input to choose a subset of singlets based on their names. This string takes the format "name1,name2,…". Note that if *remap_singlets* is specified, subsetting happens after remapping, i.e. you should use the new singlet names for choosing subset. **Notice:** This input is enabled only when *select_only_singlets* input is `true`.	"Group2,CB6,CB7"	
output_filtration_results	If write cell and gene filtration results to a spreadsheet	true	true
plot_filtration_results	If plot filtration results as PDF files	true	true
plot_filtration_figsize	Figure size for filtration plots. <figsize> is a comma-separated list of two numbers, the width and height of the figure (e.g. 6,4)	6,4	
output_h5ad	Generate Seurat-compatible h5ad file. Must set to `true` if performing DE analysis, cell type annotation, or plotting.	true	true
output_loom	If generate loom-formatted file	false	false
min_genes	Only keep cells with at least <min_genes> of genes	500	500
max_genes	Only keep cells with less than <max_genes> of genes	6000	6000
min_umis	Only keep cells with at least <min_umis> of UMIs. By default, don't filter cells due to UMI lower bound.	100	
max_umis	Only keep cells with less than <max_umis> of UMIs. By default, don't filter cells due to UMI upper bound.	600000	

Continued on next page

Table 3 – continued from previous page

Name	Description	Example	Default

mito_prefix	Prefix of mitochondrial gene names. This is to identify mitochondrial genes.	"mt-"	"MT-" for *GRCh38* reference genome data;
			"mt-" for *mm10* reference genome data;
			for other reference genome data, must specify this prefix manually.

percent_mito	Only keep cells with mitochondrial ratio less than <percent_mito>% of total counts	50	20.0
gene_percent_cells	Only use genes that are expressed in at <gene_percent_cells>% of cells to select variable genes	50	0.05
counts_per_cell_after	Total counts per cell after normalization, before transforming the count matrix into Log space.	1e5	1e5
select_hvf_flavor	Highly variable feature selection method. Options: • "pegasus": New selection method proposed in Pegasus, the analysis module of Cumulus workflow. • "Seurat": Conventional selection method used by Seurat and SCANPY.	"pegasus"	"pegasus"
select_hvf_ngenes	Select top <select_hvf_ngenes> highly variable features. If <select_hvf_flavor> is "Seurat" and <select_hvf_ngenes> is "None", select HVGs with z-score cutoff at 0.5.	2000	2000
no_select_hvf	Do not select highly variable features.	false	false
plot_hvf	Plot highly variable feature selection. Will not work if `no_select_hvf` is `true`.	false	false
correct_batch_effect	If correct batch effects	false	false
correction_method	Batch correction method. Options: • "harmony": Harmony algorithm (Korsunsky et al. Nature Methods 2019). • "L/S": Location/Scale adjustment algorithm (Li and Wong. The analysis of Gene Expression Data, 2003). • "scanorama": Scanorama algorithm (Hie et al. Nature Biotechnology 2019).	"harmony"	"harmony"

Continued on next page

Table 3 – continued from previous page

Name	Description	Example	Default
batch_group_by	Batch correction assumes the differences in gene expression between channels are due to batch effects. However, in many cases, we know that channels can be partitioned into several groups and each group is biologically different from others. In this case, we will only perform batch correction for channels within each group. This option defines the groups. If <expression> is None, we assume all channels are from one group. Otherwise, groups are defined according to <expression>. <expression> takes the form of either 'attr', or 'attr1+attr2+…+attrn', or 'attr=value11,…,value1n_1;value21,…,value2n_2;…;valuem1,…,valuemn_m'. In the first form, 'attr' should be an existing sample attribute, and groups are defined by 'attr'. In the second form, 'attr1',…,'attrn' are n existing sample attributes and groups are defined by the Cartesian product of these n attributes. In the last form, there will be m + 1 groups. A cell belongs to group i (i > 0) if and only if its sample attribute 'attr' has a value among valuei1,…,valuein_i. A cell belongs to group 0 if it does not belong to any other groups	"Donor"	None
random_state	Random number generator seed	0	0
calc_signature_scores	Gene set for calculating signature scores. It can be either of the following forms: • String chosen from: `cell_cycle_human`, `cell_cycle_mouse`, `gender_human`, `gender_mouse`, `mitochondrial_genes_human`, `mitochondrial_genes_mouse`, `robosomal_genes_human`, `robosomal_genes_mouse`, `apoptosis_human`, and `apoptosis_mouse`. • Google bucket URL of a GMT format file. For example: `gs://fc-e0000000-0000-0000-0000-000000000000/cell_cycle_sig.gmt`.	"cell_cycle_human"	
nPC	Number of principal components	50	50
knn_K	Number of nearest neighbors used for constructing affinity matrix.	50	100

Continued on next page

Table 3 – continued from previous page

Name	Description	Example	Default
knn_full_speed	For the sake of reproducibility, we only run one thread for building kNN indices. Turn on this option will allow multiple threads to be used for index building. However, it will also reduce reproducibility due to the racing between multiple threads.	false	false
run_diffmap	Whether to calculate diffusion map or not. It will be automatically set to `true` when input **run_fle** or **run_net_fle** is set.	false	false
diffmap_ndc	Number of diffusion components	100	100
diffmap_maxt	Maximum time stamp in diffusion map computation to search for the knee point.	5000	5000
run_louvain	Run Louvain clustering algorithm	true	true
louvain_resolution	Resolution parameter for the Louvain clustering algorithm	1.3	1.3
louvain_class_label	Louvain cluster label name in analysis result.	"louvain_labels"	"louvain_labels"
run_leiden	Run Leiden clustering algorithm.	false	false
leiden_resolution	Resolution parameter for the Leiden clustering algorithm.	1.3	1.3
leiden_niter	Number of iterations of running the Leiden algorithm. If negative, run Leiden iteratively until no improvement.	2	-1
leiden_class_label	Leiden cluster label name in analysis result.	"leiden_labels"	"leiden_labels"
run_spectral_louvain	Run Spectral Louvain clustering algorithm	false	false
spectral_louvain_basis	Basis used for KMeans clustering. Use diffusion map by default. If diffusion map is not calculated, use PCA coordinates. Users can also specify "pca" to directly use PCA coordinates.	"diffmap"	"diffmap"
spectral_louvain_resolution	Resolution parameter for louvain.	1.3	1.3
spectral_louvain_class_label	Spectral louvain label name in analysis result.	"spectral_louvain_labels"	"spectral_louvain_labels"
run_spectral_leiden	Run Spectral Leiden clustering algorithm.	false	false
spectral_leiden_basis	Basis used for KMeans clustering. Use diffusion map by default. If diffusion map is not calculated, use PCA coordinates. Users can also specify "pca" to directly use PCA coordinates.	"diffmap"	"diffmap"
spectral_leiden_resolution	Resolution parameter for leiden.	1.3	1.3
spectral_leiden_class_label	Spectral leiden label name in analysis result.	"spectral_leiden_labels"	"spectral_leiden_labels"
run_tsne	Run FIt-SNE for visualization.	false	false
tsne_perplexity	t-SNE's perplexity parameter.	30	30
tsne_initialization	Initialization method for FIt-SNE. It can be either: 'random' refers to random initialization; 'pca' refers to PCA initialization as described in [Kobak et al. 2019].	"pca"	"pca"
run_umap	Run UMAP for visualization	true	true
umap_K	K neighbors for UMAP.	15	15
umap_min_dist	UMAP parameter.	0.5	0.5
umap_spread	UMAP parameter.	1.0	1.0
run_fle	Run force-directed layout embedding (FLE) for visualization	false	false
fle_K	Number of neighbors for building graph for FLE	50	50
fle_target_change_per_node	Target change per node to stop FLE.	2.0	2.0
fle_target_steps	Maximum number of iterations before stopping the algoritm	5000	5000

Continued on next page

Table 3 – continued from previous page

Name	Description	Example	Default
net_down_sample_fraction	Down sampling fraction for net-related visualization	0.1	0.1
run_net_umap	Run Net UMAP for visualization	false	false
net_umap_out_basis	Basis name for Net UMAP coordinates in analysis result	"net_umap"	"net_umap"
run_net_fle	Run Net FLE for visualization	false	false
net_fle_out_basis	Basis name for Net FLE coordinates in analysis result.	"net_fle"	"net_fle"
infer_doublets	Infer doublets using the Pegasus method. When finished, Scrublet-like doublet scores are in cell attribute `doublet_score`, and "doublet/singlet" assignment on cells are stored in cell attribute `demux_type`.	false	false
expected_doublet_rate	The expected doublet rate per sample. If not specified, calculate the expected rate based on number of cells from the 10x multiplet rate table.	0.05	
doublet_cluster_attribute	Specify which cluster attribute (e.g. "louvain_labels") should be used for doublet inference. Then doublet clusters will be marked with the following criteria: passing the Fisher's exact test and having >= 50% of cells identified as doublets. If not specified, the first computed cluster attribute in the list of "leiden", "louvain", "spectral_ledein" and "spectral_louvain" will be used.	"louvain_labels"	
citeseq	Perform CITE-Seq data analysis. Set to `true` if input data contain both RNA and CITE-Seq modalities. This will set *focus* to be the RNA modality and *append* to be the CITE-Seq modality. In addition, `"ADT-"` will be added in front of each antibody name to avoid name conflict with genes in the RNA modality.	false	false
citeseq_umap	For high quality cells kept in the RNA modality, calculate a distinct UMAP embedding based on their antibody expression.	false	false
citeseq_umap_exclude	A comma-separated list of antibodies to be excluded from the CITE-Seq UMAP calculation (e.g. Mouse-IgG1,Mouse-IgG2a).	"Mouse-IgG1,Mouse-IgG2a"	

cluster outputs

Name	Type	Description
output_zarr	File	Output file in zarr format (output_name.zarr.zip). To load this file in Python, you need to first install PegasusIO on your local machine. Then use `import pegasusio as io; data = io.read_input('output_name.zarr.zip')` in Python environment. `data` is a *MultimodalData* object, and points to its default *UnimodalData* element. You can set its default *UnimodalData* to others by `data.set_data(focus_key)` where `focus_key` is the key string to the wanted *UnimodalData* element. For its default *UnimodalData* element, the log-normalized expression matrix is stored in `data.X` as a Scipy CSR-format sparse matrix, with cell-by-gene shape. Alternatively, to get the raw count matrix, first run `data.select_matrix('raw.X')`, then `data.X` will be switched to point to the raw matrix. The `obs` field contains cell related attributes, including clustering results. For example, `data.obs_names` records cell barcodes; `data.obs['Channel']` records the channel each cell comes from; `data.obs['n_genes']`, `data.obs['n_counts']`, and `data.obs['percent_mito']` record the number of expressed genes, total UMI count, and mitochondrial rate for each cell respectively; `data.obs['louvain_labels']`, `data.obs['leiden_labels']`, `data.obs['spectral_louvain_labels']`, and `data.obs['spectral_leiden_labels']` record each cell's cluster labels using different clustering algorithms; The `var` field contains gene related attributes. For example, `data.var_names` records gene symbols, `data.var['gene_ids']` records Ensembl gene IDs, and `data.var['highly_variable_features']` records selected variable genes. The `obsm` field records embedding coordinates. For example, `data.obsm['X_pca']` records PCA coordinates, `data.obsm['X_tsne']` records t-SNE coordinates, `data.obsm['X_umap']` records UMAP coordinates, `data.obsm['X_diffmap']` records diffusion map coordinates, and `data.obsm['X_fle']` records the force-directed layout coordinates. The `uns` field stores other related information, such as reference genome (`data.uns['genome']`), kNN on PCA coordinates (`data.uns['pca_knn_indices']` and `data.uns['pca_knn_distances']`), etc.
output_log	File	This is a copy of the logging module output, containing important intermediate messages
output_h5ad	Array[File]	List of output file(s) in Seurat-compatible h5ad format (output_name.focus_key.h5ad), in which each file is associated with a focus of the input data. To load this file in Python, first install PegasusIO on your local machine. Then use `import pegasusio as io; data = io.read_input('output_name.focus_key.h5ad')` in Python

de_analysis

de_analysis inputs

Name	Description	Example	Default
perform_de_analysis	Whether perform differential expression (DE) analysis. If performing, by default calculate AUROC scores and Mann-Whitney U test.	true	true
cluster_labels	Specify the cluster label used for DE analysis	"louvain_labels"	"louvain_labels"
alpha	Control false discovery rate at <alpha>	0.05	0.05
fisher	Calculate Fisher's exact test	false	false
t_test	Calculate Welch's t-test.	false	false
find_markers_lightgbm	If also detect markers using LightGBM	false	false
remove_ribo	Remove ribosomal genes with either RPL or RPS as prefixes. Currently only works for human data	false	false
min_gain	Only report genes with a feature importance score (in gain) of at least <gain>	1.0	1.0
annotate_cluster	If also annotate cell types for clusters based on DE results	false	false
annotate_de_test	Differential Expression test to use for inference on cell types. Options: `mwu`, `t`, or `fisher`	"mwu"	"mwu"
organism	Organism, could either of the follow: • Preset markers: `human_immune`, `mouse_immune`, `human_brain`, `mouse_brain`, `human_lung`, or a combination of them as a string separated by comma. • User-defined marker file: A Google bucket link to a user-specified JSON file describing the markers. For example: `gs://fc-e0000000/my_markers.json`.	"mouse_immune,mouse_brain"	"human_immune"
minimum_report_score	Minimum cell type score to report a potential cell type	0.5	0.5

de_analysis outputs

Name	Type	Description
output_de_h5ad	Array[File]	List of h5ad-formatted results with DE results updated (output_name.focus_key.h5ad), in which each file is associated with a focus of the input data. To load this file in Python, you need to first install PegasusIO on your local machine. Then type `import pegasusio as io; data = io.read_input('output_name.focus_key.h5ad')` in Python environment. After loading, `data` has the similar structure as *UnimodalData* object in Description of **output_zarr** in cluster outputs section. Besides, there is one additional field `varm` which records DE analysis results in `data.varm['de_res']`. You can use Pandas DataFrame to convert it into a reader-friendly structure: `import pandas as pd; df = pd.DataFrame(data.varm['de_res'], index=data.var_names)`. Then in the resulting data frame, genes are rows, and those DE test statistics are columns. DE analysis in cumulus is performed on each cluster against cells in all the other clusters. For instance, in the data frame, column `1:log2Mean` refers to the mean expression of genes in log-scale for cells in Cluster 1. The number before colon refers to the cluster label to which this statistic belongs.
output_de_xlsx	Array[File]	List of spreadsheets reporting DE results (output_name.focus_key.de.xlsx), in which each file is associated with a focus of the input data. Each cluster has two tabs: one for up-regulated genes for this cluster, one for down-regulated ones. In each tab, genes are ranked by AUROC scores. Genes which are not significant in terms of q-values in any of the DE test are not included (at false discovery rate specified in **alpha** field of de_analysis inputs).
output_markers_xlsx	Array[File]	List of Excel spreadsheets containing detected markers (output_name.focus_key.markers.xlsx), in which each file is associated with a focus of the input data. Each cluster has one tab in the spreadsheet and each tab has three columns, listing markers that are strongly up-regulated, weakly up-regulated and down-regulated.
output_anno_file	Array[File]	List of cluster-based cell type annotation files (output_name.focus_key.anno.txt), in which each file is associated with a focus of the input data.

