Demuxafy

Release 0.0.3

Drew Neavin

Mar 10, 2022

GENERAL

CHAPTER

ONE

REASON FOR THIS SOFTWARE

Demultiplexing and detecting doublets is an important part of droplet-based scRNA-seq processing pipelines (see additional information and details *below*).

We described some of the best combinations of methods in our manuscript for demultiplexing and doublet detecting (also detailed in the *Software Selection Recommendations*). However, we acknowledge that each dataset is different and may have unique characteristics that make other softwares more suited. Therefore, we have developed Demuxafy to enable each user to choose and run the demultiplexing and doublet detecting analyses of their choice smoothly and efficiently.

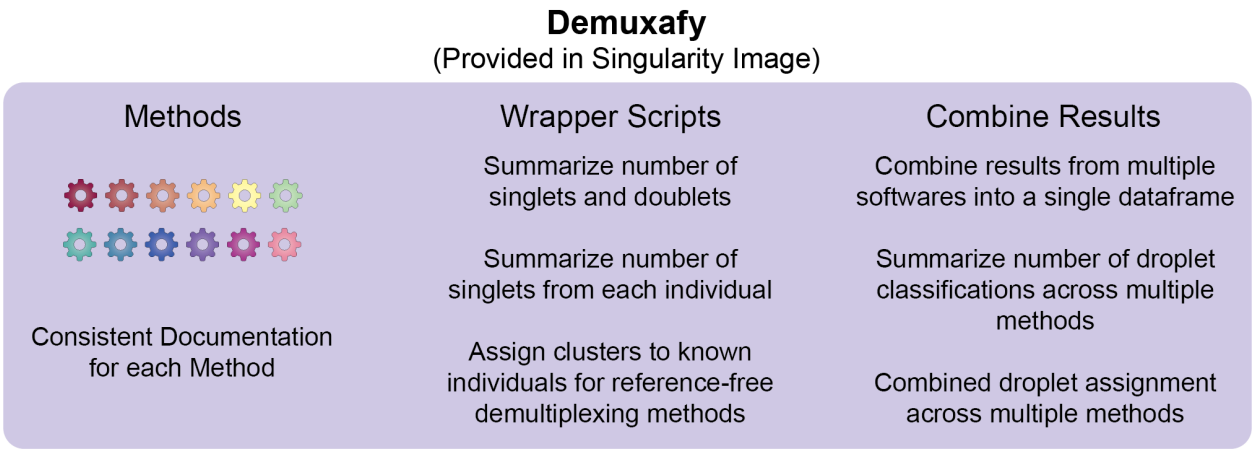


Fig. 1: **Figure 1: Demuxafy** | Demuxafy provides a simple framework for demultiplexing and classifying doublets in a wide range of droplet-based scRNA-seq captures.

Demuxafy contains all of the software you will need for any analysis into a singularity image that can be easily run on most HPCs. This means that you do not need to install each software separately and provides standardization across studies and/or collaborations. We have also built scripts that will easly summarize the results from each software for you - making the assessment of the success of a software faster and easier. Finally, we provide a simple command that will easily combine the results from each of the individual softwares into a common dataframe and provide summary statistics about that combination.

We try our best to include all the possible methods for demultiplexing and doublet detecting in this image and maintain them up-to-date. If you notice a demultiplexing or doublet detecting software for scRNA-seq data that we have not included, please *reach out to us*.

CHAPTER

TWO

DEMULTIPLEXING AND DOUBLET DETECTING SUMMARY

As droplet-based single cell technologies have advanced, increasingly larger sample numbers have been used to answer research questions at single cell resolution. This has been made possible because, as the droplet-based capture technologies have been optimized, methods to pool and then demultiplex samples - assign droplets to each individual in the pool - have been developed (*Figure 2*). These multiplexing methods clearly decrease cost and time of scRNA-seq experiments.

With a larger number of droplets captured, there is an increase in the proportion of the droplets that are doublets (*Figure 3*). If left in the dataset, doublets can significantly impact scientific conclusions such as identifying spurious cell trajectories or false novel cell types. Therefore, it's crucial to effectively clean datasets prior to downstream analyses.

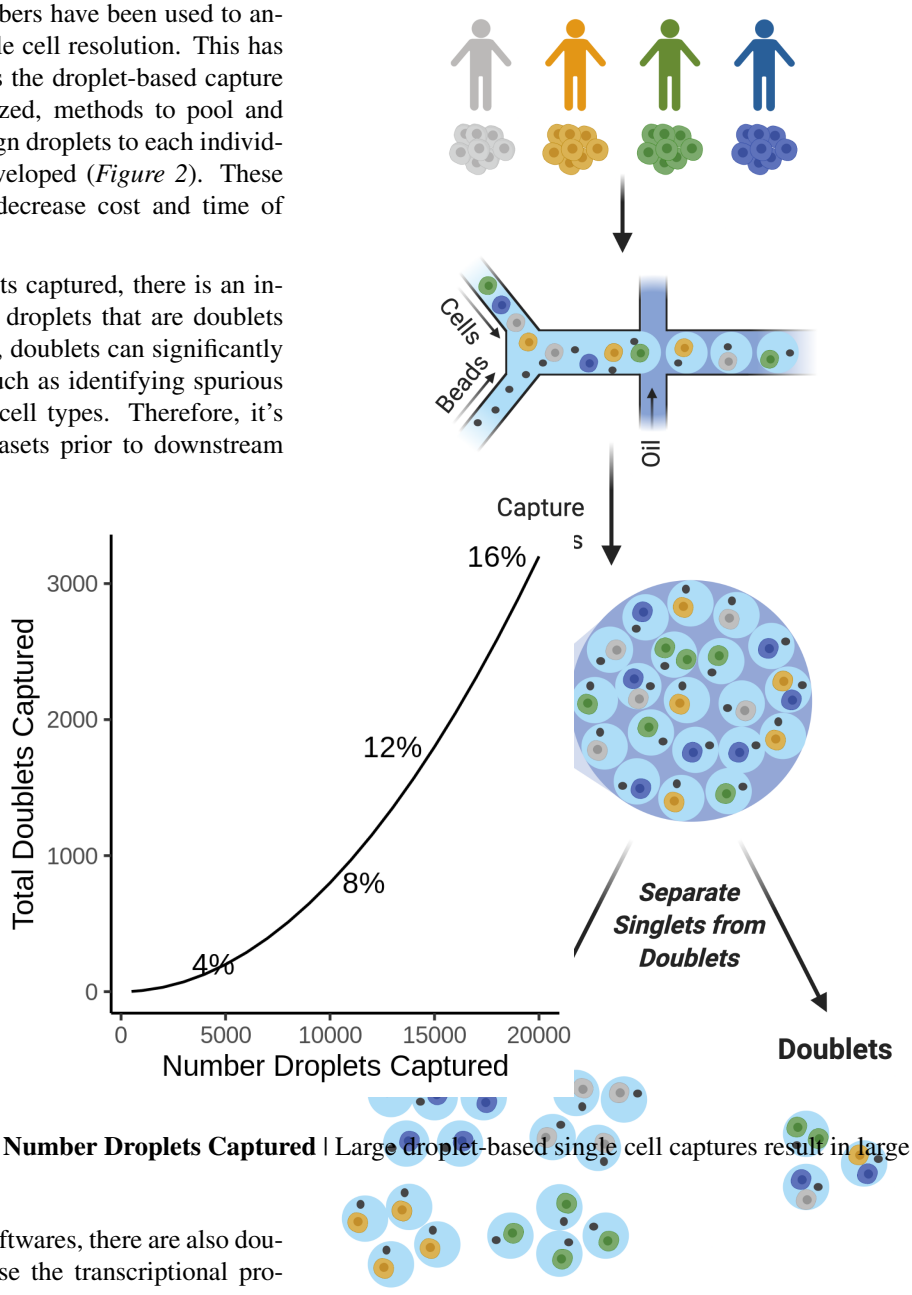


Fig. 2: **Figure 3: Doublets per Number Droplets Captured** | Large droplet-based single cell captures result in large proportions of doublets.

In addition to demultiplexing softwares, there are also doublet detecting softwares that use the transcriptional pro-

Fig. 1: **Figure 2: Multiplexed Single Cell Captures** | Cells from multiple donors can be pooled before capture. The goal post-capture is then to assign the singlets to the

files of droplets to identify doublets by simulating doublets. Both demultiplexing and doublet doublet detecting softwares can be used to identify doublets in a dataset but identify differen types of doublets. Demultiplexing methods can identify doublets from two different individuals whereases transcription-based doublet detecting methods can identify doublets between two different cell types (*Figure 4*). This makes these two method types complementary for demultiplexing and doublet removal.

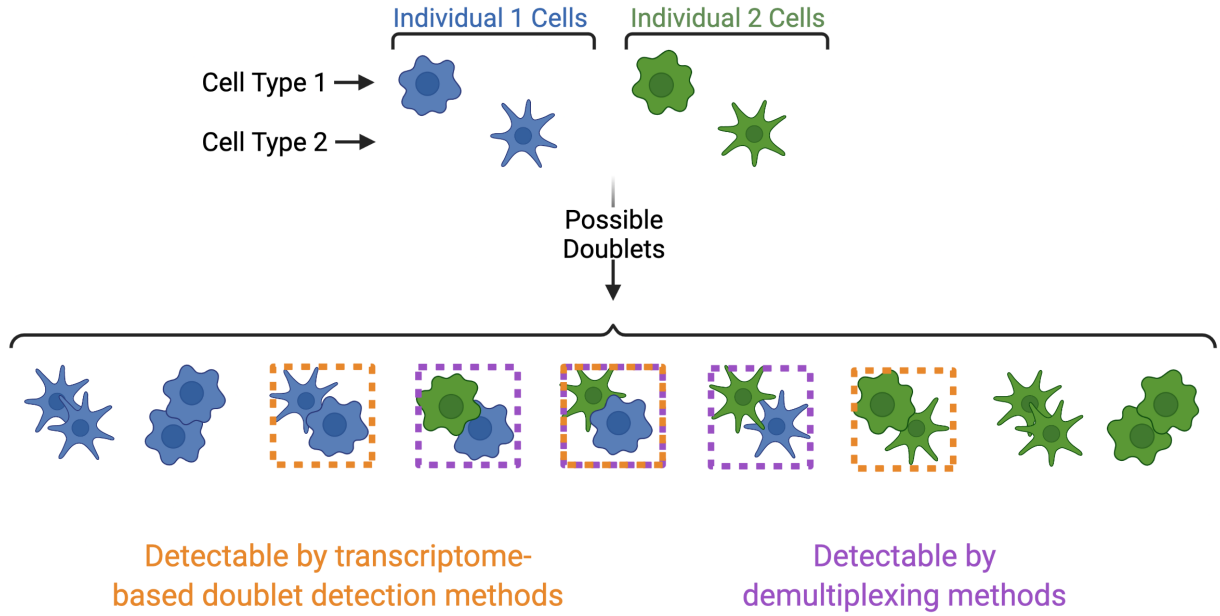


Fig. 3: **Figure 4: Doublets Identified by Demultiplexing and Doublet Detecting Methods** | Doublets can contain cells from different individuals or the same donor and the same or different cell types. Demultiplexing methods can only identify doublets from two different individuals while doublet detecting methods can only identify doublets from two different cell types.