How cell type annotation works

In this subsection, we will describe the format of input JSON cell type marker file, the *ad hoc* cell type inference algorithm, and the format of the output putative cell type file.

JSON file

The top level of the JSON file is an object with two name/value pairs:

- **title**: A string to describe what this JSON file is for (e.g. "Mouse brain cell markers").
- **cell_types**: List of all cell types this JSON file defines. In this list, each cell type is described using a separate object with 2 to 3 name/value pairs:
  - **name**: Cell type name (e.g. "GABAergic neuron").
  - **markers**: List of gene-marker describing objects, each of which has 2 name/value pairs:
    - **genes**: List of positive and negative gene markers (e.g. `["Rbfox3+", "Flt1-"]`).
    - **weight**: A real number between `0.0` and `1.0` to describe how much we trust the markers in **genes**.

    All markers in **genes** share the weight evenly. For instance, if we have 4 markers and the weight is 0.1, each marker has a weight of `0.1 / 4 = 0.025`.

    The weights from all gene-marker describing objects of the same cell type should sum up to 1.0.
  - **subtypes**: Description on cell subtypes for the cell type. It has the same structure as the top level JSON object.

See below for an example JSON snippet:

```
{
  "title" : "Mouse brain cell markers",
    "cell_types" : [
      {
        "name" : "Glutamatergic neuron",
        "markers" : [
          {
            "genes" : ["Rbfox3+", "Reln+", "Slc17a6+", "Slc17a7+"],
            "weight" : 1.0
          }
        ],
        "subtypes" : {
          "title" : "Glutamatergic neuron subtype markers",
            "cell_types" : [
              {
                "name" : "Glutamatergic layer 4",
                "markers" : [
                  {
                    "genes" : ["Rorb+", "Paqr8+"],
                    "weight" : 1.0
                  }
                ]
              }
            ]
        }
      }
    ]
}
```

Inference Algorithm

We have already calculated the up-regulated and down-regulated genes for each cluster in the differential expression analysis step.

First, load gene markers for each cell type from the JSON file specified, and exclude marker genes, along with their associated weights, that are not expressed in the data.

Then scan each cluster to determine its putative cell types. For each cluster and putative cell type, we calculate a score between `0` and `1`, which describes how likely cells from the cluster are of this cell type. The higher the score is, the more likely cells are from the cell type.

To calculate the score, each marker is initialized with a maximum impact value (which is `2`). Then do case analysis as follows:

- For a positive marker:
  - If it is not up-regulated, its impact value is set to `0`.
  - Otherwise, if it is up-regulated:
    - If it additionally has a fold change in percentage of cells expressing this marker (within cluster vs. out of cluster) no less than `1.5`, it has an impact value of `2` and is recorded as a **strong supporting marker**.
    - If its fold change (`fc`) is less than `1.5`, this marker has an impact value of `1 + (fc - 1) / 0.5` and is recorded as a **weak supporting marker**.
- For a negative marker:
  - If it is up-regulated, its impact value is set to `0`.
  - If it is neither up-regulated nor down-regulated, its impact value is set to `1`.
  - Otherwise, if it is down-regulated:
    - If it additionally has `1 / fc` (where `fc` is its fold change) no less than `1.5`, it has an impact value of `2` and is recorded as a **strong supporting marker**.
    - If `1 / fc` is less than `1.5`, it has an impact value of `1 + (1 / fc - 1) / 0.5` and is recorded as a **weak supporting marker**.

The score is calculated as the weighted sum of impact values weighted over the sum of weights multiplied by 2 from all expressed markers. If the score is larger than 0.5 and the cell type has cell subtypes, each cell subtype will also be evaluated.

Output annotation file

For each cluster, putative cell types with scores larger than `minimum_report_score` will be reported in descending order with respect to their scores. The report of each putative cell type contains the following fields:

- **name**: Cell type name.
- **score**: Score of cell type.
- **average marker percentage**: Average percentage of cells expressing marker within the cluster between all positive supporting markers.
- **strong support**: List of strong supporting markers. Each marker is represented by a tuple of its name and percentage of cells expressing it within the cluster.
- **weak support**: List of week supporting markers. It has the same structure as **strong support**.

plot

The h5ad file contains a default cell attribute `Channel`, which records which channel each that single cell comes from. If the input is a CSV format sample sheet, `Channel` attribute matches the `Sample` column in the sample sheet. Otherwise, it's specified in `channel` field of the cluster inputs.

Other cell attributes used in plot must be added via `attributes` field in the `aggregate_matrices` inputs.

plot inputs

Name	Description	Example	Default
plot_composition	Takes the format of "label:attr,label:attr,. . . ,label:attr". If non-empty, generate composition plot for each "label:attr" pair. "label" refers to cluster labels and "attr" refers to sample conditions	"louvain_labels:Donor"	None
plot_tsne	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored FIt-SNEs side by side	"louvain_labels,Donor"	None
plot_umap	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored UMAP side by side	"louvain_labels,Donor"	None
plot_fle	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored FLE (force-directed layout embedding) side by side	"louvain_labels,Donor"	None
plot_net_umap	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored UMAP side by side based on net UMAP result.	"leiden_labels,Donor"	None
plot_net_fle	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored FLE (force-directed layout embedding) side by side based on net FLE result.	"leiden_labels,Donor"	None
plot_citeseq_umap	Takes the format of "attr,attr,. . . ,attr". If non-empty, plot attr colored UMAP side by side based on CITE-Seq UMAP result.	"louvain_labels,Donor"	None

plot outputs

Name	Type	Description
output_pdfs	Array[File]	Outputted pdf files
output_htmls	Array[File]	Outputted html files

Generate input files for Cirrocumulus

Generate Cirrocumulus inputs for visualization using Cirrocumulus .

cirro_output inputs

Name	Description	Example	Default
generate_cirro_inputs	Whether to generate input files for Cirrocumulus	false	false

cirro_output outputs

Name	Type	Description
output_cirro_path	Google Bucket URL	Path to Cirrocumulus inputs

---

Generate SCP-compatible output files

Generate analysis result in Single Cell Portal (SCP) compatible format.

scp_output inputs

Name	Description	Example	Default
generate_scp_outputs	Whether to generate SCP format output or not.	false	false
output_dense	Output dense expression matrix, instead of the default sparse matrix format.	false	false

scp_output outputs

Name	Type	Description
output_scp_files	Array[File]	Outputted SCP format files.

Run CITE-Seq analysis

Users now can use *cumulus/cumulus* workflow solely to run CITE-Seq analysis.

1. Prepare a sample sheet in the following format:

```
Sample,Location,Modality
sample_1,gs://your-bucket/rna_raw_counts.h5,rna
sample_1,gs://your-bucket/citeseq_cell_barcodes.csv,citeseq
```

Each row stands for one modality:

- **Sample:** Sample name, which *must* be the same in the two rows to let Cumulus aggregate RNA and CITE-Seq matrices.
- **Location:** Google bucket URL of the corresponding count matrix file.
- **Modality:** Modality type. `rna` for RNA count matrix; `citeseq` for CITE-Seq antibody count matrix.

2. Run *cumulus/cumulus* workflow using this sample sheet as the input file, and specify the following input fields:
   - **citeseq**: Set this to `true` to enable CITE-Seq analysis.
   - **citeseq_umap**: Set this to `true` to calculate the CITE-Seq UMAP embedding on cells.
   - **citeseq_umap_exclude**: A list of CITE-Seq antibodies to be excluded from UMAP calculation. This list should be written in a string format with each antobidy name separated by comma.
   - **plot_citeseq_umap**: A list of cell barcode attributes to be plotted based on CITE-Seq UMAP embedding. This list should be written in a string format with each attribute separated by comma.

Load Cumulus results into Pegasus

Pegasus is a Python package for large-scale single-cell/single-nucleus data analysis, and it uses PegasusIO for read/write. To load Cumulus results into Pegasus, we provide instructions based on file format:

- **zarr**: Annotated Zarr file in zip format. This is the standard output format of Cumulus. You can load it by:

```
import pegasusio as io
data = io.read_input("output_name.zarr.zip")
```

- **h5ad**: When setting **"output_h5ad"** field in *Cumulus cluster* to *true*, a list of annotated H5AD file(s) will be generated besides Zarr result. If the input data have multiple foci, Cumulus will generate one H5AD file per focus. You can load it by:

```
import pegasusio as io
adata = io.read_input("output_name.focus_key.h5ad")
```

Sometimes you may also want to specify how the result is loaded into memory. In this case, `read_input` has argument `mode`. Please see its documentation for details.

- **loom**: When setting **"output_loom"** field in *Cumulus cluster* to **true**, a list of loom format file(s) will be generated besides Zarr result. Similarly as H5AD output, Cumulus generates multiple loom files if the input data have more than one foci. To load loom file, you can optionally set its genome name in the following way as this information is not contained by loom file:

```
import pegasusio as io
data = pg.read_input("output_name.focus_key.loom", genome = "GRCh38")
```

After loading, Pegasus manipulate the data matrix in PegasusIO *MultimodalData* structure.

Load Cumulus results into Seurat

Seurat is a single-cell data analysis package written in R.

Load H5AD File into Seurat

First, you need to set **"output_h5ad"** field to **true** in cumulus cluster inputs to generate Seurat-compatible output files `output_name.focus_key.h5ad`, in addition to the standard result `output_name.zarr.zip`. If the input data have multiple foci, Cumulus will generate one H5AD file per focus.

Notice that Python, and Python package anndata with version at least `0.6.22.post1`, and R package reticulate are required to load the result into Seurat.

Execute the R code below to load the h5ad result into Seurat (working with both Seurat v2 and v3):

```
source("https://raw.githubusercontent.com/lilab-bcb/cumulus/master/workflows/cumulus/
↪h5ad2seurat.R")
ad <- import("anndata", convert = FALSE)
test_ad <- ad$read_h5ad("output_name.focus_key.h5ad")
result <- convert_h5ad_to_seurat(test_ad)
```

The resulting Seurat object `result` has three data slots:

- **raw.data** records filtered raw count matrix.
- **data** records filtered and log-normalized expression matrix.
- **scale.data** records variable-gene-selected, standardized expression matrix that are ready to perform PCA.

Load loom File into Seurat

First, you need to set **"output_loom"** field to **true** in cumulus cluster inputs to generate a loom format output file, say `output_name.focus_key.loom`, in addition to the standard result `output_name.zarr.zip`. If the input data have multiple foci, Cumulus will generate one loom file per focus.

You also need to install *loomR* package in your R environment:

```
install.package("devtools")
devtools::install_github("mojaveazure/loomR", ref = "develop")
```

Execute the R code below to load the loom file result into Seurat (working with Seurat v3 only):

```
source("https://raw.githubusercontent.com/lilab-bcb/cumulus/master/workflows/cumulus/
↪loom2seurat.R")
result <- convert_loom_to_seurat("output_name.focus_key.loom")
```

In addition, if you want to set an active cluster label field for the resulting Seurat object, do the following:

```
Idents(result) <- result@meta.data$louvain_labels
```

where `louvain_labels` is the key to the Louvain clustering result in Cumulus, which is stored in cell attributes `result@meta.data`.

Load Cumulus results into SCANPY

SCANPY is another Python package for single-cell data analysis. We provide instructions on loading Cumulus output into SCANPY based on file format:

- **h5ad**: Annotated H5AD file. This is the standard output format of Cumulus:

```
import scanpy as sc
adata = sc.read_h5ad("output_name.h5ad")
```

Sometimes you may also want to specify how the result is loaded into memory. In this case, `read_h5ad` has argument `backed`. Please see SCANPY documentation for details.

- **loom**: This format is generated when setting **"output_loom"** field in Cumulus cluster to **true**:

```
import scanpy as sc
adata = sc.read_loom("output_name.loom")
```

Besides, `read_loom` has a boolean `sparse` argument to decide whether to read the data matrix as sparse, with default value `True`. If you want to load it as a dense matrix, simply type:

```
adata = sc.read_loom("output_name.loom", sparse = False)
```

After loading, SCANPY manipulates the data matrix in anndata structure.

Visualize Cumulus results in Python

Ensure you have Pegasus installed.

Download your analysis result data, say `output_name.zarr.zip`, from Google bucket to your local machine.

Follow Pegasus plotting tutorial for visualizing your data in Python.

1.1.10 Run Terra pipelines via command line

You can run Terra pipelines via the command line by installing the Altocumulus package (version `2.0.0` or later is required).

Install Altocumulus

1. Make sure you have `conda` installed. If you haven't installed conda, use the following commands to install it on Linux:

```
wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh .
bash Miniconda3-latest-Linux-x86_64.sh -p /home/foo/miniconda3
mv Miniconda3-latest-Linux-x86_64.sh /home/foo/miniconda3
```

   where `/home/foo/miniconda3` should be replaced by your own folder holding Miniconda3.

Or use the following commdands for MacOS installation:

```
curl -O curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
bash Miniconda3-latest-MacOSX-x86_64.sh -p /Users/foo/miniconda3
mv Miniconda3-latest-MacOSX-x86_64.sh /Users/foo/miniconda3

where ``/Users/foo/miniconda3`` should be replaced by your own folder holding␣
↪Miniconda3.
```

1. Create a conda environment named "alto" and install Altocumulus:

   ```
   conda create -n alto -y pip
   source activate alto
   pip install altocumulus
   ```

When the installation is done, type `alto -h` in terminal to see if you can see the help information.

Set up Google Cloud Account

Install gcloud CLI on your local machine.

Then type the following command in your terminal

```
gcloud auth application-default login
```

and follow the pop-up instructions to set up your Google cloud account.

Run workflows on Terra

alto terra run submits workflows to Terra for execution. Features:

- Uploads local files/directories in your inputs to a Google Cloud bucket updates the file paths to point to the Google Cloud bucket.

  Your sample sheet can point to local file paths. In this case, `alto terra run` will take care of uploading directories smartly (e.g. only upload necessary files in BCL folders) and modifying the sample sheet to point to a Google Cloud bucket.

- Creates or uses an existing workspace.
- Uses the latest version of a method unless the method version is specified.

Options

Required options are in bold.

Name	Description
-m <METHOD> –method <METHOD>	Specify a Terra workflow *<METHOD>* to use. *<METHOD>* is of format *Namespace/Name* (e.g. `cumulus/cellranger_workflow`). Workflow name. The workflow can come from either Dockstore or Broad Methods Repository. If it comes from Dockstore, specify the name as organization:collection:name:version (e.g. broadinstitute:cumulus:cumulus:1.5.0) and the default version would be used if version is omitted. If it comes from Broad Methods Repository, specify the name as namespace/name/version (e.g. cumulus/cumulus/43) and the latest snapshot would be used if version is omitted.
-w <WORKSPACE> –workspace <WORKSPACE>	Specify which Terra workspace *<WORKSPACE>* to use. *<WORKSPACE>* is also of format *Namespace/Name* (e.g. `foo/bar`). The workspace will be created if it does not exist.
-i <WDL_INPUTS> –inputs <WDL_INPUTS>	Specify the WDL input JSON file to use. It can be a local file, a JSON string, or a Google bucket URL directing to a remote JSON file.
–bucket-folder <folder>	Store inputs to <folder> under workspace's google bucket.
-o <updated_json> –upload <updated_json>	Upload files/directories to Google bucket of the workspace, and generate an updated input JSON file (with local paths replaced by Google bucket URLs) to <updated_json> on local machine.
–no-cache	Disable Terra cache calling

Example run on Terra

This example shows how to use `alto terra run` to run cellranger_workflow to extract gene-count matrices from sequencing output.

1. Prepare your sample sheet `example_sample_sheet.csv` as the following:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry
sample_1,GRCh38,/my-local-path/flowcell1,1-2,SI-GA-A8,threeprime
sample_2,GRCh38,/my-local-path/flowcell1,3-4,SI-GA-B8,threeprime
sample_3,mm10,/my-local-path/flowcell1,5-6,SI-GA-C8,fiveprime
```

(continues on next page)

(continued from previous page)

```
sample_4,mm10,/my-local-path/flowcell1,7-8,SI-GA-D8,fiveprime
sample_1,GRCh38,/my-local-path/flowcell2,1-2,SI-GA-A8,threeprime
sample_2,GRCh38,/my-local-path/flowcell2,3-4,SI-GA-B8,threeprime
sample_3,mm10,/my-local-path/flowcell2,5-6,SI-GA-C8,fiveprime
sample_4,mm10,/my-local-path/flowcell2,7-8,SI-GA-D8,fiveprime
```

where `/my-local-path` is the top-level directory of your BCL files on your local machine.

Note that `sample_1`, `sample_2`, `sample_3`, and `sample_4` are sequenced on 2 flowcells.

2. Prepare your JSON input file `inputs.json` for cellranger_workflow:

```
{
    "cellranger_workflow.input_csv_file" : "/my-local-path/sample_sheet.csv",
    "cellranger_workflow.output_directory" : "gs://url/outputs",
    "cellranger_workflow.delete_input_bcl_directory": true
}
```

where `gs://url/outputs` is the folder on Google bucket of your workspace to hold output.

3. Run the following command to kick off your Terra workflow:

```
alto terra run -m cumulus/cellranger_workflow -i inputs.json -w myworkspace_
↪namespace/myworkspace_name -o inputs_updated.json
```

where `myworkspace_namespace/myworkspace_name` should be replaced by your workspace namespace and name.

Upon success, `alto terra run` returns a URL pointing to the submitted Terra job for you to monitor.

If for any reason, your job failed. You could rerun it without uploading files again via the following command:

```
alto terra run -m cumulus/cellranger_workflow -i inputs_updated.json -w myworkspace_
↪namespace/myworkspace_name
```

because `inputs_updated.json` is the updated version of `inputs.json` with all local paths being replaced by their corresponding Google bucket URLs after uploading.

Run workflows on a Cromwell server

alto cromwell run submits WDL jobs to a Cromwell server for execution. Features:

- Uploads local files/directories in your inputs to an appropriate location depending on backend chosen and updates the file paths to point to the bucket information.
- Uses the method parameter to pull in appropriate worflow to import and run.

Options

Required options are in bold.

Name	Description
-s <SERVER> –server <SERVER>	Server hostname or IP address.
-p <PORT> –port <PORT>	Port number for Cromwell service. The default port is 8000.
-m <METHOD_STR> –method <METHOD_STR>	Workflow name from Dockstore, with name specified as organization:collection:name:version (eg. broadinstitute:cumulus:cumulus:1.5.0). The default version would be used if version is omitted.
-i <INPUT> –input <INPUT>	Path to a local JSON file specifying workflow inputs.
-o <updated_json> –upload <INPUT>	Upload files/directories to the workspace cloud bucket and output updated input json (with local path replaced by cloud bucket urls) to <updated_json>.
-b <[s3|gs]://<bucket-name>/<bucket-folder>> –bucket <[s3|gs]://<bucket-name>/<bucket-folder>>	Cloud bucket folder for uploading local input data. Start with ‘s3://’ if an AWS S3 bucket is used, ‘gs://’ for a Google bucket. Must be specified when ‘-o’ option is used.
–no-ssl-verify	Disable SSL verification for web requests. Not recommended for general usage, but can be useful for intra-networks which don’t support SSL verification.

Example import of any Cumulus workflow

This example shows how to use `alto cromwell run` to run demultiplexing workflow on any backend.

1. Prepare your sample sheet `demux_sample_sheet.csv` as the following:

```
OUTNAME,RNA,TagFile,TYPE
sample_1,gs://exp/data_1/raw_feature_bc_matrix.h5,gs://exp/data_1/sample_1_ADT.
↪csv,cell-hashing
```

(continues on next page)

(continued from previous page)

```
sample_2,gs://exp/data_2/raw_feature_bc_matrix.h5,gs://exp/data_3/possorted_
↪genome_bam.bam,genetic-pooling
```

2. Prepare your JSON input file `cumulus_inputs.json` for cellranger_workflow:

```
{
   "demultiplexing.input_sample_sheet" : "demux_sample_sheet.csv",
   "demultiplexing.output_directory" : "gs://url/outputs",
   "demultiplexing.zones" : "us-west1-a us-west1-b us-west1-c",
   "demultiplexing.backend" : "gcp",
   "demultiplexing.genome" : "GRCh38-2020-A"
}
```

   where `gs://url/outputs` is the folder on Google bucket of your workspace to hold output.

3. Run the following command to kick off your run on a chosen backend:

```
alto cromwell run -s 10.10.10.10 -p 3000 -m␣
↪broadinstitute:cumulus:Demultiplexing:master \
                  -i cumulus_inputs.json
```

1.1.11 Examples

Example of Gene expression, Hashing and CITE-Seq Analysis on Cloud

In this example, you'll learn how to perform Gene expression, Hashing and CITE-Seq data analysis on Cloud.

This example covers the cases of both Terra platform and a custom cloud server running Cromwell. When reading through the tutorial, you may check out the corresponding part based on your working situation.

---

0. Prerequisite

0-a. Cromwell server

If you use a Cromwell server on Cloud, on your local machine, you need to install the corresponding Cloud SDK tool if not:

- gcloud CLI if your Cloud bucket is on Google Cloud.
- AWS CLI v2 if your Cloud bucket is on Amazon AWS Cloud.

And then install Altocumulus in your Python environment. This is the tool for data transfer between local machine and cloud bucket, as well as communication with the Cromwell server on cloud.

0-b. Terra Platform

If you use Terra, after registering on Terra and creating a workspace there, you'll need the following information:

- **Terra workspace name**. This is shown on your Terra workspace webpage, with format "<*workspace-namespace*>/<*workspace-name*>". For example, if your Terra workspace has full name `ws-lab/ws-01`, then **ws-lab** is the namespace and **ws-01** is the workspace name winthin that namespace.

- The corresponding **Google Cloud Bucket** of your Terra workspace. You can check it under "*Google Bucket*" title on the right panel of your Terra workspace's *Dashboard* tab. The bucket name associated with your workspace starts with `fc-` followed by a sequence of heximal numbers. For example, `gs://fc-e0000000`, where "*gs://*" is the header of GS URI.

Besides, install gcloud CLI and Altocumulus on your local machine for data uploading. These tools will be used for data transfer between local machine and Cloud bucket.

Alternatively, you can also use Terra web UI for job submission instead of command-line submission. This will be discussed in Section Run Analysis with Terra Web UI below.

---

1. Extract Gene-Count Matrices

This phase is to extract gene-count matrices from sequencing output.

There are two cases: (1) from BCL data, which includes *mkfastq* step to generate FASTQ files and *count* step to generate gene-count matrices; (2) from FASTQ files, which only runs the *count* step.

1-a. Extract Genen-Count Matrices from BCL data

This section covers the case starting from BCL data.

Step 1. Sample Sheet Preparation

First, prepare a feature index file for your dataset. Say its filename is `antibody_index.csv`, which has format "*feature_barcode, feature_name,feature_type*". See an example below:

```
TTCCTGCCATTACTA,HTO_1,hashing
CCGTACCTCATTGTT,HTO_2,hashing
GGTAGATGTCCTCAG,HTO_3,hashing
TGGTGTCATTCTTGA,Ab1,citeseq
CTCATTGTAACTCCT,Ab2,citeseq
GCGCAACTTGATGAT,Ab3,citeseq
... ...
```

where each line contains the barcode and the name of a Hashing/CITE-Seq index: `hashing` indicates a Cell/Nucleus-Hashing index, while `citeseq` indicates a CITE-Seq index.

Next, create a sample sheet `cellranger_sample_sheet.csv` for Cell Ranger processing on your local machine. Below is an example:

```
Sample,Reference,Flowcell,Lane,Index,Chemistry,DataType,FeatureBarcodeFile
sample_control,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A1,fiveprime,rna
sample_gex,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A2,fiveprime,rna
sample_cell_hashing,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-NN-A1,fiveprime,
↪hashing,/path/to/antibody_index.csv
sample_cite_seq,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-NN-A2,fiveprime,citeseq,/
↪path/to/antibody_index.csv
```

where

- `GRCh38-2020-A` is the is the Human GRCh38 (GENCODE v32/Ensembl 98) genome reference prebuilt by Cumulus. See Cumulus single-cell genome reference list for a complete list of genome references.

- /path/to/flowcell/folder should be replaced by the actual local path to the BCL folder of your sequencing data.
- /path/to/antibody_index.csv should be replaced by the actual local path to antibody_index.csv file we just created above.
- rna, hashing and citeseq refer to gene expression data, cell/nucleus-hashing data, and CITE-Seq data, respectively.
- Samples of type rna do not need any feature barcode file for indexing.

For the details on how to prepare this sample sheet, please refer to Step 3 of Cell Ranger sample sheet instruction.

Step 2. Workflow Input Preparation

Now prepare a JSON file for **cellranger_workflow** WDL workflow input on your local machine (say named cellranger_inputs.json):

```
{
        "cellranger_workflow.input_csv_file": "/path/to/cellranger_sample_sheet.csv",
        "cellranger_workflow.output_directory": "gs://my-bucket/cellranger_output"
}
```

where

- /path/to/cellranger_sample_sheet.csv should be replaced by the actual local path to your sample sheet created above.
- gs://my-bucket/cellranger_output is the target folder on Google bucket to store your result when the workflow job is finished, where my-bucket should be replaced by your own Google bucket name.

For details on the all the workflow inputs of *cellranger_workflow*, please refer to Cell Ranger workflow inputs.

Step 3. Job Submission

Now we are ready to submit a job to cloud for computing:

- If you use a Cromwell server on cloud, run the following Altocumulus command:

```
alto cromwell run -s <server-address> -p <port-number> -m␣
↪broadinstitute:cumulus:cellranger -i /path/to/cellranger_inputs.json -o␣
↪cellranger_inputs_updated.json -b gs://my-bucket/data_source
```

where

- -s specifies the server's IP address (or hostname), where <server-address> should be replaced by the actual IP address (or hostname).
- -p specifies the server's port number for Cromwell, where <port-number> should be replaced by the actual port number.
- -m specifies which WDL workflow to use. You should use the Dockstore name of Cumulus cellranger_workflow. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. broadinstitute:cumulus:cellranger:master to use its development version in *master* branch.
- -i specifies the workflow input JSON file.
- -o and -b are used when the input data (which are specified in the workflow input JSON file and sample sheet CSV file) are local and need to be uploaded to Cloud bucket first.

- `-o` specifies the updated workflow input JSON file after uploading the input data, with all the local paths updated to Cloud bucket URIs. This is useful when resubmitting jobs running the same input data, without uploading the same input data again.
- `-b` specifies which folder on Cloud bucket to upload the local input data, where `my-bucket` should be replaced by your own Google bucket name. Feel free to choose the folder name other than `data_source`.

Notice that `-o` and `-b` options can be dropped if all of your input data are already on Cloud bucket.

After submission, you'll get the job's ID for tracking its status:

```
alto cromwell check_status -s <server-address> -p <port-number> --id <your-job-ID>
```

where `<your-job-ID>` should be replaced by the actual Cromwell job ID.

- If you use Terra, run the following Altocumulus command:

  ```
  alto terra run -m broadinstitute:cumulus:cellranger -w ws-lab/ws-01 --bucket-
  ↪folder data_source -i /path/to/cellranger_inputs.json -o cellranger_inputs_
  ↪updated.json
  ```

where

- `-m` specifies which WDL workflow to use. You should use the Dockstore name of Cumulus cellranger_workflow. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. `broadinstitute:cumulus:cellranger:master` to use its development version in *master* branch.
- `-w` specifies the Terra workspace full name to use, where `ws-lab/ws-01` should be replaced by your own Terra workspace full name.
- `--bucket-folder` specifies the folder name on the Google bucket associated with the Terra workspace to store the uploaded data. Feel free to choose folder name other than `data_source`.
- `-i` specifies the workflow input JSON file, where `/path/to/cellranger_inputs.json` should be replaced by the actual local path to `cellranger_inputs.json` file.
- `-o` specifies the updated workflow input JSON file after uploading the input data, with all the local paths updated to Cloud bucket URIs. This is useful when resubmitting jobs running the same input data, without uploading the same input data again.

Notice that `--bucket-folder` and `-o` options can be dropped if all of your input data are already on Cloud bucket.

After submission, you can check the job's status in the *Job History* tab of your Terra workspace page.

When the job is done, you'll get results in `gs://my-bucket/cellranger_output`, which is specified in `cellranger_inputs.json` above. It should contain 4 subfolders, each of which is associated with one sample specified in `cellranger_sample_sheet.csv` above.

For the next phases, you'll need 3 files from the output:

- RNA count matrix of the sample group of interest: `gs://my-bucket/cellranger_output/sample_gex/raw_feature_bc_matrix.h5`;
- Cell-Hashing Antibody count matrix: `gs://my-bucket/cellranger_output/sample_cell_hashing/sample_cell_hashing.csv`;
- CITE-Seq Antibody count matrix: `gs://my-bucket/cellranger_output/sample_cite_seq/sample_cite_seq.csv`.

1-b. Extract Gene-Cound Matrices from FASTQ files

This section covers the case starting from FASTQ files.

Similarly as above, First, prepare a feature index file for your dataset. Say its filename is `antibody_index.csv`, which has format "*feature_barcode, feature_name,feature_type*". See an example below:

```
TTCCTGCCATTACTA,HTO_1,hashing
CCGTACCTCATTGTT,HTO_2,hashing
GGTAGATGTCCTCAG,HTO_3,hashing
TGGTGTCATTCTTGA,Ab1,citeseq
CTCATTGTAACTCCT,Ab2,citeseq
GCGCAACTTGATGAT,Ab3,citeseq
... ...
```

where each line contains the barcode and the name of a Hashing/CITE-Seq index: `hashing` indicates a Cell/Nucleus-Hashing index, while `citeseq` indicates a CITE-Seq index.

Next, create a sample sheet `cellranger_sample_sheet.csv` for Cell Ranger processing on your local machine. Below is an example:

```
Sample,Reference,Flowcell,Chemistry,DataType,FeatureBarcodeFile
sample_1_rna,GRCh38-2020-A,/path/to/fastq/gex,fiveprime,rna
sample_2_rna,GRCh38-2020-A,/path/to/fastq/gex,fiveprime,rna
sample_3_rna,GRCh38-2020-A,/path/to/fastq/gex,fiveprime,rna
sample_1_adt,GRCh38-2020-A,/path/to/fastq/hashing_citeseq,fiveprime,adt,/path/to/
↪antibody_index.csv
sample_2_adt,GRCh38-2020-A,/path/to/fastq/hashing_citeseq,fiveprime,adt,/path/to/
↪antibody_index.csv
sample_3_adt,GRCh38-2020-A,/path/to/fastq/hashing_citeseq,fiveprime,adt,/path/to/
↪antibody_index.csv
```

where

- `GRCh38-2020-A` is the is the Human GRCh38 (GENCODE v32/Ensembl 98) genome reference prebuilt by Cumulus. See Cumulus single-cell genome reference list for a complete list of genome references.
- `/path/to/fastq/gex` should be replaced by the actual local path to the folder containing FASTQ files of RNA samples.
- `/path/to/fastq/hashing_citeseq` should be replaced by the actual local path to the folder containing FASTQ files of Cell/Nucleus-Hashing and CITE-Seq samples.
- `/path/to/antibody_index.csv` should be replaced by the actual local path to `antibody_index.csv` file we just created above.
- `rna` and `adt` refer to gene expression data and antibody data, respectively. In specific, `adt` covers both `citeseq` and `hashing` types, i.e. it includes both Hashing and CITE-Seq data types.
- Samples of type `rna` do not need any feature barcode file for indexing.
- Columns *Lane* and *Index* are not needed if starting from FASTQ files, as *mkfastq* step will be skipped.

For the details on how to prepare this sample sheet, please refer to Step 3 of Cell Ranger sample sheet instruction.

Now prepare a JSON file for **cellranger_workflow** WDL workflow input on your local machine (say named `cellranger_inputs.json`):

```
{
        "cellranger_workflow.input_csv_file": "/path/to/cellranger_sample_sheet.csv",
```

(continues on next page)

(continued from previous page)

```
        "cellranger_workflow.output_directory": "gs://my-bucket/cellranger_output",
        "cellranger_workflow.run_mkfastq": false
}
```

where

- `/path/to/cellranger_sample_sheet.csv` should be replaced by the actual local path to your sample sheet created above.
- `gs://my-bucket/cellranger_output` is the target folder on Google bucket to store your result when the workflow job is finished, where `my-bucket` should be replaced by your own Google bucket name.
- Set *run_mkfastq* to `false` to skip the *mkfastq* step, as we start from FASTQ files.

For details on the all the workflow inputs of *cellranger_workflow*, please refer to Cell Ranger workflow inputs.

Now we are ready to submit a job to cloud for computing. Follow instructions in Section 1-a above.

When finished, you'll get results in `gs://my-bucket/cellranger_output`, which is specified in `cellranger_inputs.json` above. It should contain 6 subfolders, each of which is associated with one sample specified in `cellranger_sample_sheet.csv` above.

In specific, for each `adt` type sample, there are both count matrix of Hashing data and that of CITE-Seq data generated inside its corresponding subfolder, with filename suffix `.hashing.csv` and `.citeseq.csv`, respectively.

---

2. Demultiplex Cell-Hashing Data using DemuxEM

Run Workflow on Cloud

Next, we need to demultiplex the resulting RNA gene-count matrices. We use DemuxEM method in this example.

To be brief, we use the output of Section 1-a for illustration:

1. On your local machine, prepare a CSV-format sample sheet `demux_sample_sheet.csv` with the following content:

   ```
   OUTNAME,RNA,TagFile,TYPE
   exp,gs://my-bucket/cellranger_output/sample_gex/raw_feature_bc_matrix.h5,gs://my-
   ↪bucket/cellranger_output/sample_cell_hashing/sample_cell_hashing.csv,cell-
   ↪hashing
   ```

   where **OUTNAME** specifies the subfolder and file names of output, which is free to be changed, **RNA** and **TagFile** columns specify the RNA and hashing tag meta-data of samples, and **TYPE** is `cell-hashing` for this phase.

2. On your local machine, also prepare an input JSON file `demux_inputs.json` for **demultiplexing** WDL workflow, `demux_inputs.json` with the following content:

   ```
   {
           "demultiplexing.input_sample_sheet" : "/path/to/demux_sample_sheet.csv",
           "demultiplexing.output_directory" : "gs://my-bucket/demux_output"
   }
   ```

   where `/path/to/demux_sample_sheet.csv` should be replaced by the actual local path to `demux_sample_sheet.csv` created above.

   For the details on these options, please refer to demultiplexing workflow inputs.

3. Submit a *demultiplexing* job with `demux_inputs.json` input above to cloud for execution.

For job submission:

- If you use a Cromwell server on cloud, run the following Altocumulus command on your local machine:

```
alto cromwell run -s <server-address> -p <port-number> -m␣
↪broadinstitute:cumulus:demultiplexing -i /path/to/demux_inputs.json -o demux_
↪inputs_updated.json -b gs://my-bucket/data_source
```

where

- `broadinstitute:cumulus:demultiplexing` refers to demultiplexing WDL workflow published on Dockstore. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. `broadinstitute:cumulus:demultiplexing:master` to use its development version in *master* branch.
- `/path/to/demux_inputs.json` should be replaced by the actual local path to `demux_inputs.json` created above.
- Replace `my-bucket` in `-b` option by your own Google bucket name, and feel free to choose folder name other than `data_source` for uploading.
- We still need `-o` and `-b` options because `demux_sample_sheet.csv` is on the local machine.

Similarly, when the submission succeeds, you'll get another job ID for demultiplexing. You can use it to track the job status.

- If you use Terra, run the following Altocumulus command:

```
alto terra run -m broadinstitute:cumulus:demultiplexing -w ws-lab/ws-01 --bucket-
↪folder data_source -i /path/to/demux_inputs.json -o demux_inputs_updated.json
```

where

- `broadinstitute:cumulus:demultiplexing` refers to demultiplexing WDL workflow published on Dockstore. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. `broadinstitute:cumulus:demultiplexing:master` to use its development version in *master* branch.
- `/path/to/demux_inputs.json` should be replaced by the actual local path to `demux_inputs.json` created above.
- `ws-lab/ws-01` should be replaced by your own Terra workspace full name.
- `--bucket-folder`: Feel free to choose folder name other than `data_source` for uploading.
- We still need `-o` and `--bucket-folder` options because `demux_sample_sheet.csv` is on the local machine.

After submission, you can check the job's status in the *Job History* tab of your Terra workspace page.

When finished, demultiplexing results are in `gs://my-bucket/demux_output/exp` folder, with the following important output files:

- `exp_demux.zarr.zip`: Demultiplexed RNA raw count matrix. This will be used for downstram analysis.
- `exp.out.demuxEM.zarr.zip`: This file contains intermediate results for both RNA and hashing count matrices, which is useful for compare with other demultiplexing methods.
- DemuxEM plots in PDF format. They are used for evaluating the performance of DemuxEM on the data.

(Optional) Extract Demultiplexing results

This is performed on your local machine with demultiplexing results downloaded from cloud to your machine.

To download the demultiplexed count matrix `exp_demux.zarr.zip`, you can either do it in Google cloud console, or using gsutil in command line:

```
gsutil -m cp gs://my-bucket/demux_output/exp/exp_demux.zarr.zip .
```

After that, in your Python environment, install Pegasus package, and follow the steps below to extract the demultiplexing results:

1. Load Libraries:

```
import numpy as np
import pandas as pd
import pegasus as pg
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load demuxEM output. For demuxEM, load RNA expression matrix with demultiplexed sample identities in Zarr format. These can be found in Google cloud console. QC 500 <= # of genes < 6000, % mito <= 10%:

```
data = pg.read_input('exp_demux.zarr.zip')
pg.qc_metrics(data, min_genes=500, max_genes=6000, mito_prefix='Mt-', percent_
↪mito=10)
pg.filter_data(data)
```

3. Demultiplexing results showing singlets, doublets and unknown:

```
data.obs['demux_type'].value_counts()
```

4. Show assignments in singlets:

```
idx = data.obs['demux_type'] == 'singlet'
data.obs.loc[idx, 'assignment'].value_counts()[0:10]
```

5. Write assignment outputs to CSV:

```
data.obs[['demux_type', 'assignment']].to_csv('demux_exp.csv')
```

---

3. Data Analysis on CITE-Seq Data

In this phase, we merge RNA and ADT matrices for CITE-Seq data, and perform the downstream analysis.

To be brief, we use the CITE-Seq count matrix generated from Section 1-a and demultiplexing results in Section 2 for illustraion here:

1. On your local machine, prepare a CSV-format sample sheet `count_matrix.csv` with the following content:

```
Sample,Location,Modality
exp,gs://my-bucket/demux_output/exp/exp_demux.zarr.zip,rna
exp,gs://my-bucket/cellranger_output/sample_cite_seq/sample_cite_seq.csv,citeseq
```

This sample sheet describes the metadata for each modality (as one row in the sheet):

- **Sample** specifies the name of the modality, and all the modalities of the same sample should have one common name, as otherwise their count matrices won't be aggregated together;
- **Location** specifies the file location. For RNA data, this is the output of Phase 2; for CITE-Seq antibody data, it's the output of Phase 1.
- **Modality** specifies the modality type, which is either `rna` for RNA matrix, or `citeseq` for CITE-Seq antibody matrix.

2. On your local machine, also prepare a JSON file `cumulus_inputs.json` for **cumulus** WDL workflow, with the following content:

```
{
        "cumulus.input_file": "/path/to/count_matrix.csv",
        "cumulus.output_directory": "gs://my-bucket/cumulus_output",
        "cumulus.output_name": "exp_merged_out",
        "cumulus.select_only_singlets": true,
        "cumulus.run_louvain": true,
        "cumulus.run_umap": true,
        "cumulus.citeseq": true,
        "cumulus.citeseq_umap": true,
        "cumulus.citeseq_umap_exclude": "Mouse_IgG1,Mouse_IgG2a,Mouse_IgG2b,Rat_
↪IgG2b",
        "cumulus.plot_composition": "louvain_labels:assignment",
        "cumulus.plot_umap": "louvain_labels,assignment",
        "cumulus.plot_citeseq_umap": "louvain_labels,assignment",
        "cumulus.cluster_labels": "louvain_labels",
        "cumulus.annotate_cluster": true,
        "cumulus.organism": "human_immune"
}
```

   where `/path/to/count_matrix.csv` should be replaced by the actual local path to `count_matrix.csv` created above.

   A typical Cumulus WDL pipeline consists of 4 steps, which is given here. For details on Cumulus workflow inputs above, please refer to cumulus inputs.

3. Submit a *demultiplexing* job with `cumulus_inputs.json` input above to cloud for execution.

For job submission:

- If you use a Cromwell server on cloud, run the following Altocumulus command to submit the job:

```
alto cromwell run -s <server-address> -p <port-number> -m␣
↪broadinstitute:cumulus:cumulus -i /path/to/cumulus_inputs.json -o cumulus_
↪inputs_updated.json -b gs://my-bucket/data_source
```

where

- `broadinstitute:cumulus:cumulus` refers to cumulus WDL workflow published on Dockstore. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. `broadinstitute:cumulus:cumulus:master` to use its development version in *master* branch.
- `/path/to/cumulus_inputs.json` should be replaced by the actual local path to `cumulus_inputs.json` created above.
- `my-bucket` in `-b` option should be replaced by your own Google bucket name, and feel free to choose folder name other than `data_source` for uploading data.
- We still need `-o` and `-b` options because `count_matrix.csv` is on the local machine.

Similarly, when the submission succeeds, you'll get another job ID for demultiplexing. You can use it to track the job status.

- If you use Terra, run the following Altocumulus command:

```
alto terra run -m broadinstitute:cumulus:cumulus -w ws-lab/ws-01 --bucket-folder␣
↪data_source -i /path/to/cumulus_inputs.json -o cumulus_inputs_updated.json
```

where

- `broadinstitute:cumulus:cumulus` refers to cumulus WDL workflow published on Dockstore. Here, the version is omitted, so that the default version will be used. Alternatively, you can explicitly specify which version to use, e.g. `broadinstitute:cumulus:cumulus:master` to use its development version in *master* branch.
- `ws-lab/ws-01` should be replaced by your own Terra workspace full name.
- `--bucket-folder`: Feel free to choose folder name other than `data_source` for uploading data.
- `/path/to/cumulus_inputs.json` should be replaced by the actual local path to `cumulus_inputs.json` created above.
- We still need `-o` and `--bucket-folder` options because `count_matrix.csv` is on the local machine.

After submission, you can check the job's status in the *Job History* tab of your Terra workspace page.

When finished, all the output files are in `gs://my-bucket/cumulus_output` folder, with the following important files:

- `exp_merged_out.aggr.zarr.zip`: The *ZARR* format file containing both the aggregated count matrix in `<genome>-rna` modality, as well as CITE-Seq antibody count matrix in `<genome>-citeseq` modality, where `<genome>` is the genome reference name of your count matrices, e.g. *GRCh38-2020-A*.
- `exp_merged_out.zarr.zip`: The *ZARR* format file containing the analysis results in `<genome>-rna` modality, and CITE-Seq antibody count matrix in `<genome>-citeseq` modality.
- `exp_merged_out.<genome>-rna.h5ad`: The processed RNA matrix data in *H5AD* format.
- `exp_merged_out.<genome>-rna.filt.xlsx`: The Quality-Control (QC) summary of the raw data.
- `exp_merged_out.<genome>-rna.filt.{UMI, gene, mito}.pdf`: The QC plots of the raw data.
- `exp_merged_out.<genome>-rna.de.xlsx`: Differential Expression analysis result.
- `exp_merged_out.<genome>-rna.anno.txt`: The putative cell type annotation output.
- `exp_merged_out.<genome>-rna.umap.pdf`: UMAP plot.
- `exp_merged_out.<genome>-rna.citeseq.umap.pdf`: CITE-Seq UMAP plot.
- `exp_merged_out.<genome>-rna.louvain_labels.assignment.composition.pdf`: Composition plot.