2.1 Background

2.1.1 Reason for this Software

Demultiplexing and detecting doublets is an important part of droplet-based scRNA-seq processing pipelines (see additional information and details *below*).

We described some of the best combinations of methods in our manuscript for demultiplexing and doublet detecting (also detailed in the *Software Selection Recommendations*). However, we acknowledge that each dataset is different and may have unique characteristics that make other softwares more suited. Therefore, we have developed Demuxafy to enable each user to choose and run the demultiplexing and doublet detecting analyses of their choice smoothly

and efficiently.

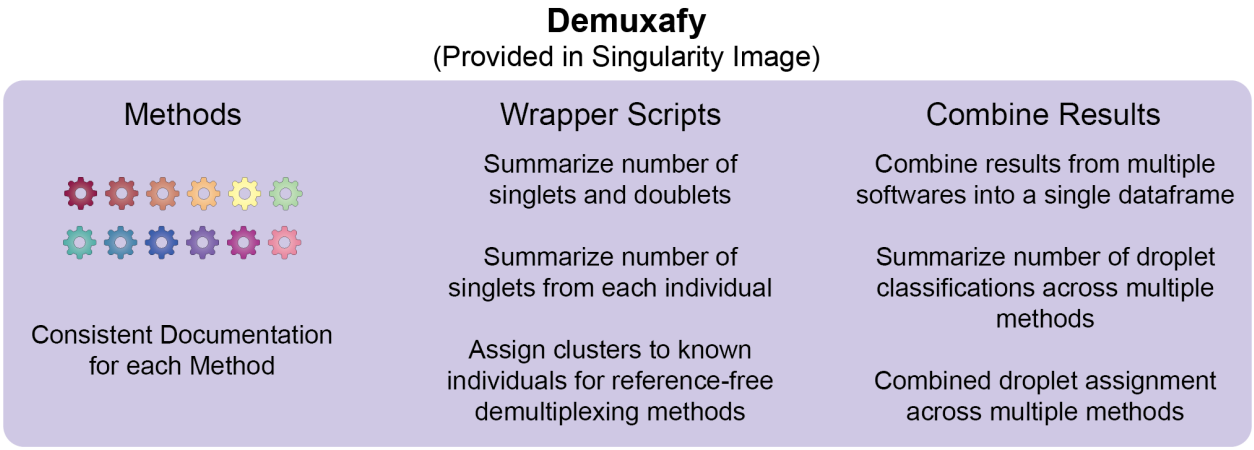


Fig. 4: **Figure 1: Demuxafy** | Demuxafy provides a simple framework for demultiplexing and classifying doublets in a wide range of droplet-based scRNA-seq captures.

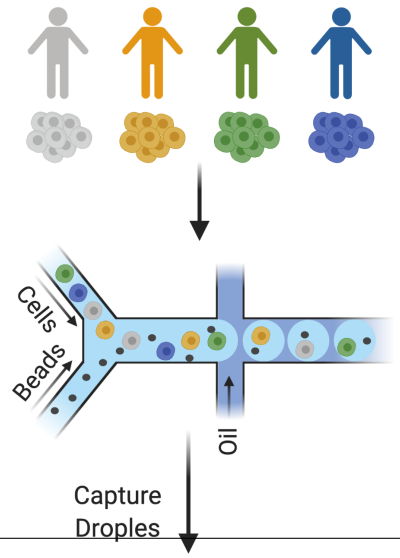
Demuxafy contains all of the software you will need for any analysis into a singularity image that can be easily run on most HPCs. This means that you do not need to install each software separately and provides standardization across studies and/or collaborations. We have also built scripts that will easly summarize the results from each software for you - making the assessment of the success of a software faster and easier. Finally, we provide a simple command that will easily combine the results from each of the individual softwares into a common dataframe and provide summary statistics about that combination.

We try our best to include all the possible methods for demultiplexing and doublet detecting in this image and maintain them up-to-date. If you notice a demultiplexing or doublet detecting software for scRNA-seq data that we have not included, please *reach out to us*.

2.1.2 Demultiplexing and Doublet Detecting Summary

As droplet-based single cell technologies have advanced, increasingly larger sample numbers have been used to answer research questions at single cell resolution. This has been made possible because, as the droplet-based capture technologies have been optimized, methods to pool and then demultiplex samples - assign droplets to each individual in the pool - have been developed (*Figure 2*). These multiplexing methods clearly decrease cost and time of scRNA-seq experiments.

With a larger number of droplets captured, there is an increase in the proportion of the droplets that are doublets (*Figure 3*). If left in the dataset, doublets can significantly impact scientific conclusions such as identifying spurious cell trajectories or false novel cell types. Therefore, it's crucial to effectively clean datasets prior to downstream analyses.



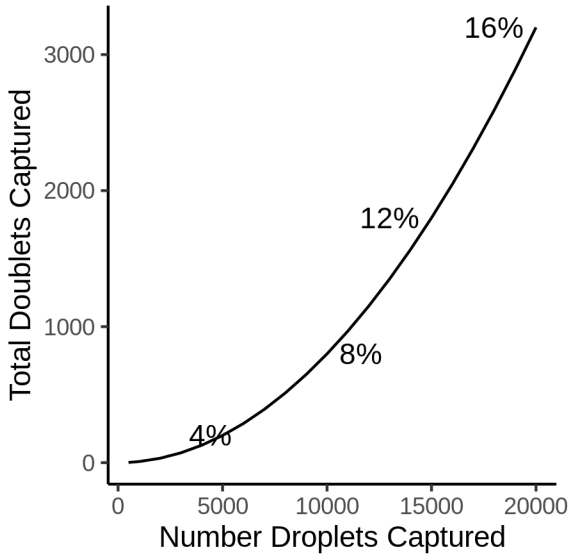


Fig. 6: **Figure 3: Doublets per Number Droplets Captured |** Large droplet-based single cell captures result in large proportions of doublets.

In addition to demultiplexing softwares, there are also doublet detecting softwares that use the transcriptional profiles of droplets to identify doublets by simulating doublets. Both demultiplexing and doublet doublet detecting softwares can be used to identify doublets in a dataset but identify differen types of doublets. Demultiplexing methods can identify doublets from two different individuals whereases transcription-based doublet detecting methods can identify doublets between two different cell types (*Figure 4*). This makes these two method types complementary for demultiplexing and doublet removal.

2.2 Support

If you're having trouble with any part of the Demultiplexing and Doublet Detecting Pipeline, feel free to submit an issue.

2.3 Software Selection Recommendations

Based on our analysis of demultiplexing and doublet detecting softwares, we have generated the following decision tree to help other researchers elect the best set of softwares for their dataset.

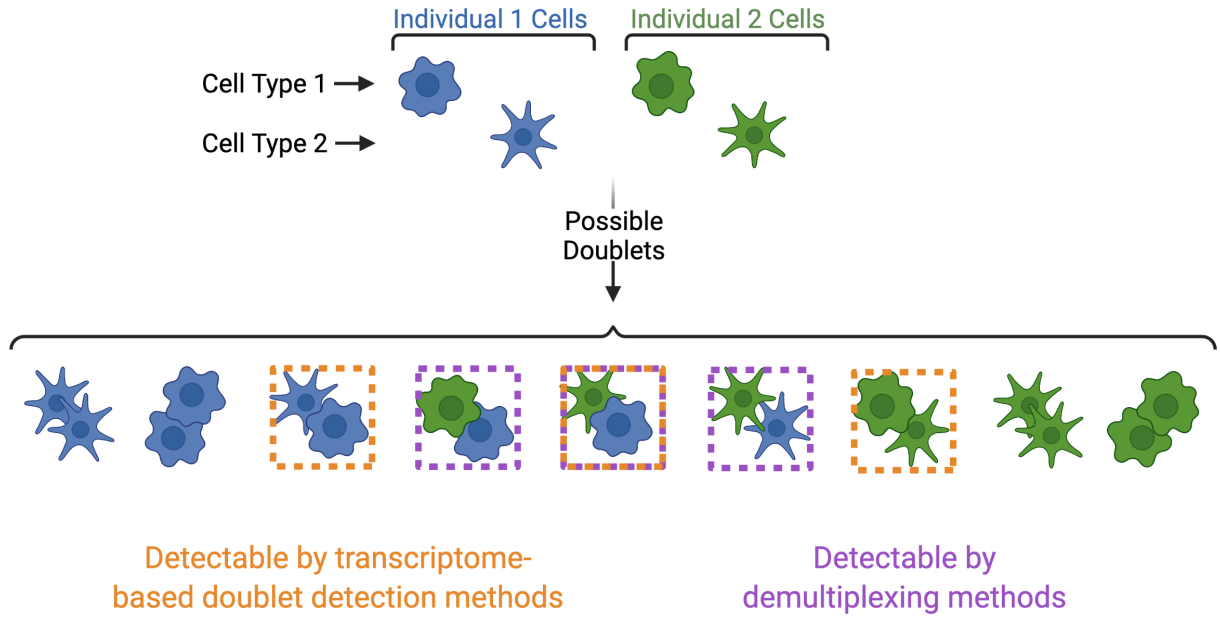
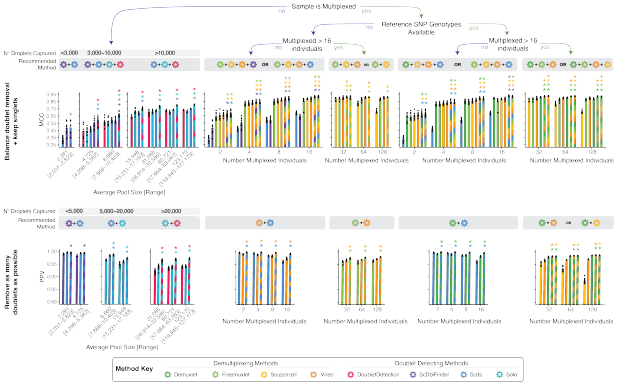


Fig. 7: **Figure 4: Doublets Identified by Demultiplexing and Doublet Detecting Methods** | Doublets can contain cells from different individuals or the same donor and the same or different cell types. Demultiplexing methods can only identify doublets from two different individuals while doublet detecting methods can only identify doublets from two different cell types.



After you have run the softwares you selected, we have provided a script that will help merge and summarize the results from ethe softwares together. See *Combine Results*.

2.4 Installation

Installation should be pretty painless (we hope). We have provided all the softwares in a singularity image which provides continuity across different computing platforms (see HPCNG Singluarity and Sylabs io for more information on singularity images). The only thing to note before you download this image is that the image is **~6.5Gb** so, depending on the internet speed, it will take **~15-30 min to**

download. The good news is that you should only need to do this once unless updates are made to the scripts or image.

Just download the singluarity image with:

```
wget https://www.dropbox.com/s/6p64z451qobi070/Demuxafy.sif
wget https://www.dropbox.com/s/0hpzm04oqbqg4ra/Demuxafy.sif.md5
```

Then you should check to make sure that the image downloaded completely by comparing the image md5sum to the original md5sum. You can do that by running the following commands:

```
md5sum Demuxafy.sif > downloaded_Demuxafy.sif.md5
diff -s Demuxafy.sif.md5 downloaded_Demuxafy.sif.md5
```

If everything was downloaded correctly, that command should report:

```
Files Demuxafy.sif.md5 and downloaded_Demuxafy.sif.md5 are identical
```

Note: Please note that the singularity image and this documentation is updated with each release. This means that the most recent documentation may not be 100% compatible with the singularity image that you have. For example, additional parameters and functionality were implemented in v1.0.2 that was not available in v0.0.4.

You can check the version of your singularity image to match with documentation with:

```
singularity inspect Demuxafy.sif
```

If you run into any issues with downloading the image or any issue with running anything from this image, you can reach out to us by submitting an issue at Github

---

Demuxafy software versions - for the curious

Image build date: 18 February, 2022

Software Group	Software	Version
Demultiplexing	**popscle** • demuxlet • freemuxlet	v0.1-beta
	scSplit	v1.0.8.2
	Souporcell	v2.0
	Vireo	v0.5.6
Doublet Detecting	DoubletDecon	v1.1.6
	DoubletDetection	v3.0
	DoubletFinder	v2.0.3
	scDblFinder	v1.8.0
	scds	v1.9.1
	scrublet	v0.2.3
	solo	v1.2
Supporting Softwares	minimap2	v2.7-r654
	bedtools2	v2.28.0
	vartrix	v1.1.3
	htslib	v1.7 & v1.13 & v1.14

continues on next page

Table 1 – continued from previous page

Software Group	Software	Version
	samtools	v1.7
	bcftools	v1.13
	freebayes	v1.3.5
	cellSNP-lite	v1.2.2
R Supporting Packages (R v4.1.2)	argparse	v2.1.3
	ComplexHeatmap	v2.10.0
	ComplexUpset	v1.3.3
	vcfR	v1.12.0
	Seurat	4.1.0
	SingleCellExperiment	v1.16.0
Python Supporting Packages (Python v3.7.2)	argparse	v1.4.0
	numpy	v1.2.2
	matplotlib	v3.2.2
	pandas	v1.3.5
	PyVCF	v0.6.8
	scipy	v1.7.3
	scvi-tools	v0.14.6
	umap-learn	v0.5.2

2.5 Data Preparation

There isn't a lot of data preparation to be done before running the demultiplexing or doublet detecting softwares.

2.5.1 Data Required

The demultiplexing and transcriptome-based doublet detecting softwares have different data input requirements:

Software Group	Single Cell Count Data Required	SNP Genotype Data Required
Demultiplexing	✓	✓
Doublet Detecting	✓	

Note: The SNP genotype data can be for multiplexed donors in the pool **OR** it can be publicly available common SNP genotypes which can be downloaded from 1000G (hg19 and hg38) or from HRC (hg19 only).

For 1000G, use the instructions at the above link to access the data per your preferences and you can find the required files at the following directories:

- The hg19 data is available at `/ftp/release/`
- The hg38 data is available at `/ftp/release/20130502/supporting/GRCh38_positions/`

You won't need to pre-process the single cell count data unless you are using *DoubletFinder* or *DoubletDecon* which need QC-filtered and normalized counts (for example with Seurat).

For the demultiplexing softwares, you should filter the SNP genotypes that you will use.

2.5.2 SNP Genotype Pre-processing

It is best to filter the SNP genotypes for common SNPs (generally > 1% or > 5% minor allele frequency) that overlap exons. Here we provide an example of how to do this filtering. We built the required softwares into the singularity image so you can run these filtering steps with the image.

Note: We have found it best to impute reference SNP genotypes so there are more SNP locations available. If you are using reference SNP genotypes for the donors in your pool, please be sure to impute before filtering.

Filter for Common SNPs

First, filter the SNP genotypes for common SNPs - 5% minor allele frequency should work for most datasets but you can change this to another minor allele frequency if you would like.

```
singularity exec Demuxafy.sif bcftools filter --include 'MAF>=0.05' -Oz --output
↪$OUTDIR/common_maf0.05.vcf.gz $VCF
```

Where `$OUTDIR` is the output directory where you want to save the results and `$VCF` is the path to the SNP genotype vcf file.

Filter for SNPs overlapping Exons

Next, filter for the SNPs that overlap exons.

Note: You can get an exon bed using the UCSC table browser (see instructions here) and we have also provided bed files for `hg19` and `hg38`

```
singularity exec Demuxafy.sif vcftools \
  --gzvcf $OUTDIR/common_maf0.05.vcf.gz \
  --max-alleles 2 \
  --remove-indels \
  --bed $BED \
  --recode \
  --recode-INFO-all \
  --out $OUTDIR/common_maf0.05_exon_filtered
```

2.5.3 Test Dataset

In addition, we have provided test data that you can use.

Information

The test dataset includes 20,982 droplets captured of PBMCs from 13 multiplexed individuals.

10x Directories + Other Necessary Files

We have provided this dataset as the complete dataset which is pretty large (~40Gb tar.gz directory). Therefore, we have also provided the same dataset where the data has been significantly reduced.

Warning: The reduced test dataset may not produce real-world results due to the small size - especially for doublet detecting softwares since the reads have been significantly downsampled to reduce the size.

You can download the dataset with one of the following commands:

Complete Dataset

Reduced Dataset

First, download the dataset and the md5sum:

```
wget https://www.dropbox.com/s/3oujqq98y400rzz/TestData4PipelineFull.tar.gz
wget https://www.dropbox.com/s/5n7u723okkf5m3l/TestData4PipelineFull.tar.gz.md5
```

After downloading the tar.gz directory, it is best to make sure the md5sum of the `TestData4PipelineFull.tar.gz` file matches the md5sum in the `TestData4PipelineFull.tar.gz.md5`:

```
md5sum TestData4PipelineFull.tar.gz > downloaded_TestData4PipelineFull.tar.gz.md5
diff -s TestData4PipelineFull.tar.gz.md5 downloaded_TestData4PipelineFull.tar.gz.md5
```

That should return the following statement indicating that the two md5sums are identical:

```
Files TestData4PipelineFull.tar.gz.md5 and downloaded_TestData4PipelineFull.tar.gz.
↪md5 are identical
```

First, download the reduced dataset and the md5sum:

```
wget https://www.dropbox.com/s/m8u61jn4i1mcktp/TestData4PipelineSmall.tar.gz
wget https://www.dropbox.com/s/ykjg86q3xw39wqr/TestData4PipelineSmall.tar.gz.md5
```

After downloading the tar.gz directory, it is best to make sure the md5sum of the `TestData4PipelineSmall.tar.gz` file matches the md5sum in the `TestData4PipelineSmall.tar.gz.md5`:

```
md5sum TestData4PipelineSmall.tar.gz > downloaded_TestData4PipelineSmall.tar.gz.md5
diff -s TestData4PipelineSmall.tar.gz.md5 downloaded_TestData4PipelineSmall.tar.gz.md5
```

That should return the following statement indicating that the two md5sums are identical:

```
Files TestData4PipelineSmall.tar.gz.md5 and downloaded_TestData4PipelineSmall.tar.gz.
↪md5 are identical
```

Seurat Object

We have also provided a filtered, QC normalized Seurat object (needed for *DoubletFinder* and *DoubletDecon*)

Download the rds object and the md5sum:

```
wget https://www.dropbox.com/s/po4gy2j3eqohhjv/TestData_Seurat.rds
wget https://www.dropbox.com/s/rmix7tt9aw28n7i/TestData_Seurat.rds.md5
```

After downloading the rds.object, it is best to make sure the md5sum of the `TestData_Seurat.rds` file matches the md5sum in the `TestData_Seurat.rds.md5`:

```
md5sum TestData_Seurat.rds > downloaded_TestData_Seurat.rds.md5
diff -s TestData_Seurat.rds.md5 downloaded_TestData_Seurat.rds.md5
```

That should return the following statement indicating that the two md5sums are identical:

```
Files TestData_Seurat.rds.md5 and downloaded_TestData_Seurat.rds.md5 are identical
```

Note: We have used this dataset for each of the tutorials. The example tables in the *Results and Interpretation* sections of each tutorial are the results from this dataset.

2.6 Considerations for Other Single Cell Data Types

This workflow was designed for demultiplexing and detecting doublets in scRNA-seq data. However, additional data types are becoming more frequently used - *i.e.* snRNA-seq, snATAC-seq and dual snRNA-seq + scATAC-seq. Based on our experiences with this data we have some recommendations and considerations to take into account when applying demultiplexing and doublet detecting softwares to these data types.

2.6.1 snRNA-seq

Demultiplexing Softwares

We have not tested any demultiplexing softwares on snRNA-seq data in our hands but we anticipate that it should behave similarly to scRNA-seq. The only difference we would suggest is to filter SNPs overlapping **genes** instead of just overlapping **exons**. If you are running in to any issues or would like a discussion about use of demultiplexing softwares on snRNA-seq data, please feel free to reach out.

Doublet Detecting Softwares

We have not tested doublet detecting softwares on snRNA-seq data but the softwares should work similarly as they do on scRNA-seq data. If you are running in to any issues or would like a discussion about use of doublet detecting softwares on snRNA-seq data, please feel free to reach out.

2.6.2 snATAC-seq

Demultiplexing Softwares

Demultiplexing snATAC-seq data can be done with the current demultiplexing softwares. However, we note that it is much more memory and time consumptive than scRNA-seq. Additionally, the SNPs should be filtered by SNPs overlapping **peak locations** instead of exon or gene locations. You may even want to filter the SNPs further if you still have many after filtering on minor allele frequency and peak location. We typically aim for ~250,000 SNP. Regardless, since UMI tags aren't used for snATAC-seq data, demultiplexing can take a lot of memory and time.

In addition, the following flags are required for each of the following softwares to effectively process snATAC-seq data.