Run Analysis with Terra Web UI

For Terra users, instead of using Altocumulus to submit jobs in command line, they can also use the Terra web UI.

First, upload the local BCL data or FASTQ files to the Google bucket associated with your Terra workspace (say `gs://fc-e0000000`) using gsutil:

```
gsutil -m cp -r /path/to/your/data/folder gs://fc-e0000000/data_source/
```

where `/path/to/your/data/folder` should be replaced by the actual local path to your data folder, and `data_source` is the folder on Google bucket to store the uploaded data.

Then for each of the 3 phases above:

1. When preparing the sample sheet, remember to replace all the local paths by the GS URIs of the corresponding folders/files that you uploaded to Google bucket. Then upload it to Google bucket as well:

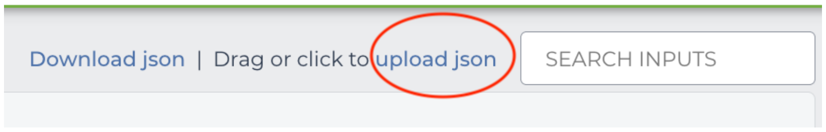
   ```
   gsutil cp /path/to/sample/sheet gs://fc-e0000000/data_source/
   ```

   where `/path/to/sample/sheet` should be replaced by the actual local path to your sample sheet. Notice that for Phase 1, `antibody_index.csv` file should also be uploaded to Google bucket, and its references in the sample sheet must be replaced by its GS URI.

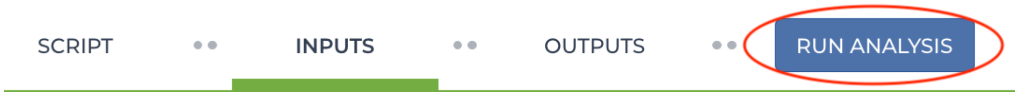
2. When preparing the workflow input JSON file, change the field of sample sheet to its GS URI on cloud.

3. Import the corresponding WDL workflow to your Terra workspace by following steps in import workflows tutorial.

4. In the workflow page (Workspace -> Workflows -> your WDL workflow), upload your input JSON file by clicking the "*upload json*" button:



5. Click "*SAVE*" button to save the configuration, and click "*RUN ANALYSIS*" button to submit the job:



You can check the job's status in the *Job History* tab of your Terra workspace page.

Example of 10X Genomics CellPlex Analysis on Cloud

In this example, you'll learn how to perform Cellplex analysis on Cloud using Cromwell.

---

0. Prerequisite

You need to install the corresponding Cloud SDK tool on your local machine if not:

- gcloud CLI for Google Cloud.
- AWS CLI v2 for Amazon AWS Cloud.

And then install Altocumulus in your Python environment. This is the tool for data transfer between local machine and Cloud VM instance.

In this example, we assume that your Cromwell server is already deployed on Cloud at IP address `10.0.0.0` with port `8000`, and also assume using Google Cloud with bucket `gs://my-bucket`.

1. Extract Genen-Count Matrices

First step is to extract gene-count matrices from sequencer output.

In this example, we have the following experiment setting:

- A sample named `cellplex_gex` by pooling all RNA data together for sequencing, with index `SI-TT-A1`;
- A sample named `cellplex_barcode` for hashing data, with index `SI-NN-A1`;
- Three samples to perform individual control:
  - Sample `A` with index `SI-TT-A2` and CMO ID `CMO_301`,
  - Sample `B` with index `SI-TT-A3` and CMO ID `CMO_302`,
  - Sample `C` with index `SI-TT-A4` and CMO ID `CMO_303`

To extract feature barcodes for the hashing data, we need to create a feature barcoding file (say named `feature_barcode.csv`). Please refer to 10X Multi CMO Reference for the sequence information of these CMO IDs:

```
ATGAGGAATTCCTGC,A
CATGCCAATAGAGCG,B
CCGTCGTCCAAGCAT,C
```

After that, create a sample sheet in CSV format (say named `cellranger_sample_sheet.csv`) as the following:

```
Sample,Reference,Flowcell,Lane,Index,DataType,FeatureBarcodeFile
cellplex_gex,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A1,rna
cellplex_barcode,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-NN-A1,cmo,/path/to/
↪feature_barcode.csv
A,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A2,rna
B,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A3,rna
C,GRCh38-2020-A,/path/to/flowcell/folder,*,SI-TT-A4,rna
```

where

- `GRCh38-2020-A` is the Human GRCh38 (GENCODE v32/Ensembl 98) genome reference prebuilt by Cumulus. See Cumulus single-cell genome reference list for a complete list of genome references.
- `/path/to/flowcell/folder` should be replaced by the local path to the BCL folder of your sequencer output.
- `/path/to/feature_barcode.csv` should be replaced by the local path to `feature_barcode.csv` file we just created above.
- `rna` and `cmo` refer to gene expression data and cell multiplexing oligos used in 10X Genomics CellPlex assay, respectively.
- Only the sample of `cmo` type needs a feature barcode file for indexing.

For details on preparing this sample sheet, please refer to CellRanger workflow sample sheet format.

Now let's prepare an input JSON file for **cellranger_workflow** WDL workflow to execute (say named `cellranger_inputs.json`):

```
{
    "cellranger_workflow.input_csv_file": "/path/to/cellranger_sample_sheet.csv",
    "cellranger_workflow.output_directory": "gs://my-bucket/cellplex/cellranger_output
↪"
}
```

where

- `/path/to/cellranger_sample_sheet.csv` should be replaced by the local path to your sample sheet created above.
- `gs://my-bucket/cellplex/cellranger_output` is the target folder on Google bucket to store your result when the workflow job is finished.

For details on these workflow inputs, please refer to CellRanger workflow inputs.

Now we are ready to submit a job to the Cromwell server on Cloud for computing. On your local machine, run the following command:

```
alto cromwell run -s 10.0.0.0 -p 8000 -m broadinstitute:cumulus:cellranger:master -i /
↪path/to/cellranger_inputs.json -o cellranger_inputs_updated.json -b gs://my-bucket/
↪cellplex
```

where

- `-s` to specify the server's IP address (or hostname), `-p` to specify the server's port number.
- `-m` to specify which WDL workflow to use. You should use the Dockstore name of Cumulus cellranger_workflow. Here, the latest version `master` is used. If omit the version info, i.e. `broadinstitute:cumulus:cellranger`, the default version will be used.
- `-i` to specify the workflow input JSON file.
- `-o` and `-b` are used when the input data are local and need to be uploaded to Cloud bucket first. This can be inferred from the workflow input JSON file and sample sheet CSV file.
- `-o` to specify the updated workflow input JSON file after uploading the input data, with all the local paths updated to Cloud bucket URLs.
- `-b` to specify which folder on Cloud bucket to upload the local input data.

Notice that `-o` and `-b` options can be dropped if all of your input data are already on Cloud bucket.

After submission, you'll get the job's ID for tracking its status:

```
alto cromwell check_status -s 10.0.0.0 -p 8000 --id <your-job-ID>
```

where `<your-job-ID>` should be replaced by the actual Cromwell job ID.

When the job is done, you'll get results in `gs://my-bucket/cellplex/cellranger_output`. It should contain 6 subfolders, each of which is associated with one sample in `cellranger_sample_sheet.csv`.

2. Demultiplexing

Next, we need to demultiplex the resulting gene-count matrices. In this example, we perform both DemuxEM and Souporcell methods, respectively.

For **DemuxEM**, we'll need the RNA raw count matrix in HDF5 format (`gs://my-bucket/cellplex/cellranger_output/cellplex_gex/raw_feature_bc_matrix.h5`) and the hashing count matrix in CSV format (`gs://my-buckjet/cellplex/cellranger_output/cellplex_barcode/cellplex_barcode.csv`).

For **Souporcell**, both the RNA raw count matrix above and its corresponding BAM file (`gs://my-bucket/cellplex/cellranger_output/cellplex_gex/possorted_genome_bam.bam`) are needed.

Prepare a sample sheet in CSV format (say named `demux_sample_sheet.csv`) for demultiplexing, one line for DemuxEM, one for Souporcell:

```
OUTNAME,RNA,TagFile,TYPE
cellplex_demux,gs://my-bucket/cellplex/cellranger_output/cellplex_gex/raw_feature_bc_
↪matrix.h5,gs://my-buckjet/cellplex/cellranger_output/cellplex_barcode/cellplex_
↪barcode.csv,cell-hashing
cellplex_souporcell,gs://my-bucket/cellplex/cellranger_output/cellplex_gex/raw_
↪feature_bc_matrix.h5,gs://my-bucket/cellplex/cellranger_output/cellplex_gex/
↪possorted_genome_bam.bam,genetic-pooling
```

where

- `cell-hashing` indicates using DemuxEM for demultiplexing, while `genetic-pooling` indicates using genetic pooling methods for demultiplexing, with Souporcell being the default.

For details on this sample sheet, please refer to Demultiplexing workflow sample sheet format.

Then prepare a workflow input JSON file (say named `demux_inputs.json`) for demultiplexing:

```
{
    "demultiplexing.input_sample_sheet": "/path/to/demux_sample_sheet.csv",
    "demultiplexing.output_directory": "gs://my-bucket/cellplex/demux_output",
    "demultiplexing.genome": "GRCh38-2020-A",
    "demultiplexing.souporcell_num_clusters": 3
}
```

where

- `/path/to/demux_sample_sheet.csv` should be replaced by the local path to your `demux_sample_sheet.csv` created above.
- `gs://my-bucket/cellplex/demux_output` is the Bucket folder to write the results when the job is finished.
- `GRCh38-2020-A` is the genome reference used by Souporcell, which should be consistent with your settings in Step 1.
- `souporcell_num_clusters` is to set the number of clusters you expect to see for Souporcell clustering. Since we have 3 donors, so set it to 3.

For details, please refer to Demultiplexing workflow inputs.

Now submit the demultiplexing job to Cromwell server on Cloud:

```
alto cromwell run -s 10.0.0.0 -p 8000 -m broadinstitute:cumulus:demultiplexing:master␣
↪-i demux_inputs.json -o demux_inputs_updated.json -b gs://my-bucket/cellplex
```

where

- `broadinstitute:cumulus:demultiplexing` refers to demultiplexing workflow published on Dockstore.
- We still need `-o` and `-b` options because `demux_sample_sheet.csv` is on the local machine.

Similarly, when the submission succeeds, you'll get another job ID for demultiplexing. You can use it to track the job status.

When finished, below are the important output files:

- DemuxEM output: In folder `gs://my-bucket/cellplex/demux_output/cellplex_demux`,
  - `cellplex_demux_demux.zarr.zip`: Demultiplexed RNA raw count matrix. This will be used for downstream analysis.

    - `cellplex_demux.out.demuxEM.zarr.zip`: This file contains intermediate results for both RNA and hashing count matrices, which is useful for compare with other demultiplexing methods.
    - DemuxEM plots in PDF format. They are used for estimating the performance of DemuxEM on the data.
- Souporcell output: In folder `gs://my-bucket/cellplex/demux_output/cellplex_souporcell`,
    - `cellplex_souporcell_demux.zarr.zip`: Demultiplexed RNA raw count matrix. This will be used for downstream analysis.
    - `clusters.tsv`: Inferred droplet type and cluster assignment for each cell barcode.
    - `cluster_genotypes.vcf`: Inferred genotypes for each cluster.

3. Interactive Data Analysis

You may use Cumulus workflow to perform the downstream analysis in a batch way. Alternatively, you can also download the demultiplexing results from the Cloud bucket to your local machine, and perform the analysis interactively. This section introduces how to use Cumulus' analysis module Pegasus to load demultiplexing results, perform quality control (QC), and compare the performance of the two methods.

You'll need to first install Pegasus in your local Python environment. Also, download the demultiplexed raw counts in `.zarr.zip` format mentioned above to your local machine.

3.1. Extract Singlet/Doublet Type and Assignment

We can load the DemuxEM result, and perform QC by:

```
import pegasus as pg
data_demuxEM = pg.read_input("cellplex_demux_demux.zarr.zip")
pg.qc_metrics(data_demuxEM, min_genes=500, max_genes=6000, mito_prefix='MT-', percent_
↪mito=20)
pg.filter_data(data_demuxEM)
```

where `qc_metrics` and `filter_data` are Pegasus functions to filter out low quality cells, and keep those with number of genes within range `[500, 6000)` and having expression of mitochondrial genes < `20%`. Please see Pegasus preprocess tools for details.

There are two columns in *data_demuxEM.obs* field related to demultiplexing results:

- **demux_type**: This column stores the singlet/doublet type of each cell: `singlet`, `doublet`, or `unknown`.
- **assignment**: This column stores the more detailed assignment of cells regarding samples/donors.

To get the distribution regarding these columns, e.g. *demux_type*:

```
data_demuxEM.obs['demux_type'].value_counts()
```

Besides, you can export the cell barcodes along with their singlet/doublet type and assignment as a CSV file by:

```
data_demuxEM.obs[['demux_type', 'assignment']].to_csv("demuxEM_assignment.csv")
```

We can also do it similarly for the Souporcell result as above, by reading `cellplex_souporcell_demux.zarr.zip` instead.

3.2. Compare the Two Demultiplexing Methods

We can compare the performance of DemuxEM and Souporcell by plotting a heatmap showing their singlet/doublet assignment results.

Assume we've already loaded the two results (`data_demuxEM` for DemuxEM result, `data_souporcell` for Souporcell result), and performed QC as in 3.1. The following Python code will generate this heatmap in an interactive Python environment (e.g. in a Jupyter notebook):

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def extract_assignment(data):
    assign = data.obs['demux_type'].values.astype('object')
    idx_singlet = (data.obs['demux_type'] == 'singlet').values
    assign[idx_singlet] = data.obs.loc[idx_singlet, 'assignment'].values.
↪astype(object)
    return assign

assign_demuxEM = extract_assignment(data_demuxEM)
assign_souporcell = extract_assignment(data_souporcell)

df = pd.crosstab(assign_demuxEM, assign_souporcell)
df.columns.name = df.index.name = ""
ax = plt.gca()
ax.xaxis.tick_top()
ax = sns.heatmap(df, annot=True, fmt='d', cmap='inferno', ax=ax)
plt.tight_layout()
plt.gcf().dpi=500
```

3.3. Downstream Analysis

To perform further downstream analysis on the singlets, please refer to Pegasus tutorials.

Examples using Terra to perform single-cell sequencing analysis are provided here. Please click the topics on the left panel under title **"Examples"** to explore.

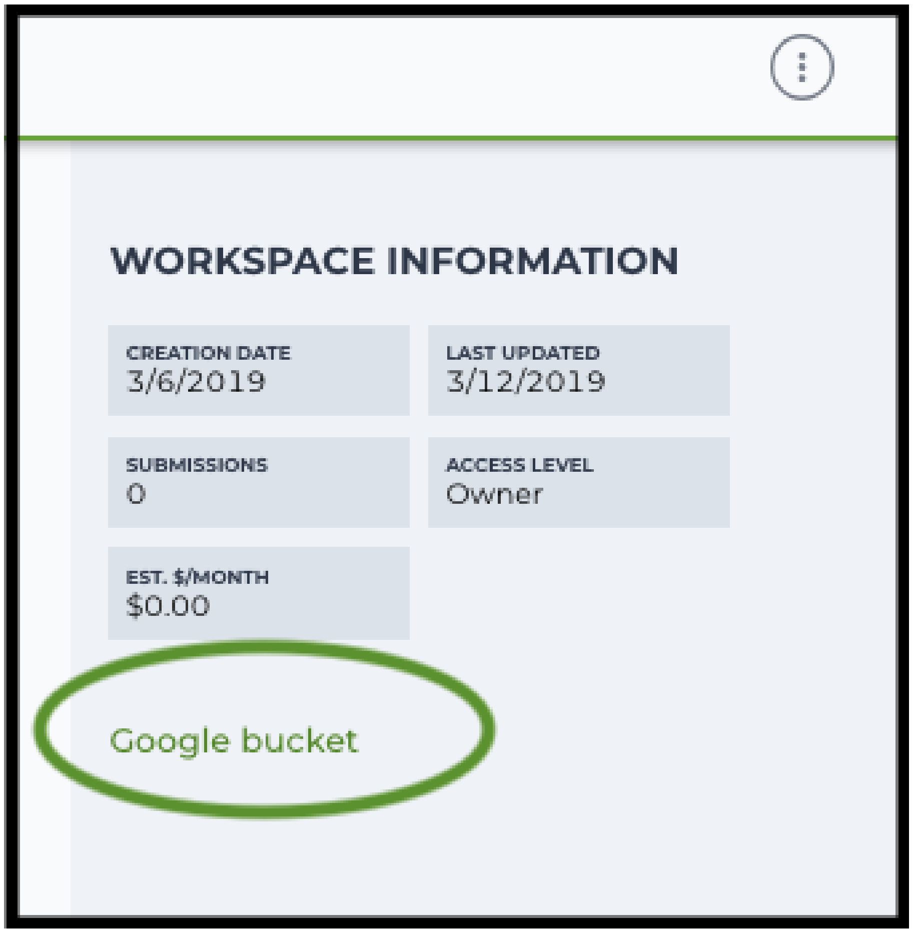
1.1.12 Extract gene-count matrices from plated-based SMART-Seq2 data

Run SMART-Seq2 Workflow

Follow the steps below to extract gene-count matrices from SMART-Seq2 data on Terra. This WDL aligns reads using *STAR*, *HISAT2*, or *Bowtie 2* and estimates expression levels using *RSEM*.

1. Copy your sequencing output to your workspace bucket using gsutil in your unix terminal.

   You can obtain your bucket URL in the dashboard tab of your Terra workspace under the information panel.



Use `gsutil cp [OPTION]... src_url dst_url` to copy data to your workspace bucket. For example, the following command copies the directory at /foo/bar/nextseq/Data/VK18WBC6Z4 to a Google bucket:

```
gsutil -m cp -r /foo/bar/nextseq/Data/VK18WBC6Z4 gs://fc-e0000000-0000-
↪0000-0000-000000000000/VK18WBC6Z4
```

`-m` means copy in parallel, `-r` means copy the directory recursively.

2. Create a sample sheet.

   Please note that the columns in the TSV can be in any order, but that the column names must match the recognized headings.

   The sample sheet provides metadata for each cell:

Column	Description
entity:sample	Cell name.
plate	Plate name. Cells with the same plate name are from the same plate.
read1	Location of the FASTQ file for read1 in the cloud (gsurl).
read2	(Optional). Location of the FASTQ file for read2 in the cloud (gsurl). This field can be skipped for single-end reads.

   Example:

```
entity:sample   plate   read1   read2
cell-1  plate-1 gs://fc-e0000000-0000-0000-0000-000000000000/smartseq2/
↪cell-1_L001_R1_001.fastq.gz      gs://fc-e0000000-0000-0000-0000-
↪000000000000/smartseq2/cell-1_L001_R2_001.fastq.gz
cell-2  plate-1 gs://fc-e0000000-0000-0000-0000-000000000000/smartseq2/
↪cell-2_L001_R1_001.fastq.gz      gs://fc-e0000000-0000-0000-0000-
↪000000000000/smartseq2/cell-2_L001_R2_001.fastq.gz
cell-3  plate-2 gs://fc-e0000000-0000-0000-0000-000000000000/smartseq2/
↪cell-3_L001_R1_001.fastq.gz
cell-4  plate-2 gs://fc-e0000000-0000-0000-0000-000000000000/smartseq2/
↪cell-4_L001_R1_001.fastq.gz
```

3. Upload your sample sheet to the workspace bucket.

   Example:

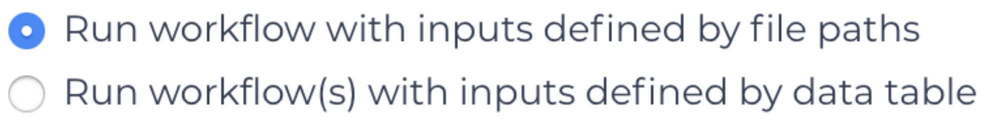
```
gsutil cp /foo/bar/projects/sample_sheet.csv gs://fc-e0000000-0000-0000-
↪0000-000000000000/
```

4. Import *smartseq2* workflow to your workspace.

   Import by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Smart-Seq2** to import.

   Moreover, in the workflow page, click `Export to Workspace...` button, and select the workspace to which you want to export *smartseq2* workflow in the drop-down menu.

5. In your workspace, open `smartseq2` in `WORKFLOWS` tab. Select `Run workflow with inputs defined by file paths` as below



[✓ Run workflow with inputs defined by file paths]
Run workflow(s) with inputs defined by data table

   and click `SAVE` button.

Inputs:

Please see the description of inputs below. Note that required inputs are shown in bold.

Name	Description	Example	Default
input_tsv_file	Sample Sheet (contains entity:sample, plate, read1, read2)	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.tsv"	
output_directory	Output directory	"gs://fc-e0000000-0000-0000-0000-000000000000/smartseq2_output"	
reference	Reference transcriptome to align reads to. Acceptable values: • Pre-created genome references: – "GRCh38_ens93filt" for human, genome version is GRCh38, gene annotation is generated using human Ensembl 93 GTF according to cellranger mkgtf; – "GRCm38_ens93filt" for mouse, genome version is GRCm38, gene annotation is generated using mouse Ensembl 93 GTF according to cellranger mkgtf; • Create a custom genome reference using smartseq2_create_reference workflow, and specify its Google bucket URL here.	"GRCh38_ens93filt", or "gs://fc-e0000000-0000-0000-0000-000000000000/rsem_ref.tar.gz"	
aligner	Which aligner to use for read alignment. Options are "hisat2-hca", "star" and "bowtie"	"star"	"hisat2-hca"
output_genome_bam	Whether to output bam file with alignments mapped to genomic coordinates and annotated with their posterior probabilities.	false	false
smartseq2_version	SMART-Seq2 version to use. Versions available: 1.3.0.	"1.3.0"	"1.3.0"
docker_registry	Docker registry to use. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of cpus to request for one node	4	4
memory	Memory size string	"3.60G"	If

Outputs:

Name	Type	Description
output_count_matrix	String	Point to a Google bucket URL for count matrix in matrix market format.
rsem_trans_bam	Array[String?]	An array of Google bucket URLs for RSEM transcriptomic BAM files
rsem_genome_bam	Array[String?]	An array of Google bucket URLs for RSEM genomic BAM files if `output_genome_bam` is `true`.
rsem_gene	Array[File?]	An array of RSEM gene expression estimation files.
rsem_isoform	Array[File?]	An array of RSEM isoform expression estimation files.
rsem_time	Array[File?]	An array of RSEM execution time log files.
aligner_log	Array[File?]	An array of Aligner log files.
rsem_cnt	Array[File?]	An array of RSEM count files.
rsem_model	Array[File?]	An array of RSEM model files.
rsem_theta	Array[File?]	An array of RSEM generated theta files.

This WDL generates one gene-count matrix in matrix market format:

- output_count_matrix is a folder containing three files: matrix.mtx.gz, barcodes.tsv.gz, and features.tsv.gz.
- matrix.mtx.gz is a gzipped matrix in matrix market format.
- barcodes.tsv.gz is a gzipped TSV file, containing 5 columns. 'barcodekey' is cell name. 'plate' is the plate name, which can be used for batch correction. 'total_reads' is the total number of reads. 'alignment_rate' is the alignment rate obtained from the aligner. 'unique_rate' is the percentage of reads aligned uniquely to a gene. Cells sequenced with single-end reads appear first in 'barcodekey'.
- features.tsv.gz is a gzipped TSV file, containing 2 columns. 'featurekey' is gene symbol. 'featureid' is Ensembl ID.

The gene-count matrix can be fed directly into **cumulus** for downstream analysis.

TPM-normalized counts are calculated as follows:

1. Estimate the gene expression levels in TPM using *RSEM*.
2. Suppose `c` reads are achieved for one cell, then calculate TPM-normalized count for gene `i` as `TPM_i / 1e6 * c`.

TPM-normalized counts reflect both the relative expression levels and the cell sequencing depth.

Custom Genome

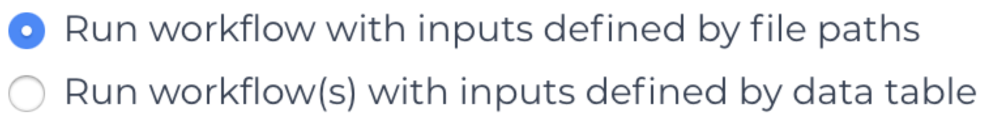
We also provide a way of generating user-customized Genome references for SMART-Seq2 workflow.

1. Import smartseq2_create_reference workflow to your workspace.

   Import by following instructions in Import workflows to Terra. You should choose **github.com/lilab-bcb/cumulus/Smart-Seq2_create_reference** to import.

Moreover, in the workflow page, click `Export to Workflow...` button, and select the workspace to which you want to export `smartseq2_create_reference` in the drop-down menu.

2. In your workspace, open `smartseq2_create_reference` in `WORKFLOWS` tab. Select `Run workflow with inputs defined by file paths` as below


Run workflow with inputs defined by file paths
Run workflow(s) with inputs defined by data table

and click `SAVE` button.

Inputs:

Please see the description of inputs below. Note that required inputs are shown in bold.

Name	Description	Type or Example	Default
fasta	Genome fasta file	File. For example, "gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.dna.primary_assembly.fa"	
gtf	GTF gene annotation file (e.g. Homo_sapiens.GRCh38.83.gtf)	File. For example, "gs://fc-e0000000-0000-0000-0000-000000000000/Homo_sapiens.GRCh38.83.gtf"	
output_directory	Google bucket url for the output folder	"gs://fc-e0000000-0000-0000-0000-000000000000/output_refs"	
genome	Output reference genome name. Output reference is a gzipped tarball with name genome_aligner.tar.gz	"GRCm38_ens97filt"	
aligner	Build indices for which aligner, choices are hisat2-hca, star, or bowtie2.	"hisat2-hca"	"hisat2-hca"
smartseq2_version	SMART-Seq2 version to use. Versions available: 1.3.0.	"1.3.0"	"1.3.0"
docker_registry	Docker registry to use. Options: • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
zones	Google cloud zones	"us-central1-c"	"us-central1-b"
cpu	Number of CPUs	Integer	If aligner is bowtie2 or hisat2-hca, 8; otherwise 32
memory	Memory size string	String	If aligner is bowtie2 or hisat2-hca, "7.2G"; otherwise "120G"

Outputs

Name	Type	Description
output_reference	File	The custom Genome reference generated. Its default file name is `genome_aligner.tar.gz`.
monitoring_log	File	CPU and memory profiling log.

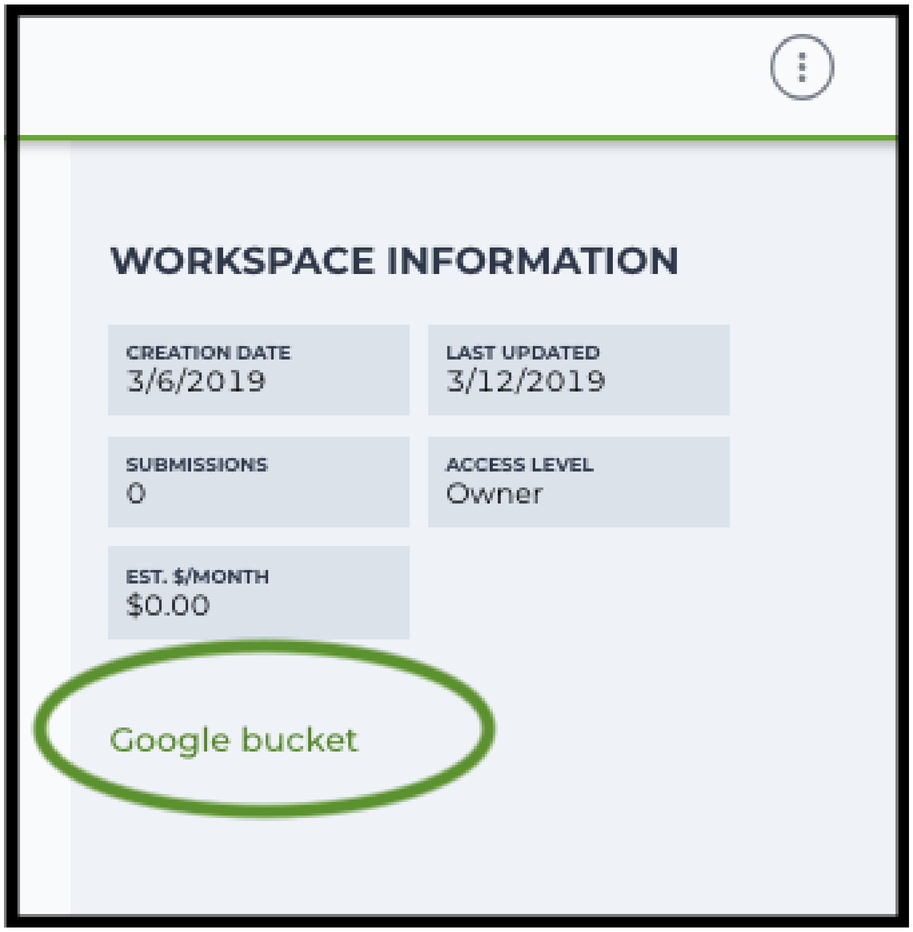
1.1.13 Bulk RNA-Seq

Run Bulk RNA-Seq Workflow

Follow the steps below to generate count matrices from bulk RNA-Seq data on Terra. This WDL estimates expression levels using *RSEM*.

1. Copy your sequencing output to your workspace bucket using gsutil in your unix terminal.

   You can obtain your bucket URL in the dashboard tab of your Terra workspace under the information panel.



   Note: Broad users need to be on an UGER node (not a login node) in order to use the `-m` flag

   Request an UGER node:

```
reuse UGER
qrsh -q interactive -l h_vmem=4g -pe smp 8 -binding linear:8 -P regevlab
```

The above command requests an interactive node with 4G memory per thread and 8 threads. Feel free to change the memory, thread, and project parameters.

Once you're connected to an UGER node, you can make gsutil available by running:

```
reuse Google-Cloud-SDK
```

Use `gsutil cp [OPTION]... src_url dst_url` to copy data to your workspace bucket. For example, the following command copies the directory at /foo/bar/nextseq/Data/VK18WBC6Z4 to a Google bucket:

```
gsutil -m cp -r /foo/bar/nextseq/Data/VK18WBC6Z4 gs://fc-e0000000-0000-
↪0000-0000-000000000000/VK18WBC6Z4
```

`-m` means copy in parallel, `-r` means copy the directory recursively.

2. Create a Terra data table

   Example:

```
entity:sample_id  read1 read2
sample-1  gs://fc-e0000000/data-1/sample1-1_L001_R1_001.fastq.gz    gs://
↪fc-e0000000/data-1/sample-1_L001_R2_001.fastq.gz
sample-2 gs://fc-e0000000/data-1/sample-2_L001_R1_001.fastq.gz  gs://fc-
↪e0000000/data-1/sample-2_L001_R2_001.fastq.gz
```

   You are free to add more columns, but sample ids and URLs to fastq files are required.

3. Upload your TSV file to your workspace. Open the `DATA` tab on your workspace. Then click the upload button on left `TABLE` panel, and select the TSV file above. When uploading is done, you'll see a new data table with name "sample":

4. Import *bulk_rna_seq* workflow to your workspace. Then open `bulk_rna_seq` in the `WORKFLOW` tab. Select `Run workflow(s) with inputs defined by data table`, and choose *sample* from the drop-down menu.

Inputs:

Please see the description of important inputs below. Note that required inputs are in bold.

Name	Description	Default
sample_name	Sample name	
read1	Array of URLs to read 1	
read2	Array of URLs to read 2	
reference	**Reference to align reads to** • **Pre-created genome references:** – "GRCh38_ens93filt" for human, genome version is GRCh38, gene annotation is generated using human Ensembl 93 GTF according to cellranger mkgtf; – "GRCm38_ens93filt" for mouse, genome version is GRCm38, gene annotation is generated using mouse Ensembl 93 GTF according to cellranger mkgtf; • Create a custom genome reference using smart-seq2_create_reference workflow, and specify its Google bucket URL here.	
aligner	Which aligner to use for read alignment. Options are "hisat2-hca", "star" and "bowtie"	"star"
output_genome_bam	Whether to output bam file with alignments mapped to genomic coordinates and annotated with their posterior probabilities.	false

Outputs:

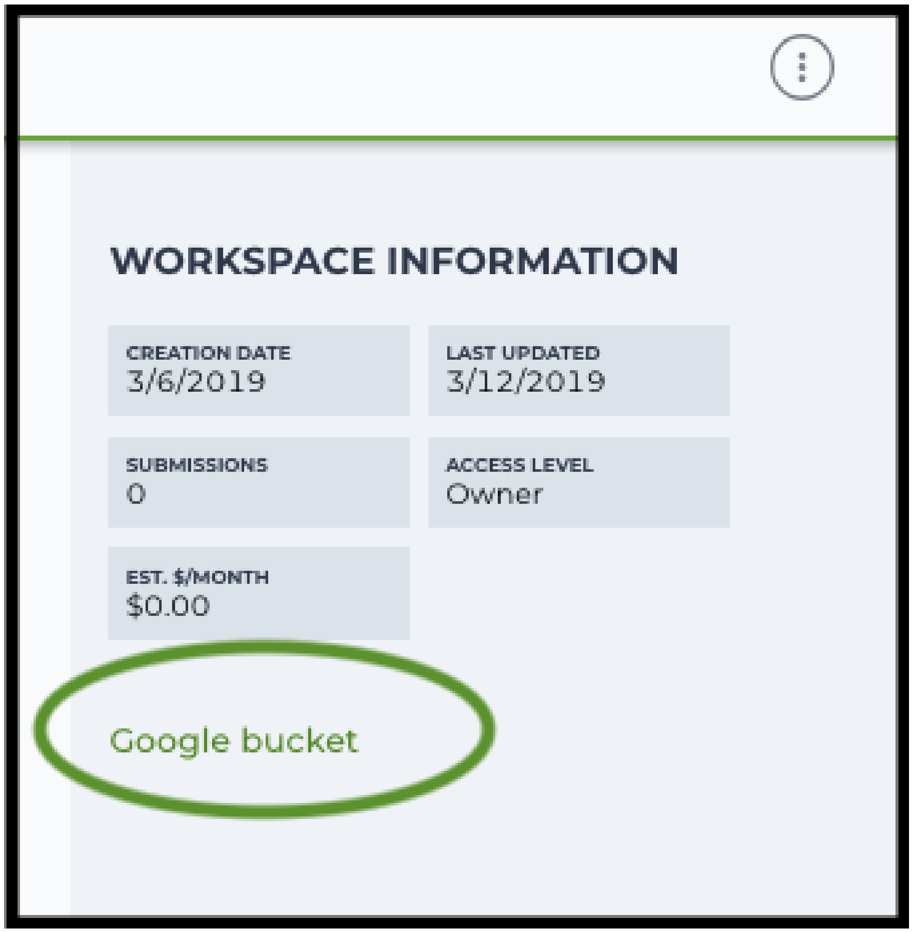
Name	Description
rsem_gene	RSEM gene expression estimation.
rsem_isoform	RSEM isoform expression estimation.
rsem_trans_bam	RSEM transcriptomic BAM.
rsem_genome_bam	RSEM genomic BAM files if `output_genome_bam` is `true`.
rsem_time	RSEM execution time log.
aligner_log	Aligner log.
rsem_cnt	RSEM count.
rsem_model	RSEM model.
rsem_theta	RSEM theta.

1.1.14 Drop-seq pipeline

This workflow follows the steps outlined in the Drop-seq alignment cookbook from the McCarroll lab , except the default STAR aligner flags are *–limitOutSJcollapsed 1000000 –twopassMode Basic*. Additionally the pipeline provides the option to generate count matrices using dropEst.

1. Copy your sequencing output to your workspace bucket using gsutil in your unix terminal.

   You can obtain your bucket URL in the dashboard tab of your Terra workspace under the information panel.



Note: Broad users need to be on an UGER node (not a login node) in order to use the `-m` flag

Request an UGER node:

```
reuse UGER
qrsh -q interactive -l h_vmem=4g -pe smp 8 -binding linear:8 -P regevlab
```

The above command requests an interactive node with 4G memory per thread and 8 threads. Feel free to change the memory, thread, and project parameters.

Once you're connected to an UGER node, you can make gsutil available by running:

```
reuse Google-Cloud-SDK
```

Use `gsutil cp [OPTION]... src_url dst_url` to copy data to your workspace bucket. For example, the following command copies the directory at /foo/bar/nextseq/Data/VK18WBC6Z4 to a Google bucket:

```
gsutil -m cp -r /foo/bar/nextseq/Data/VK18WBC6Z4 gs://fc-e0000000-0000-
↪0000-0000-000000000000/VK18WBC6Z4
```

`-m` means copy in parallel, `-r` means copy the directory recursively.

2. Non Broad Institute users that wish to run bcl2fastq must create a custom docker image.

   See *bcl2fastq* instructions.

3. Create a sample sheet.

Please note that the columns in the CSV must be in the order shown below and does not contain a header line. The sample sheet provides either the FASTQ files for each sample if you've already run bcl2fastq or a list of BCL directories if you're starting from BCL directories. Please note that BCL directories must contain a valid bcl2fastq sample sheet (SampleSheet.csv):

Column	Description
Name	Sample name.
Read1	Location of the FASTQ file for read1 in the cloud (gsurl).
Read2	Location of the FASTQ file for read2 in the cloud (gsurl).

Example using FASTQ input files:

```
sample-1,gs://fc-e0000000-0000-0000-0000-000000000000/dropseq-1/sample1-1_
↪L001_R1_001.fastq.gz,gs://fc-e0000000-0000-0000-0000-000000000000/
↪dropseq-1/sample-1_L001_R2_001.fastq.gz
sample-2,gs://fc-e0000000-0000-0000-0000-000000000000/dropseq-1/sample-2_
↪L001_R1_001.fastq.gz,gs://fc-e0000000-0000-0000-0000-000000000000/
↪dropseq-1/sample-2_L001_R2_001.fastq.gz
sample-1,gs://fc-e0000000-0000-0000-0000-000000000000/dropseq-2/sample1-1_
↪L001_R1_001.fastq.gz,gs://fc-e0000000-0000-0000-0000-000000000000/
↪dropseq-2/sample-1_L001_R2_001.fastq.gz
```

Note that in this example, sample-1 was sequenced across two flowcells.

Example using BCL input directories:

```
gs://fc-e0000000-0000-0000-0000-000000000000/flowcell-1
gs://fc-e0000000-0000-0000-0000-000000000000/flowcell-2
```

Note that the flow cell directory must contain a bcl2fastq sample sheet named SampleSheet.csv.

4. Upload your sample sheet to the workspace bucket.

   Example:

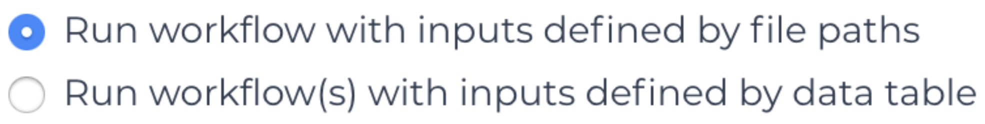
   ```
   gsutil cp /foo/bar/projects/sample_sheet.csv gs://fc-e0000000-0000-0000-
   ↪0000-000000000000/
   ```

5. Import *dropseq_workflow* workflow to your workspace.

   See the Terra documentation for adding a workflow. The *dropseq_workflow* is under `Broad Methods Repository` with name "**cumulus/dropseq_workflow**".

   Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace you want to export *dropseq_workflow* workflow in the drop-down menu.

6. In your workspace, open `dropseq_workflow` in `WORKFLOWS` tab. Select `Run workflow with inputs defined by file paths` as below



● Run workflow with inputs defined by file paths

○ Run workflow(s) with inputs defined by data table

   and click the `SAVE` button.

Inputs

Please see the description of important inputs below.

Name	Description
input_csv_file	CSV file containing sample name, read1, and read2 or a list of BCL directories.
output_directory	Pipeline output directory (gs URL e.g. "gs://fc-e0000000-0000-0000-0000-000000000000/dropseq_output")
reference	hg19, GRCh38, mm10, hg19_mm10, mmul_8.0.1 or a path to a custom reference JSON file
run_bcl2fastq	Whether your sample sheet contains one BCL directory per line or one sample per line (default false)
run_dropseq_tools	Whether to generate count matrixes using Drop-Seq tools from the McCarroll lab (default true)
run_dropest	Whether to generate count matrixes using dropEst (default false)
cellular_barcode_whitelist	Optional whitelist of known cellular barcodes
drop_seq_tools_force_cells	If supplied, bypass the cell detection algorithm (the elbow method) and use this number of cells.
dropest_cells_max	Maximal number of output cells
dropest_genes_min	Minimal number of genes for cells after the merge procedure (default 100)
dropest_min_merge_fraction	Threshold for the merge procedure (default 0.2)
dropest_max_cb_merge_edit_distance	Max edit distance between barcodes (default 2)
dropest_max_umi_merge_edit_distance	Max edit distance between UMIs (default 1)
dropest_min_genes_before_merge	Minimal number of genes for cells before the merge procedure. Used mostly for optimization. (default 10)
dropest_merge_barcodes_precise	Use precise merge strategy (can be slow), recommended to use when the list of real barcodes is not available (default true)
dropest_velocyto	Save separate count matrices for exons, introns and exon/intron spanning reads (default true)
trim_sequence	The sequence to look for at the start of reads for trimming (default "AAGCAGTGGTAT-CAACGCAGAGTGAATGGG")
trim_num_bases	How many bases at the beginning of the sequence must match before trimming occur (default 5)
umi_base_range	The base location of the molecular barcode (default 13-20)
cellular_barcode_base_range	The base location of the cell barcode (default 1-12)
star_flags	Additional options to pass to STAR aligner

Please note that run_bcl2fastq must be set to true if you're starting from BCL files instead of FASTQs.

Custom Genome JSON

If you're reference is not one of the predefined choices, you can create a custom JSON file. Example:

```
{
        "refflat":        "gs://fc-e0000000-0000-0000-0000-000000000000/human_mouse/
↪hg19_mm10_transgenes.refFlat",
        "genome_fasta":    "gs://fc-e0000000-0000-0000-0000-000000000000/human_mouse/
↪hg19_mm10_transgenes.fasta",
        "star_genome":    "gs://fc-e0000000-0000-0000-0000-000000000000/human_mouse/
↪STAR2_5_index_hg19_mm10.tar.gz",
        "gene_intervals":        "gs://fc-e0000000-0000-0000-0000-000000000000/human_
↪mouse/hg19_mm10_transgenes.genes.intervals",
        "genome_dict":    "gs://fc-e0000000-0000-0000-0000-000000000000/human_mouse/
↪hg19_mm10_transgenes.dict",
        "star_cpus": 32,
        "star_memory": "120G"
}
```

The fields star_cpus and star_memory are optional and are used as the default cpus and memory for running STAR with your genome.

Outputs

The pipeline outputs a list of google bucket urls containing one gene-count matrix per sample. Each gene-count matrix file produced by Drop-seq tools has the suffix 'dge.txt.gz', matrices produced by dropEst have the extension .rds.

Building a Custom Genome

The tool **dropseq_bundle** can be used to build a custom genome. Please see the description of important inputs below.

Name	Description
fasta_file	Array of fasta files. If more than one species, fasta and gtf files must be in the same order.
gtf_file	Array of gtf files. If more than one species, fasta and gtf files must be in the same order.
genomeSAindexNbases	Length (bases) of the SA pre-indexing string. Typically between 10 and 15. Longer strings will use much more memory, but allow faster searches. For small genomes, must be scaled down to min(14, log2(GenomeLength)/2 - 1)

dropseq_workflow Terra Release Notes

Version 11

- Added fastq_to_sam_memory and trim_bam_memory workflow inputs

Version 10

- Updated workflow to WDL version 1.0

Version 9

- Changed input bcl2fastq_docker_registry from optional to required

Version 8

- Added additional parameters for bcl2fastq

Version 7

- Added support for multi-species genomes (Barnyard experiments)

Version 6

- Added star_extra_disk_space and star_disk_space_multiplier workflow inputs to adjust disk space allocated for STAR alignment task.

Version 5

- Split preprocessing steps into separate tasks (FastqToSam, TagBam, FilterBam, and TrimBam).

Version 4

- Handle uncompressed fastq files as workflow input.
- Added optional prepare_fastq_disk_space_multiplier input.

Version 3

- Set default value for docker_registry input.

Version 2

- Added docker_registry input.

Version 1

- Renamed sccloud to cumulus
- Added use_bases_mask option when running bcl2fastq

Version 18

- Created a separate docker image for running bcl2fastq

Version 17

- Fixed bug that ignored WDL input star_flags (thanks to Carly Ziegler for reporting)
- Changed default value of star_flags to the empty string (Prior versions of the WDL incorrectly indicated that basic 2-pass mapping was done)

Version 16

- Use cumulus dockerhub organization
- Changed default dropEst version to 0.8.6

Version 15

- Added drop_deq_tools_prep_bam_memory and drop_deq_tools_dge_memory options

Version 14

- Fix for downloading files from user pays buckets

Version 13

- Set GCLOUD_PROJECT_ID for user pays buckets

Version 12

- Changed default dropEst memory from 52G to 104G

Version 11

- Updated formula for computing disk size for dropseq_count

Version 10

- Added option to specify merge_bam_alignment_memory and sort_bam_max_records_in_ram

Version 9

- Updated default drop_seq_tools_version from 2.2.0 to 2.3.0

Version 8

- Made additional options available for running dropEst

Version 7

- Changed default dropEst memory from 104G to 52G

Version 6

- Added option to run dropEst

Version 5

- Specify full version for bcl2fastq (2.20.0.422-2 instead of 2.20.0.422)

Version 4

- Fixed issue that prevented bcl2fastq from running

Version 3

- Set default run_bcl2fastq to false
- Create shortcuts for commonly used genomes

Version 2

- Updated QC report

Version 1

- Initial release

dropseq_bundle Terra Release Notes

Version 4

- Added create_intervals_memory and extra_star_flags inputs

Version 3

- Added extra disk space inputs
- Fixed bug that prevented creating multi-genome bundles

Version 2

- Added docker_registry input

Version 1

- Renamed sccloud to cumulus

Version 1

- Changed docker organization

Version 1

- Initial release

1.1.15 bcl2fastq

License

bcl2fastq license

Workflows

Workflows such as **cellranger_workflow** and **dropseq_workflow** provide the option of running `bcl2fastq`. We provide dockers containing `bcl2fastq` that are accessible only by members of the Broad Institute. Non-Broad Institute members will have to provide their own docker images. Please note that if you're a Broad Institute member and are not able to pull the docker image, please check https://app.terra.bio/#groups to see that you're a member of the all_broad_users group. If not, please contact Terra support and ask to be added to the all_broad_users@firecloud.org group.

Docker

Read this tutorial if you are new to Docker.

Then for a Debian based docker (e.g. continuumio/miniconda3), create the Dockerfile as follows:

```
RUN apt-get update && apt-get install --no-install-recommends -y alien unzip
ADD bcl2fastq2-v2-20-0-linux-x86-64.zip /software/
RUN unzip -d /software/ /software/bcl2fastq2-v2-20-0-linux-x86-64.zip && alien -i /
↪software/bcl2fastq2-v2.20.0.422-Linux-x86_64.rpm && rm /software/bcl2fastq2-v2*
```

Next, download `bcl2fastq` from the Illumina website, which requires registration. Choose the `Linux rpm` file format and download bcl2fastq2-v2-20-0-linux-x86-64.zip to the same directory as your Dockerfile.

You can host your private docker images in the Google Container Registry.

Example

In this example we create a docker image for running `cellranger mkfastq` version 3.0.2.

1. Create a GCP project or reuse an existing project.
2. Enable the Google Container Registry
3. Clone the cumulus repository:

   ```
   git clone https://github.com/lilab-bcb/cumulus.git
   ```

4. Add the lines to cumulus/docker/cellranger/3.0.2/Dockerfile to include bcl2fastq (see *Docker*).
5. Ensure you have Docker installed
6. Download cellranger from https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/3.0
7. Build, tag, and push the docker. Remember to replace PROJECT_ID with your GCP project id:

   ```
   cd cumulus/docker/cellranger/3.0.2/
   docker build -t cellranger-3.0.2 .
   docker tag cellranger-3.0.2 gcr.io/PROJECT_ID/cellranger:3.0.2
   gcr.io/PROJECT_ID/cellranger:3.0.2
   ```

8. Import **cellranger_workflow** workflow to your workspace (see cellranger_workflow steps), and enter your docker registry URL (in this example, `"gcr.io/PROJECT_ID/"`) in `cellranger_mkfastq_docker_registry` field of cellranger_workflow inputs.

1.1.16 Cell Ranger alternatives to generate gene-count matrices for 10X data

This `count` workflow generates gene-count matrices from 10X FASTQ data using alternative methods other than Cell Ranger.

Prepare input data and import workflow

1. Run `cellranger_workflow` to generate FASTQ data

You can skip this step if your data are already in FASTQ format.

Otherwise, you need to first run *cellranger_workflow* to generate FASTQ files from BCL raw data for each sample. Please follow cellranger_workflow manual.

Notice that you should set **run_mkfastq** to `true` to get FASTQ output. You can also set **run_count** to `false` if you want to skip Cell Ranger count, and only use the result from *count* workflow.

For Non-Broad users, you'll need to build your own docker for *bcl2fastq* step. Instructions are here.

2. Import `count`

Import *count* workflow to your workspace.

See the Terra documentation for adding a workflow. The *count* workflow is under `Broad Methods Repository` with name "**cumulus/count**".

Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *count* workflow in the drop-down menu.

3. Prepare a sample sheet

3.1 Sample sheet format:

The sample sheet for *count* workflow should be in TSV format, i.e. columns are seperated by tabs not commas. Please note that the columns in the TSV can be in any order, but that the column names must match the recognized headings.

The sample sheet describes how to identify flowcells and generate channel-specific count matrices.

A brief description of the sample sheet format is listed below **(required column headers are shown in bold)**.

Column	Description
Sample	Contains sample names. Each 10x channel should have a unique sample name.
Flowcells	Indicates the Google bucket URLs of folder(s) holding FASTQ files of this sample.

The sample sheet supports sequencing the same 10x channel across multiple flowcells. If a sample is sequenced across multiple flowcells, simply list all of its flowcells in a comma-seperated way. In the following example, we have 2 samples sequenced in two flowcells.

Example:

```
Sample  Flowcells
sample_1        gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4/
↪sample_1_fastqs,gs://fc-e0000000-0000-0000-0000-000000000000/VK10WBC9Z2/
↪sample_1_fastqs
sample_2        gs://fc-e0000000-0000-0000-0000-000000000000/VK18WBC6Z4/
↪sample_2_fastqs
```

Moreover, if one flowcell of a sample contains multiple FASTQ files for each read, i.e. sequences from multiple lanes, you should keep your sample sheet as the same, and *count* workflow will automatically merge lanes altogether for the sample before performing counting.

3.2 Upload your sample sheet to the workspace bucket:

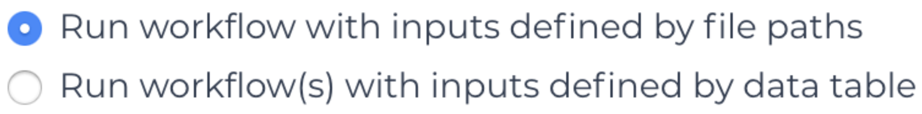
Use gsutil (you already have it if you've installed Google cloud SDK) in your unix terminal to upload your sample sheet to workspace bucket.

Example:

```
gsutil cp /foo/bar/projects/sample_sheet.tsv gs://fc-e0000000-0000-0000-0000-
↪000000000000/
```

4. Launch analysis

In your workspace, open `count` in `WORKFLOWS` tab. Select the desired snapshot version (e.g. latest). Select `Process single workflow from files` as below


Run workflow with inputs defined by file paths
Run workflow(s) with inputs defined by data table

and click `SAVE` button. Select `Use call caching` and click `INPUTS`. Then fill in appropriate values in the `Attribute` column. Alternative, you can upload a JSON file to configure input by clicking `Drag or click to upload json`.

Once INPUTS are appropriated filled, click `RUN ANALYSIS` and then click `LAUNCH`.

Workflow inputs

Below are inputs for *count* workflow. Notice that required inputs are in bold.

Name	Description	Example	Default
input_tsv_file	Input TSV sample sheet describing metadata of each sample.	"gs://fc-e0000000-0000-0000-0000-000000000000/sample_sheet.tsv"	
genome	Genome reference name. Current support: GRCh38, mm10.	"GRCh38"	
chemistry	10X genomics' chemistry name. Current support: "tenX_v3" (for V3 chemistry), "tenX_v2" (for V2 chemistry), "dropseq" (for Drop-Seq).	"tenX_v3"	
output_directory	GS URL of output directory.	"gs://fc-e0000000-0000-0000-0000-000000000000/count_result"	
run_count	If you want to run count tools to generate gene-count matrices.	true	true
count_tool	Count tool to generate result. Options: • "StarSolo": Use STARsolo. • "Optimus": Use Optimus pipeline, developed by the Data Coordination Platform team of the Human Cell Atlas. • "Bustools": Use Kallisto BUSTools. • "Alevin": Use Salmon Alevin.	"StarSolo"	"StarSolo"
docker_registry	Docker registry to use. Notice that docker image for Bustools is seperate. • "quay.io/cumulus" for images on Red Hat registry; • "cumulusprod" for backup images on Docker Hub.	"quay.io/cumulus"	"quay.io/cumulus"
config_version	Version of config docker image to use. This docker is used for parsing the input sample sheet for downstream execution. Available options: `0.2`, `0.1`.	"0.2"	"0.2"
zones	Google cloud zones to consider for execution.	"us-east1-d us-west1-a us-west1-b"	"us-central1-a us-central1-b us-central1-c us-central1-f us-east1-b us-east1-c us-east1-d us-west1-a us-west1-b us-west1-c"
num_cpu	Number of CPUs to request for count per channel. Notice that when use Optimus for count, this input only affects steps of copying files. Optimus uses CPUs due to its own strategy.	32	32
disk_space	Disk space in GB needed for count per channel. Notice that when use Optimus for count, this input only affects steps of copying files. Optimus uses disk space due to its own strategy.	500	500

Workflow outputs

See the table below for *count* workflow outputs.

Name	Type	Description
output_folder	String	Google Bucket URL of output directory. Within it, each folder is for one sample in the input sample sheet.

1.1.17 Topic modeling

Prepare input data

Follow the steps below to run **topic_modeling** on Terra.

1. Prepare your count matrix. **Cumulus** currently supports the following formats: 'zarr', 'h5ad', 'loom', '10x', 'mtx', 'csv', 'tsv' and 'fcs' (for flow/mass cytometry data) formats
2. Upload your count matrix to the workspace.

   Example:

   ```
   gsutil cp /foo/bar/projects/dataset.h5ad gs://fc-e0000000-0000-0000-0000-
   ↪000000000000/
   ```

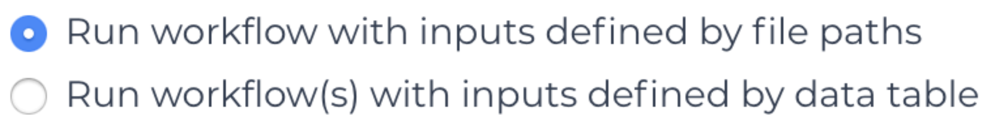
   where `/foo/bar/projects/dataset.h5ad` is the path to your dataset on your local machine, and `gs://fc-e0000000-0000-0000-0000-000000000000/` is the Google bucket destination.

3. Import *topic_modeling* workflow to your workspace.

   See the Terra documentation for adding a workflow. The *cumulus* workflow is under `Broad Methods Repository` with name "**cumulus/topic_modeling**".

   Moreover, in the workflow page, click the `Export to Workspace...` button, and select the workspace to which you want to export *topic_modeling* workflow in the drop-down menu.

4. In your workspace, open `topic_modeling` in `WORKFLOWS` tab. Select `Run workflow with inputs defined by file paths` as below


Run workflow with inputs defined by file paths
Run workflow(s) with inputs defined by data table

   and click the `SAVE` button.

Workflow input

Inputs for the *topic_modeling* workflow are described below. Required inputs are in bold.

Name	Description	Example	Default
input_file	Google bucket URL of the input count matrix.	"gs://fc-e0000000-0000-0000-0000-000000000000/my_dataset.h5ad"	
number_of_topics	Array of number of topics.	[10,15,20]	
prefix_exclude	Comma separated list of features to exclude that start with prefix.	"mt-,Rpl,Rps"	"mt-,Rpl,Rps"
min_percent_expressed	Exclude features expressed below min_percent.	2	
max_percent_expressed	Exclude features expressed below min_percent.	98	
random_number_seed	Random number seed for reproducibility.	0	0

Workflow output

Name	Type	Description
coherence_plot	File	Plot of coherence scores vs. number of topics
perplexity_plot	File	Plot of perplexity values vs. number of topics
cell_scores	Array[File]	Topic by cells (one file for each topic number)
feature_topics	Array[File]	Topic by features (one file for each topic number)
report	Array[File]	HTML visualization report (one file for each topic number)
stats	Array[File]	Computed coherence and perplexity (one file for each topic number)
model	Array[File]	Serialized LDA model (one file for each topic number)
corpus	File	Serialized corpus
dictionary	File	Serialized dictionary

1.1.18 Contributions

We welcome contributions to our repositories that make up the Cumulus ecosystem:

- pegasus
- pegasusio
- demuxEM
- cumulus
- cumulus_feature_barcoding
- cirrocumulus
- altocumulus
- stratocumulus

In addition to the Cumulus team, we would like to sincerely thank the following contributors:

Name	Note
Kirk Gosik	Assistance with topic modeling workflow

1.1.19 Contact us

If you have any questions related to Cumulus, please feel free to contact us via one of the following ways:

- Report new issues on our GitHub repository.
- Send emails to Cumulus Support Google Group.
- Join Cumulus Support Google Group for discussion and release update.