Souporcell

- `--no_umi True`

Doublet Detecting Softwares

Technically, doublet detecting softwares cannot be applied to snATAC-seq data as they rely on the unique transcriptomes of each cell type to identify heterotypic doublets. However, if snATAC-seq peaks can be made into a scRNA-seq-like matrices (*i.e.* by linking peaks to genes or some other method), the doublet detecting softwares outlined in this workflow could be applied to snATAC-seq data. This has been shown previously by SnapATAC and ArchR has a method built in that uses a very similar method to *Scrublet*. AMULET is another doublet detecting method that has been developed specifically for snATAC-seq data.

2.6.3 Combined snRNA-seq + snATAC-seq

Demultiplexing Softwares

We have noticed a higher percentage of ambient RNA from our combined snRNA-seq + scATAC-seq experiments as compared to our scRNA-seq (we haven't tested multiplexed snRNA-seq in our hands) but similar snATAC-seq ambient DNA estimations as detected with *Souporcell* Therefore, we recommend running *Souporcell*, if only to estimate the ambient RNA in your multiplexed pool. If you are doing the demultiplexing with the snRNA-seq results, please see the *snRNA-seq Section*. If you are using the snATAC-seq data for demultiplexing, please see the *snATAC-seq Section*.

Doublet Detecting Softwares

Doublet detecting softwares for the combined snRNA-seq + snATAC-seq should work similarly to the doublet detecting softwares for each assay separately (snRNA-seq and snATAC-seq). However, as noted int the *Demultiplexing Softwares Section* above, we have observed much higher ambient RNA percentages than for either assay run separately. ..Our results (**CITATION**) indicate that increased ambient RNA showed a slight decrease in the MCC and balanced accuracy. However, we did not simulate up to the level of ambient RNA percent that we have observed using this assay.

2.7 Notes About Singularity Images

Singularity images effectively store an operating system with files, softwares etc. that can be easily transported across different operating systems - ensuring reproducibility. Most HPCs have singularity installed making it easy to implement. There are some tips and tricks we have identified through using singularity images that we thought might help new users.

2.7.1 Tips and Tricks

1. Error: File Not Found

Reason

Singularity only loads the directories directly downstream from where you execute the singularity command. If any of the files that need to be accessed by the command are not downstream of the that location, you will receive an error similar to this one:

```
Failed to open file "/path/to/readfile.tsv" : No such file or directory
```

If you then check for that file:

```
ll /path/to/readfile.tsv
```

We can see that the file does truly exist:

```
-rw-rw-r-- 1 user group 70636291 Dec 21  2020 /path/to/readfile.tsv
```

Solution

The easiest solution to this problem is to "bind" a path upstream of all the files that will need to be accessed by your command:

```
singularity exec --bind /path Demuxafy.sif ...
```

If you don't have access to Singularity on your HPC, you can ask your HPC administrators to install it (see the Singularity page)

2.8 Contact

Demuxafy has been developed by Drew Neavin in Joseph Powell's Lab at the Garvan Institute of Medical Research.

You can contact us with questions, issues or recommendations with a Github issue.

If you use this resource, please cite our publication (**REFERENCE**).

2.9 Overview of Demultiplexing Softwares

Demultiplexing softwares use the inherent genetic differences between donors multiplexed in a single pool to assign droplets to each donor and to identify doublets. There are five demultiplexing softwares that have different capabilities and advantages depending on your dataset. As you can see from this table, only *Demuxlet* absolutely requires reference SNP genotypes for the donors multiplexed in your pool. However, *Souporcell* and *Vireo* are also capable of accomodating reference SNP genotypes as well.

Demultiplexing Software	Requires Reference SNP Genotypes	Can Use Reference SNP Genotypes	Estimates Ambient RNA
Demuxlet	✓	✓	
Femuxlet			
scSplit			
Souporcell		✓	✓
Vireo		✓	

We highly recommend using *Souporcell* if only to estimate the percentage of ambient RNA in your pool. As far as we are aware, this is the only software that leverages SNP genotype data to estimate ambient RNA in multiplexed pools and it is helpful to identify high ambient RNA which is sometimes undetectable with basic QC metrics. We view this as supplementary to other ambient RNA methods that use the transcriptional profile to estimate and remove ambient RNA per droplet.

If you don't know which demultiplexing software(s) to run, take a look at our *Software Selection Recommendations* based on your dataset or use our **add widget link here**

2.10 Demuxlet

Demuxlet is a genotype demultiplexing software that requires reference genotypes to be available for each individual in the pool. Therefore, if you don't have reference genotypes, you may want to demultiplex with one of the softwares that do not require reference genotype data (*Freemuxlet*, *scSplit*, *Souporcell* or *Vireo*)

2.10.1 Data

This is the data that you will need to have prepare to run Demuxlet:

Required

- Reference SNP genotypes for each individual (`$VCF`)
  - Filter for common SNPs (> 5% minor allele frequency) and SNPs overlapping genes
  - Demuxlet is very sensitive to missing data in a vcf so please make sure you only have complete cases in your reference donor SNP genotype file

- Genotype field in `$VCF` (`$FIELD`)
  - This is `GP` by default but could also be `GT` others
- Barcode file (`$BARCODES`)
- Bam file (`$BAM`)
  - Aligned single cell reads
- Output directory (`$DEMUXLET_OUTDIR`)

Optional

- A text file with the individual ids (`$INDS`)
  - File containing the individual ids (separated by line) as they appear in the vcf file
  - For example, this is the `individual file` for our example dataset

2.10.2 Run Demuxlet

Popscle Pileup

First we will need to identify the number of reads from each allele at each SNP location:

With `$INDS` file

Without `$INDS` file

The `$INDS` file allows demuxlet to only consider the individual in this pool

```
singularity exec Demuxafy.sif popscle dsc-pileup --sam $BAM --vcf $VCF --group-list
↪$BARCODES --out $DEMUXLET_OUTDIR/pileup --sm-list $INDS
```

This will use all the individuals in your reference SNP genotype `$VCF`. If your `$VCF` only has the individuals multiplexed in your pool, then the `$INDS` file is not required.

```
singularity exec Demuxafy.sif popscle dsc-pileup --sam $BAM --vcf $VCF --group-list
↪$BARCODES --out $DEMUXLET_OUTDIR/pileup
```

If the pileup is successful, you will have these files in your `$DEMUXLET_OUTDIR`:

```
.
├── pileup.cel.gz
├── pileup.plp.gz
├── pileup.umi.gz
└── pileup.var.gz
```

Additional details about outputs are available below in the *Demuxlet Results and Interpretation*.

Popscle Demuxlet

Once you have run `popscle pileup`, you can demultiplex your samples:

With `$INDS` file

Without `$INDS` file

The `$INDS` file allows demuxlet to only consider the individual in this pool

```
singularity exec Demuxafy.sif popscle demuxlet --plp $DEMUXLET_OUTDIR/pileup --vcf
↪$VCF --field $FIELD --group-list $BARCODES --geno-error-coeff 1.0 --geno-error-
↪offset 0.05 --out $DEMUXLET_OUTDIR/demuxlet --sm-list $INDS
```

This will use all the individuals in your reference SNP genotype `$VCF`. If your `$VCF` only has the individuals multiplexed in your pool, then the `$INDS` file is not required.

```
singularity exec Demuxafy.sif popscle demuxlet --plp $DEMUXLET_OUTDIR/pileup --vcf
↪$VCF --field $FIELD --group-list $BARCODES --geno-error-coeff 1.0 --geno-error-
↪offset 0.05 --out $DEMUXLET_OUTDIR/demuxlet
```

Note

Demuxlet by default assumes that your `$VCF` uses `R2` to indicate the imputation score. If you have a different imputation metric (`INFO` is also commonly used), then you should use `--r2-info` to indicate the metric it should use (for example: `--r2-info INFO`)

If demuxlet is successful, you will have these new files in your `$DEMUXLET_OUTDIR`:

```
.
├── demuxlet.best
├── pileup.cel.gz
├── pileup.plp.gz
├── pileup.umi.gz
└── pileup.var.gz
```

Additional details about outputs are available below in the *Demuxlet Results and Interpretation*.

Demuxlet Summary

We have provided a script that will summarize the number of droplets classified as doublets, ambiguous and assigned to each donor by Demuxlet and write it to the `$DEMUXLET_OUTDIR`. You can run this to get a fast and easy summary of your results by providing the path to your result file:

```
singularity exec Demuxafy.sif bash Demuxlet_summary.sh $DEMUXLET_OUTDIR/demuxlet.best
```

which will return:

Classification	Assignment N
113_113	1334
349_350	1458
352_353	1607
39_39	1297
40_40	1078
41_41	1127
42_42	1419
43_43	1553
465_466	1094
596_597	1255
597_598	1517
632_633	868
633_634	960
660_661	1362
doublet	3053

or you can write it straight to a file:

```
singularity exec Demuxafy.sif bash Demuxlet_summary.sh $DEMUXLET_OUTDIR/demuxlet.best␣
↪> $DEMUXLET_OUTDIR/demuxlet_summary.tsv
```

Note

To check if these numbers are consistent with the expected doublet rate in your dataset, you can use our Doublet Estimation Calculator.

2.10.3 Demuxlet Results and Interpretation

After running the Demuxlet steps and summarizing the results, you will have a number of files from some of the intermediary steps. These are the files that most users will find the most informative:

- `demuxlet.best`
  - Metrics for each droplet including the singlet, doublet or ambiguous assignment (`DROPLET.TYPE`), final assignment (`BEST.GUESS`), log likelihood of the final assignment (`BEST.LLK`) and other QC metrics.

INT_ID	BARCODE	NUM.SNPS	NUM.READS	DROPLET.TYPE	BEST.GUESS	BEST.LLK	NEXT.GUESS	NEXT.LLK	DIFF.LLK.BEST.NEXT	BEST.POSTERIOR	SNG.POSTERIOR	SNG.BEST.GUESS	SNG.BEST.LLK	SNG.NEXT.GUESS	SNG.NEXT.LLK	SNG.ONLY.POSTERIOR	DBL.BEST.GUESS	DBL.BEST.LLK	DIFF.LLK.SNG.DBL
0	AAACCTGAGATAGCAT-1	70	231	SNG	41_41,41_41,0.00	-29.42	40_40,41_41,0.50	-39.12	9.70	[illegible]33	1	41_41	-29.42	597_598	-76.24	0.00	40_40,41_41,0.50	-39.12	9.70
1	[illegible]-1	325	583	SNG	465_466,465_466,0.00	-70.61	42_42,465_466,0.50	-94.85	24.24	[illegible]74	1	465_466	-70.61	42_42	-166.61	0.00	42_42,465_466,0.50	-94.85	24.24
2	[illegible]-1	147	227	SNG	113_113,113_113,0.00	-25.05	39_39,113_113,0.50	-29.85	4.80	[illegible]28	1	113_113	-25.05	349_350	-51.63	0.00	39_39,113_113,0.50	-29.85	4.80
3	[illegible]-1	180	235	SNG	349_350,349_350,0.00	-33.14	349_350,632_633,0.50	-44.78	11.64	[illegible]36	1	349_350	-33.14	632_633	-77.41	0.00	349_350,632_633,0.50	-44.78	11.64
4	[illegible]-1	248	444	SNG	632_633,632_633,0.00	-54.79	352_353,632_633,0.50	-72.23	17.43	[illegible]58	1	632_633	-54.79	633_634	-163.24	0.00	352_353,632_633,0.50	-72.23	17.43
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

2.10.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.10.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as Demuxlet.

2.11 Freemuxlet

Freemuxlet is a genotype-free demultiplexing software that does not require you to have SNP genotypes the donors in your multiplexed capture. In fact, it can't natively integrate SNP genotypes into the demultiplexing. We have provided some scripts that will help identify clusters from given donors if you do have SNP genotypes but use Freemuxlet. However, it might be better to use a software that is designed integrate SNP genotypes while assigning donor/cluster (*Demuxlet*, *Souporcell* or *Vireo*).

2.11.1 Data

This is the data that you will need to have prepare to run Freemuxlet:

Required

- Common SNP genotypes vcf (`$VCF`)
  - While not exactly required, using common SNP genotype locations enhances accuracy
    - If you have reference SNP genotypes for individuals in your pool, you can use those
    - If you do not have reference SNP genotypes, they can be from any large population resource (i.e. 1000 Genomes or HRC)
  - Filter for common SNPs (> 5% minor allele frequency) and SNPs overlapping genes
- Barcode file (`$BARCODES`)
- Number of samples in pool (`$N`)
- Bam file (`$BAM`)
  - Aligned single cell reads
- Output directory (`$FREEMUXLET_OUTDIR`)

2.11.2 Run Freemuxlet

Popscle Pileup

First we will need to identify the number of reads from each allele at each of the common SNP location:

```
singularity exec Demuxafy.sif popscle dsc-pileup --sam $BAM --vcf $VCF --group-list
↪$BARCODES --out $FREEMUXLET_OUTDIR/pileup
```

If the pileup is successfull, you will have these files in your `$FREEMUXLET_OUTDIR`:

```
.
├── pileup.cel.gz
├── pileup.plp.gz
├── pileup.umi.gz
└── pileup.var.gz
```

Additional details about outputs are available below in the *Freemuxlet Results and Interpretation*.

Popscle Freemuxlet

Once you have run `popscle pileup`, you can demultiplex your samples with Freemuxlet:

```
singularity exec Demuxafy.sif popscle freemuxlet --plp $FREEMUXLET_OUTDIR/pileup --
↪out $FREEMUXLET_OUTDIR/freemuxlet --group-list $BARCODES --nsample $N
```

If freemuxlet is successfull, you will have these new files in your `$FREEMUXLET_OUTDIR`:

```
.
├── freemuxlet.clust1.samples.gz
├── freemuxlet.clust1.vcf.gz
├── freemuxlet.lmix
├── pileup.cel.gz
├── pileup.plp.gz
├── pileup.umi.gz
└── pileup.var.gz
```

Additional details about outputs are available below in the *Freemuxlet Results and Interpretation*.

Freemuxlet Summary

We have provided a script that will summarize the number of droplets classified as doublets, ambiguous and assigned to each donor by Freemuxlet and write it to the `$FREEMUXLET_OUTDIR`. You can run this to get a fast and easy summary of your results by providing the result file of interest:

```
singularity exec Demuxafy.sif bash Freemuxlet_summary.sh $FREEMUXLET_OUTDIR/
↪freemuxlet.clust1.samples.gz
```

which will return:

Classification	Assignment N
0	1575
1	1278
10	972
11	1477
12	1630
13	1446
2	1101
3	1150
4	1356
5	1540
6	1110
7	1313
8	1383
9	884
DBL	2767

or you can write it straight to a file:

```
singularity exec Demuxafy.sif bash Freemuxlet_summary.sh $FREEMUXLET_OUTDIR/
↪freemuxlet.clust1.samples.gz > $FREEMUXLET_OUTDIR/freemuxlet_summary.tsv
```

Note

To check if these numbers are consistent with the expected doublet rate in your dataset, you can use our Doublet Estimation Calculator.

Correlating Cluster to Donor Reference SNP Genotypes (optional)

If you have reference SNP genotypes for some or all of the donors in your pool, you can identify which cluster is best correlated with each donor in your reference SNP genotypes. We have provided a script that will do this and provide a heatmap correlation figure and the predicted individual that should be assigned for each cluster. You can either run it with the script by providing the reference SNP genotypes (`$VCF`), the cluster SNP genotypes (`$FREEMUXLET_OUTDIR/freemuxletOUT.clust1.vcf.gz`) and the output directory (`$FREEMUXLET_OUTDIR`) You can run this script with:

Note

In order to do this, your `$VCF` must be reference SNP genotypes for the individuals in the pool and cannot be a general vcf with common SNP genotype locations from 1000 Genomes or HRC.

With Script

Run in R

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -r $VCF -c $FREEMUXLET_OUTDIR/
↪freemuxlet.clust1.vcf.gz -o $FREEMUXLET_OUTDIR
```

To see the parameter help menu, type:

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -h
```

Which will print:

```
usage: Assign_Indiv_by_Geno.R [-h] -r REFERENCE_VCF -c CLUSTER_VCF -o OUTDIR

optional arguments:
-h, --help            show this help message and exit
-r REFERENCE_VCF, --reference_vcf REFERENCE_VCF
                                                    The output directory where results␣
↪will be saved
-c CLUSTER_VCF, --cluster_vcf CLUSTER_VCF
                                                    A QC, normalized seurat object with
                                                    classifications/clusters as Idents().
-o OUTDIR, --outdir OUTDIR
                                                    Number of genes to use in
                                                    'Improved_Seurat_Pre_Process'␣
↪function.
```

You can run the reference vs cluster genotypes manually (possibly because your data doesn't have GT, DS or GP genotype formats) or because you would prefer to alter some of the steps. To run the correlations manually, simply start R from the singularity image:

```
singularity exec Demuxafy.sif R
```

Once, R has started, you can load the required libraries (included in the singularity image) and run the code.

```
.libPaths("/usr/local/lib/R/site-library") ### Required so that libraries are loaded␣
↪from the image instead of locally
library(tidyr)
library(tidyverse)
library(dplyr)
library(vcfR)
library(lsa)
library(ComplexHeatmap)


########## Set up paths and variables ##########

reference_vcf <- "/path/to/reference.vcf"
cluster_vcf <- "/path/to/freemuxlet/out/freemuxletOUT.clust1.vcf.gz"
outdir <- "/path/to/freemuxlet/out/"


########## Set up functions ##########
##### Calculate DS from GP if genotypes in that format #####
calculate_DS <- function(GP_df){
    columns <- c()
    for (i in 1:ncol(GP_df)){
        columns <- c(columns, paste0(colnames(GP_df)[i],"-0"), paste0(colnames(GP_
↪df)[i],"-1"), paste0(colnames(GP_df)[i],"-2"))
    }
    df <- GP_df
    colnames(df) <- paste0("c", colnames(df))
    colnames_orig <- colnames(df)
    for (i in 1:length(colnames_orig)){
        df <- separate(df, sep = ",", col = colnames_orig[i], into = columns[(1+(3*(i-
↪1))):(3+(3*(i-1)))])
```

(continues on next page)

(continued from previous page)

```
    }
    df <- mutate_all(df, function(x) as.numeric(as.character(x)))
    for (i in 1: ncol(GP_df)){
        GP_df[,i] <- df[,(2+((i-1)*3))] + 2* df[,(3+((i-1)*3))]
    }
    return(GP_df)
}

pearson_correlation <- function(df, ref_df, clust_df){
    for (col in colnames(df)){
        for (row in rownames(df)){
            df[row,col] <- cor(as.numeric(pull(ref_df, col)), as.numeric(pull(clust_
↪df, row)), method = "pearson", use = "complete.obs")
        }
    }
    return(df)
}


########## Read in vcf files for each of three non-reference genotype softwares ######
↪####
ref_geno <- read.vcfR(reference_vcf)
cluster_geno <- read.vcfR(cluster_vcf)



########## Convert to tidy data frame ##########
####### Identify which genotype FORMAT to use #######
##### Cluster VCF #####
### Check for each of the different genotype formats ##
## DS ##
format_clust=NA
cluster_geno_tidy <- as_tibble(extract.gt(element = "DS",cluster_geno, IDtoRowNames =␣
↪F))
if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
 ␣
↪message("Found DS genotype format in cluster vcf. Will use that metric for cluster correlation.")
  format_clust = "DS"
}

## GT ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GT",cluster_geno, IDtoRowNames␣
↪= F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
   ␣
↪message("Found GT genotype format in cluster vcf. Will use that metric for cluster correlation.")
    format_clust = "GT"

    if (any(grepl("\\|",cluster_geno_tidy[1,]))){
      separator = "|"
      message("Detected | separator for GT genotype format in cluster vcf")
    } else if (any(grepl("/",cluster_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in cluster vcf")
    } else {
      format_clust = NA
```

(continues on next page)

(continued from previous page)

```
↪message("Can't identify a separator for the GT field in cluster vcf, moving on to using GP.")
    }

    cluster_geno_tidy <- as_tibble(lapply(cluster_geno_tidy, function(x)
↪{gsub(paste0("0",separator,"0"),0, x)}) %>%
                                   lapply(., function(x) {gsub(paste0("0",separator,"1"),1,
↪x)}) %>%
                                   lapply(., function(x) {gsub(paste0("1",separator,"0"),1,
↪x)}) %>%
                                   lapply(., function(x) {gsub(paste0("1",separator,"1"),2,
↪x)}))

  }
}

## GP ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GP",cluster_geno, IDtoRowNames
↪=F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
    format_clust = "GP"
    cluster_geno_tidy <- calculate_DS(cluster_geno_tidy)

↪message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")

  } else {

↪print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. P
    q()
  }
}




### Reference VCF ###
### Check for each of the different genotype formats ##
## DS ##
format_ref = NA
ref_geno_tidy <- as_tibble(extract.gt(element = "DS",ref_geno, IDtoRowNames = F))
if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){

↪message("Found DS genotype format in reference vcf. Will use that metric for cluster correlation.")
  format_ref = "DS"
}

## GT ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GT",ref_geno, IDtoRowNames = F))
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){

↪message("Found GT genotype format in reference vcf. Will use that metric for cluster correlation.")
    format_ref = "GT"

    if (any(grepl("\\|",ref_geno_tidy[1,]))){
```

(continues on next page)

(continued from previous page)

```
      separator = "|"
      message("Detected | separator for GT genotype format in reference vcf")
    } else if (any(grepl("/",ref_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in reference vcf")
    } else {
      format_ref = NA
      message("Can't identify a separator for the GT field in reference vcf, moving on to using GP.")
    }

    ref_geno_tidy <- as_tibble(lapply(ref_geno_tidy, function(x) {gsub(paste0("0",separator,"0"),0, x)}) %>%
                               lapply(., function(x) {gsub(paste0("0",separator,"1"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"0"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"1"),2, x)}))

  }
}

## GP ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GP",ref_geno, IDtoRowNames = F))
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){
    format_clust = "GP"
    ref_geno_tidy <- calculate_DS(ref_geno_tidy)
    message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")

  } else {
    print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. E
    q()
  }
}



### Get SNP IDs that will match between reference and cluster ###
## Account for possibility that the ref or alt might be missing
if ((all(is.na(cluster_geno@fix[,'REF'])) & all(is.na(cluster_geno@fix[,'ALT']))) |(all(is.na(ref_geno@fix[,'REF'])) & all(is.na(ref_geno@fix[,'ALT'])))){
  message("The REF and ALT categories are not provided for the reference and/or the cluster vcf. Will
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,'POS'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'])
} else if (all(is.na(cluster_geno@fix[,'REF'])) | all(is.na(ref_geno@fix[,'REF']))){
  message("The REF categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,'POS'],"_", cluster_geno@fix[,'REF'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",ref_geno@fix[,'REF'])
```

(continues on next page)

(continued from previous page)

```
} else if (all(is.na(cluster_geno@fix[,'ALT'])) | all(is.na(ref_geno@fix[,'ALT']))){
 ␣
↪message("The ALT categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
↪'POS'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",␣
↪ref_geno@fix[,'ALT'])
} else {
 ␣
↪message("Found REF and ALT in both cluster and reference genotype vcfs. Will use chromosome, positi
    cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
↪'POS'],"_", cluster_geno@fix[,'REF'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",␣
↪ref_geno@fix[,'REF'],"_", ref_geno@fix[,'ALT'])
}


### Update the vcf dfs to remove SNPs with no genotyopes
cluster_geno_tidy <- cluster_geno_tidy[colSums(!is.na(cluster_geno_tidy)) > 0]
ref_geno_tidy <- ref_geno_tidy[colSums(!is.na(ref_geno_tidy)) > 0]



########## Get a unique list of SNPs that is in both the reference and cluster␣
↪genotypes ##########
locations  <- inner_join(ref_geno_tidy[,"ID"],cluster_geno_tidy[,"ID"])
locations <- locations[!(locations$ID %in% locations[duplicated(locations),]$ID),]

########## Keep just the SNPs that overlap ##########
ref_geno_tidy <- left_join(locations, ref_geno_tidy)
cluster_geno_tidy <- left_join(locations, cluster_geno_tidy)

########## Correlate all the cluster genotypes with the individuals genotyped ########
↪##
##### Make a dataframe that has the clusters as the row names and the individuals as␣
↪the column names #####
pearson_correlations <- as.data.frame(matrix(nrow = (ncol(cluster_geno_tidy) -1),␣
↪ncol = (ncol(ref_geno_tidy) -1)))
colnames(pearson_correlations) <- colnames(ref_geno_tidy)[2:(ncol(ref_geno_tidy))]
rownames(pearson_correlations) <- colnames(cluster_geno_tidy)[2:(ncol(cluster_geno_
↪tidy))]
pearson_correlations <- pearson_correlation(pearson_correlations, ref_geno_tidy,␣
↪cluster_geno_tidy)
cluster <- data.frame("Cluster" = rownames(pearson_correlations))
pearson_correlations_out <- cbind(cluster, pearson_correlations)

########## Save the correlation dataframes ##########
write_delim(pearson_correlations_out, file = paste0(outdir,
↪"/ref_clust_pearson_correlations.tsv"), delim = "\t" )


########## Create correlation figures ##########
col_fun = colorRampPalette(c("white", "red"))(101)
pPearsonCorrelations <- Heatmap(as.matrix(pearson_correlations), cluster_rows = T,␣
↪col = col_fun)

########## Save the correlation figures ##########
```

(continues on next page)

(continued from previous page)

```
png(filename = paste0(outdir,"/ref_clust_pearson_correlation.png"), width = 500)
print(pPearsonCorrelations)
dev.off()

########## Assign individual to cluster based on highest correlating individual ######
↪####
key <- as.data.frame(matrix(nrow = ncol(pearson_correlations), ncol = 3))
colnames(key) <- c("Genotype_ID","Cluster_ID","Correlation")
key$Genotype_ID <- colnames(pearson_correlations)
for (id in key$Genotype_ID){
    if (max(pearson_correlations[,id]) == max(pearson_correlations[rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])],])){
        key$Cluster_ID[which(key$Genotype_ID == id)] <- rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])]
        key$Correlation[which(key$Genotype_ID == id)] <- max(pearson_correlations[,
↪id])
    } else {
        key$Cluster_ID[which(key$Genotype_ID == id)] <- "unassigned"
        key$Correlation[which(key$Genotype_ID == id)] <- NA
    }
}

write_delim(key, file = paste0(outdir,"/Genotype_ID_key.txt"), delim = "\t")
```

After correlating the reference SNP genotypes with the cluster SNP genotypes using either the script or manually, you should have three new files in your `$FREEMUXLET_OUTDIR`:

```
.
├── freemuxlet.clust1.samples.gz
├── freemuxlet.clust1.vcf.gz
├── freemuxlet.lmix
├── freemuxlet_summary.tsv
├── Genotype_ID_key.txt
├── pileup.cel.gz
├── pileup.plp.gz
├── pileup.umi.gz
├── pileup.var.gz
├── ref_clust_pearson_correlation.png
└── ref_clust_pearson_correlations.tsv
```

2.11.3 Freemuxlet Results and Interpretation

After running the Freemuxlet steps and summarizing the results, you will have a number of files from some of the intermediary steps. Theses are the files that most users will find the most informative:

- `freemuxlet.clust1.samples.gz`
  - Metrics for each droplet including the singlet, doublet or ambiguous assignment (`DROPLET.TYPE`), final assignment (`BEST.GUESS`), log likelihood of the final assignment (`BEST.LLK`) and other QC metrics.

INT_ID	BARCODE	NUM.SNPS	NUM.READS	DROPLET.TYPE	BEST.GUESS	BEST.LLK	NEXT.GUESS	NEXT.LLK	DIFF.LLK.BEST.NEXT	BEST.POSTERIOR	SNG.POSTERIOR	SNG.BEST.GUESS	SNG.BEST.LLK	SNG.NEXT.GUESS	SNG.NEXT.LLK	SNG.ONLY.POSTERIOR	DBL.BEST.GUESS	DBL.BEST.LLK	DIFF.LLK.SNG.DBL
0	GTG[illegible]TT-1	600	1500	DBL	12,1	-1001.09	12,4	-1030.21	29.13	-0.00000	6.7e-16	12	-1037.90	1	-1135.80	1.00000	12,1	-1001.09	-36.81
1	CGA[illegible]CC-1	354	678	SNG	7,7	-560.30	13,7	-583.64	23.35	-0.00000	1	7	-560.30	13	-650.83	1.00000	13,7	-583.64	23.35
2	CGCT[illegible]CG-1	1029	2847	DBL	9,3	-1651.22	9,6	-1777.52	126.30	[illegible]	1.5e-65	9	-1802.35	3	-1838.25	1.00000	9,3	-1651.22	-151.13
3	CAG[illegible]TC-1	167	219	SNG	5,5	-261.97	6,5	-272.51	10.54	-0.00001	1	5	-261.97	6	-303.97	1.00000	6,5	-272.51	10.54
4	CGT[illegible]AT-1	287	465	SNG	1,1	-451.79	4,1	-479.98	28.18	-0.00000	1	1	-451.79	10	-562.57	1.00000	4,1	-479.98	28.18
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

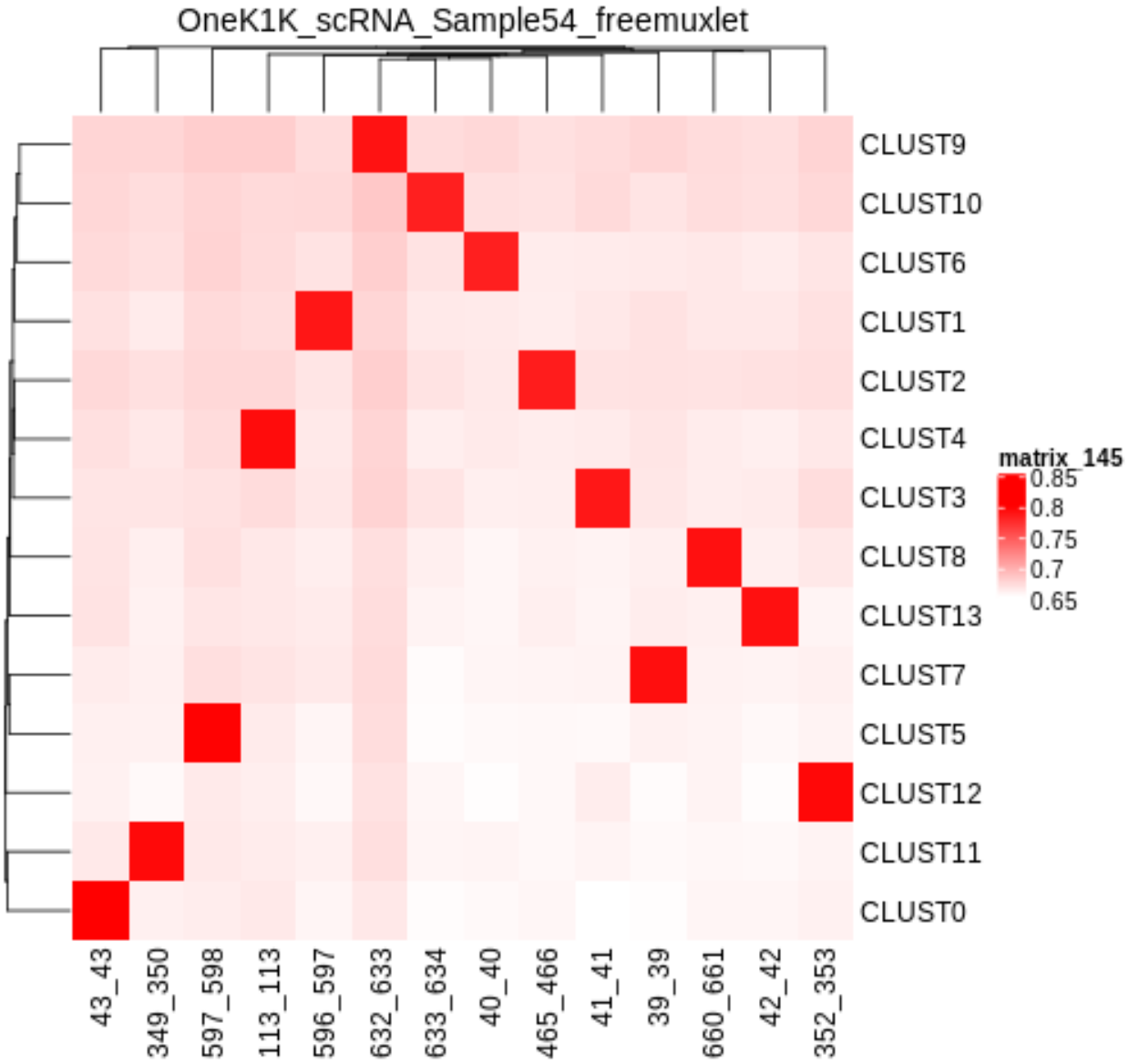
If you ran the `Assign_Indiv_by_Geno.R` script, you will also have the following files:

- `Genotype_ID_key.txt`
  - Key of the cluster and assignments for each individual and the pearson correlation coefficient.

Genotype_ID	Cluster_ID	Correlation
113_113	CLUST4	0.7939599
349_350	CLUST11	0.7954687
352_353	CLUST12	0.7962697
39_39	CLUST7	0.7927807
40_40	CLUST6	0.7833879
41_41	CLUST3	0.7877763
42_42	CLUST13	0.7915233
43_43	CLUST0	0.8008066
465_466	CLUST2	0.7849719
596_597	CLUST1	0.7883125
597_598	CLUST5	0.7996224
632_633	CLUST9	0.7904012
633_634	CLUST10	0.7834359
660_661	CLUST8	0.7914850

- `ref_clust_pearson_correlation.png`
  - Figure of the pearson correlation coefficients for each cluster-individual pair.
- `ref_clust_pearson_correlations.tsv`
  - All of the pearson correlation coefficients between the clusters and the individuals

Cluster	113_113	349_350	352_353	39_39	40_40	…
0	0.6710138155015[illegible]	[illegible]670772417845[illegible]	[illegible]662437546886[illegible]	[illegible]597059348730[illegible]	[illegible]661561196478[illegible]	…
1	0.676832450411[illegible]	[illegible]69804124522[illegible]	[illegible]75336579483[illegible]	[illegible]746102593436[illegible]	[illegible]702202327135[illegible]	…
2	0.680371000427	0.67566064136290[illegible]	[illegible]76486932988[illegible]	[illegible]742600575280[illegible]	[illegible]712474637813[illegible]	…
3	0.6782452606023[illegible]	[illegible]672901336787[illegible]	[illegible]773636626488[illegible]	[illegible]721979348026[illegible]	[illegible]667276727783[illegible]	…
4	0.793959860486[illegible]	[illegible]714745697877[illegible]	[illegible]713909926031[illegible]	[illegible]973064058187[illegible]	[illegible]670269016929[illegible]	…
…	…	…	…	…	…	…


OneK1K_scRNA_Sample54_freemuxlet
CLUST9
CLUST10
CLUST6
CLUST1
CLUST2
CLUST4
CLUST3
CLUST8
CLUST13
CLUST7
CLUST5
CLUST12
CLUST11
CLUST0
43_43
349_350
597_598
113_113
596_597
632_633
633_634
40_40
465_466
41_41
39_39
660_661
42_42
352_353
matrix_145
0.85
0.8
0.75
0.7
0.65

2.11.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.11.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as Freemuxlet.

2.12 ScSplit

ScSplit is a reference-free demultiplexing software. If you have reference SNP genotypes, it would be better to use a demultiplexing software that can handle reference SNP genotypes (*Demuxlet*, *Souporcell* or *Vireo*)

2.12.1 Data

This is the data that you will need to have prepared to run ScSplit:

Required

- Bam file (`$BAM`)
  - Aligned single cell reads
- Genome reference fasta file (`$FASTA`)
- Barcode file (`$BARCODES`)
- Common SNP genotypes vcf (`$VCF`)
  - While not exactly required, using common SNP genotype locations enhances accuracy
    - If you have reference SNP genotypes for individuals in your pool, you can use those
    - If you do not have reference SNP genotypes, they can be from any large population resource (i.e. 1000 Genomes or HRC)
  - Filter for common SNPs (> 5% minor allele frequency) and SNPs overlapping genes
- Number of samples in pool (`$N`)
- Output directory (`$SCSPLIT_OUTDIR`)

2.12.2 Run ScSplit

Prepare Bam file

First, you will need to prepare the bam file so that it only contains high quality, primarily mapped reads without any PCR duplicated reads.

```
singularity exec Demuxafy.sif samtools view -b -S -q 10 -F 3844 $BAM > $SCSPLIT_
↪OUTDIR/filtered_bam.bam
singularity exec Demuxafy.sif samtools rmdup $SCSPLIT_OUTDIR/filtered_bam.bam
↪$SCSPLIT_OUTDIR/filtered_bam_dedup.bam
singularity exec Demuxafy.sif samtools sort -o $SCSPLIT_OUTDIR/filtered_bam_dedup_
↪sorted.bam $SCSPLIT_OUTDIR/filtered_bam_dedup.bam
singularity exec Demuxafy.sif samtools index $SCSPLIT_OUTDIR/filtered_bam_dedup_
↪sorted.bam
```

After running these bam preparation steps, you will have the following files in your `$SCSPLIT_OUTDIR`:

```
.
├── filtered_bam.bam
├── filtered_bam_dedup.bam
├── filtered_bam_dedup_sorted.bam
└── filtered_bam_dedup_sorted.bam.bai
```

Call Sample SNVs

Next, you will need to identify SNV genotypes in the pooled bam.

```
singularity exec Demuxafy.sif freebayes -f $FASTA -iXu -C 2 -q 1 $SCSPLIT_OUTDIR/
↪filtered_bam_dedup_sorted.bam > $SCSPLIT_OUTDIR/freebayes_var.vcf
singularity exec Demuxafy.sif vcftools --gzvcf $SCSPLIT_OUTDIR/freebayes_var.vcf --
↪minQ 30 --recode --recode-INFO-all --out $SCSPLIT_OUTDIR/freebayes_var_qual30
```

After running these SNV calling steps, you will have the following new files in your `$SCSPLIT_OUTDIR`:

```
.
├── filtered_bam.bam
├── filtered_bam_dedup.bam
├── filtered_bam_dedup_sorted.bam
├── filtered_bam_dedup_sorted.bam.bai
├── freebayes_var_qual30.log
├── freebayes_var_qual30.recode.vcf
└── freebayes_var.vcf
```

Demultiplex with ScSplit

The prepared SNV genotypes and bam file can then be used to demultiplex and call genotypes in each cluster.

```
singularity exec Demuxafy.sif scSplit count -c $VCF -v $SCSPLIT_OUTDIR/freebayes_var_
↪qual30.recode.vcf -i $SCSPLIT_OUTDIR/filtered_bam_dedup_sorted.bam -b $BARCODES -r
↪$SCSPLIT_OUTDIR/ref_filtered.csv -a $SCSPLIT_OUTDIR/alt_filtered.csv -o $SCSPLIT_
↪OUTDIR
singularity exec Demuxafy.sif scSplit run -r $SCSPLIT_OUTDIR/ref_filtered.csv -a
↪$SCSPLIT_OUTDIR/alt_filtered.csv -n $N -o $SCSPLIT_OUTDIR
singularity exec Demuxafy.sif scSplit genotype -r $SCSPLIT_OUTDIR/ref_filtered.csv -a
↪$SCSPLIT_OUTDIR/alt_filtered.csv -p $SCSPLIT_OUTDIR/scSplit_P_s_c.csv -o $SCSPLIT_
↪OUTDIR
```

After running these demultiplexing steps, you will have the following new results:

```
.
├── alt_filtered.csv
├── filtered_bam.bam
├── filtered_bam_dedup.bam
├── filtered_bam_dedup_sorted.bam
├── filtered_bam_dedup_sorted.bam.bai
├── freebayes_var_qual30.log
├── freebayes_var_qual30.recode.vcf
├── freebayes_var.vcf
├── ref_filtered.csv
├── scSplit_dist_matrix.csv
├── scSplit_dist_variants.txt
├── scSplit.log
├── scSplit_PA_matrix.csv
├── scSplit_P_s_c.csv
├── scSplit_result.csv
└── scSplit.vcf
```

Additional details about outputs are available below in the *Demuxlet Results and Interpretation*.

ScSplit Summary

We have provided a script that will provide a summary of the number of droplets classified as doublets, ambiguous and assigned to each cluster by ScSplit. You can run this to get a fast and easy summary of your results. Just pass the ScSplit result file:

```
singularity exec Demuxafy.sif bash scSplit_summary.sh $SCSPLIT_OUTDIR/scSplit_result.
↪csv
```

which will return the following summary:

Classification	Assignment N
DBL	1055
SNG-0	1116
SNG-10	1654
SNG-11	1207
SNG-12	1564
SNG-13	1428
SNG-14	1640
SNG-2	514
SNG-3	1314
SNG-4	1587
SNG-5	1774
SNG-6	1484
SNG-7	1662
SNG-8	1578
SNG-9	1282

You can save the summary to file pointing it to the desired output file:

```
singularity exec Demuxafy.sif bash scSplit_summary.sh $SCSPLIT_OUTDIR/scSplit_result.
↪csv > $SCSPLIT_OUTDIR/scSplit_summary.tsv
```

Note

To check if these numbers are consistent with the expected doublet rate in your dataset, you can use our Doublet Estimation Calculator.

Correlating Cluster to Donor Reference SNP Genotypes (optional)

If you have reference SNP genotypes for some or all of the donors in your pool, you can identify which cluster is best correlated with each donor in your reference SNP genotypes. We have provided a script that will do this and provide a heatmap correlation figure and the predicted individual that should be assigned for each cluster. You can either run it with the script by providing the reference SNP genotypes (`$VCF`), the cluster SNP genotypes (`$SCSPLIT_OUTDIR/scSplit.vcf`) and the output directory (`$SCSPLIT_OUTDIR`) You can run this script with:

Note

In order to do this, your $VCF must be reference SNP genotypes for the individuals in the pool and cannot be a general vcf with common SNP genotype locations from 1000 Genomes or HRC.

With Script

Run in R

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -r $VCF -c $SCSPLIT_OUTDIR/
↪scSplit.vcf -o $SCSPLIT_OUTDIR
```

To see the parameter help menu, type:

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -h
```

Which will print:

```
usage: Assign_Indiv_by_Geno.R [-h] -r REFERENCE_VCF -c CLUSTER_VCF -o OUTDIR

optional arguments:
-h, --help            show this help message and exit
-r REFERENCE_VCF, --reference_vcf REFERENCE_VCF
                                                The output directory where results␣
↪will be saved
-c CLUSTER_VCF, --cluster_vcf CLUSTER_VCF
                                                A QC, normalized seurat object with
                                                classifications/clusters as Idents().
-o OUTDIR, --outdir OUTDIR
                                                Number of genes to use in
                                                'Improved_Seurat_Pre_Process'␣
↪function.
```

You can run the reference vs cluster genotypes manually (possibly because your data doesn't have GT, DS or GP genotype formats) or because you would prefer to alter some of the steps. To run the correlations manually, simply start R from the singularity image:

```
singularity exec Demuxafy.sif R
```

Once, R has started, you can load the required libraries (included in the singularity image) and run the code.

```
.libPaths("/usr/local/lib/R/site-library") ### Required so that libraries are loaded
↪from the image instead of locally
library(tidyr)
library(tidyverse)
library(dplyr)
library(vcfR)
library(lsa)
library(ComplexHeatmap)


########## Set up paths and variables ##########

reference_vcf <- "/path/to/reference.vcf"
cluster_vcf <- "/path/to/scSplit/out/scSplit.vcf"
outdir <- "/path/to/scSplit/out/"


########## Set up functions ##########
##### Calculate DS from GP if genotypes in that format #####
calculate_DS <- function(GP_df){
    columns <- c()
    for (i in 1:ncol(GP_df)){
        columns <- c(columns, paste0(colnames(GP_df)[i],"-0"), paste0(colnames(GP_
↪df)[i],"-1"), paste0(colnames(GP_df)[i],"-2"))
    }
    df <- GP_df
    colnames(df) <- paste0("c", colnames(df))
    colnames_orig <- colnames(df)
    for (i in 1:length(colnames_orig)){
        df <- separate(df, sep = ",", col = colnames_orig[i], into = columns[(1+(3*(i-
↪1))):(3+(3*(i-1)))])
    }
    df <- mutate_all(df, function(x) as.numeric(as.character(x)))
    for (i in 1: ncol(GP_df)){
        GP_df[,i] <- df[,(2+((i-1)*3))] + 2* df[,(3+((i-1)*3))]
    }
    return(GP_df)
}

pearson_correlation <- function(df, ref_df, clust_df){
    for (col in colnames(df)){
        for (row in rownames(df)){
            df[row,col] <- cor(as.numeric(pull(ref_df, col)), as.numeric(pull(clust_
↪df, row)), method = "pearson", use = "complete.obs")
        }
    }
    return(df)
}


########## Read in vcf files for each of three non-reference genotype softwares ######
↪####
ref_geno <- read.vcfR(reference_vcf)
cluster_geno <- read.vcfR(cluster_vcf)
```

(continues on next page)

(continued from previous page)

```
########## Convert to tidy data frame ##########
####### Identify which genotype FORMAT to use #######
##### Cluster VCF #####
### Check for each of the different genotype formats ##
## DS ##
format_clust=NA
cluster_geno_tidy <- as_tibble(extract.gt(element = "DS",cluster_geno, IDtoRowNames =␣
↪F))
if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
 ␣
↪message("Found DS genotype format in cluster vcf. Will use that metric for cluster correlation.")
  format_clust = "DS"
}

## GT ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GT",cluster_geno, IDtoRowNames␣
↪= F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
   ␣
↪message("Found GT genotype format in cluster vcf. Will use that metric for cluster correlation.")
    format_clust = "GT"

    if (any(grepl("\\|",cluster_geno_tidy[1,]))){
      separator = "|"
      message("Detected | separator for GT genotype format in cluster vcf")
    } else if (any(grepl("/",cluster_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in cluster vcf")
    } else {
      format_clust = NA
     ␣
↪message("Can't identify a separator for the GT field in cluster vcf, moving on to using GP.")
    }

    cluster_geno_tidy <- as_tibble(lapply(cluster_geno_tidy, function(x)
↪{gsub(paste0("0",separator,"0"),0, x)}) %>%
                             lapply(., function(x) {gsub(paste0("0",separator,"1"),1,␣
↪x)}) %>%
                             lapply(., function(x) {gsub(paste0("1",separator,"0"),1,␣
↪x)}) %>%
                             lapply(., function(x) {gsub(paste0("1",separator,"1"),2,␣
↪x)}))

  }
}

## GP ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GP",cluster_geno, IDtoRowNames␣
↪=F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
    format_clust = "GP"
    cluster_geno_tidy <- calculate_DS(cluster_geno_tidy)
   ␣
↪message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")
```

(continues on next page)

(continued from previous page)

```
  } else {
    print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. F
    q()
  }
}




### Reference VCF ###
### Check for each of the different genotype formats ##
## DS ##
format_ref = NA
ref_geno_tidy <- as_tibble(extract.gt(element = "DS",ref_geno, IDtoRowNames = F))
if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){
  message("Found DS genotype format in reference vcf. Will use that metric for cluster correlation.")
  format_ref = "DS"
}

## GT ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GT",ref_geno, IDtoRowNames = F))
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){
    message("Found GT genotype format in reference vcf. Will use that metric for cluster correlation.")
    format_ref = "GT"

    if (any(grepl("\\|",ref_geno_tidy[1,]))){
      separator = "|"
      message("Detected | separator for GT genotype format in reference vcf")
    } else if (any(grepl("/",ref_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in reference vcf")
    } else {
      format_ref = NA
      message("Can't identify a separator for the GT field in reference vcf, moving on to using GP.")
    }

    ref_geno_tidy <- as_tibble(lapply(ref_geno_tidy, function(x) {gsub(paste0("0",separator,"0"),0, x)}) %>%
                               lapply(., function(x) {gsub(paste0("0",separator,"1"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"0"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"1"),2, x)}))

  }
}

## GP ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GP",ref_geno, IDtoRowNames = F))
```

(continues on next page)

(continued from previous page)

```
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){
    format_clust = "GP"
    ref_geno_tidy <- calculate_DS(ref_geno_tidy)

 ↪message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")

  } else {

 ↪print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. P
    q()
  }
}



### Get SNP IDs that will match between reference and cluster ###
## Account for possibility that the ref or alt might be missing
if ((all(is.na(cluster_geno@fix[,'REF'])) & all(is.na(cluster_geno@fix[,'ALT']))) |
 ↪(all(is.na(ref_geno@fix[,'REF'])) & all(is.na(ref_geno@fix[,'ALT'])))){

 ↪message("The REF and ALT categories are not provided for the reference and/or the cluster vcf. Will
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
 ↪'POS'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'])
} else if (all(is.na(cluster_geno@fix[,'REF'])) | all(is.na(ref_geno@fix[,'REF']))){

 ↪message("The REF categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
 ↪'POS'],"_", cluster_geno@fix[,'REF'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",
 ↪ref_geno@fix[,'REF'])
} else if (all(is.na(cluster_geno@fix[,'ALT'])) | all(is.na(ref_geno@fix[,'ALT']))){

 ↪message("The ALT categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
 ↪'POS'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",
 ↪ref_geno@fix[,'ALT'])
} else {

 ↪message("Found REF and ALT in both cluster and reference genotype vcfs. Will use chromosome, positi
    cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
 ↪'POS'],"_", cluster_geno@fix[,'REF'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",
 ↪ref_geno@fix[,'REF'],"_", ref_geno@fix[,'ALT'])
}


### Update the vcf dfs to remove SNPs with no genotyopes
cluster_geno_tidy <- cluster_geno_tidy[colSums(!is.na(cluster_geno_tidy)) > 0]
ref_geno_tidy <- ref_geno_tidy[colSums(!is.na(ref_geno_tidy)) > 0]



########## Get a unique list of SNPs that is in both the reference and cluster
 ↪genotypes ##########
locations  <- inner_join(ref_geno_tidy[,"ID"],cluster_geno_tidy[,"ID"])
```

(continues on next page)

(continued from previous page)

```
locations <- locations[!(locations$ID %in% locations[duplicated(locations),]$ID),]

########## Keep just the SNPs that overlap ##########
ref_geno_tidy <- left_join(locations, ref_geno_tidy)
cluster_geno_tidy <- left_join(locations, cluster_geno_tidy)

########## Correlate all the cluster genotypes with the individuals genotyped ########
##
##### Make a dataframe that has the clusters as the row names and the individuals as␣
↪the column names #####
pearson_correlations <- as.data.frame(matrix(nrow = (ncol(cluster_geno_tidy) -1),␣
↪ncol = (ncol(ref_geno_tidy) -1)))
colnames(pearson_correlations) <- colnames(ref_geno_tidy)[2:(ncol(ref_geno_tidy))]
rownames(pearson_correlations) <- colnames(cluster_geno_tidy)[2:(ncol(cluster_geno_
↪tidy))]
pearson_correlations <- pearson_correlation(pearson_correlations, ref_geno_tidy,␣
↪cluster_geno_tidy)
cluster <- data.frame("Cluster" = rownames(pearson_correlations))
pearson_correlations_out <- cbind(cluster, pearson_correlations)

########## Save the correlation dataframes ##########
write_delim(pearson_correlations_out, file = paste0(outdir,
↪"/ref_clust_pearson_correlations.tsv"), delim = "\t" )


########## Create correlation figures ##########
col_fun = colorRampPalette(c("white", "red"))(101)
pPearsonCorrelations <- Heatmap(as.matrix(pearson_correlations), cluster_rows = T,␣
↪col = col_fun)

########## Save the correlation figures ##########
png(filename = paste0(outdir,"/ref_clust_pearson_correlation.png"), width = 500)
print(pPearsonCorrelations)
dev.off()

########## Assign individual to cluster based on highest correlating individual ######
↪####
key <- as.data.frame(matrix(nrow = ncol(pearson_correlations), ncol = 3))
colnames(key) <- c("Genotype_ID","Cluster_ID","Correlation")
key$Genotype_ID <- colnames(pearson_correlations)
for (id in key$Genotype_ID){
    if (max(pearson_correlations[,id]) == max(pearson_correlations[rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])],])){
        key$Cluster_ID[which(key$Genotype_ID == id)] <- rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])]
        key$Correlation[which(key$Genotype_ID == id)] <- max(pearson_correlations[,
↪id])
    } else {
        key$Cluster_ID[which(key$Genotype_ID == id)] <- "unassigned"
        key$Correlation[which(key$Genotype_ID == id)] <- NA
    }
}

write_delim(key, file = paste0(outdir,"/Genotype_ID_key.txt"), delim = "\t")
```

2.12.3 ScSplit Results and Interpretation

After running the ScSplit steps and summarizing the results, you will have a number of files from some of the intermediary steps. Theses are the files that most users will find the most informative:

- `scSplit_doublets_singlets.csv`
  - The droplet assignment results. The first column is the droplet barcode and the second column is the droplet type and cluster assignment separated by a dash. For example SNG-9 would indicate that cluster 9 are singlets.

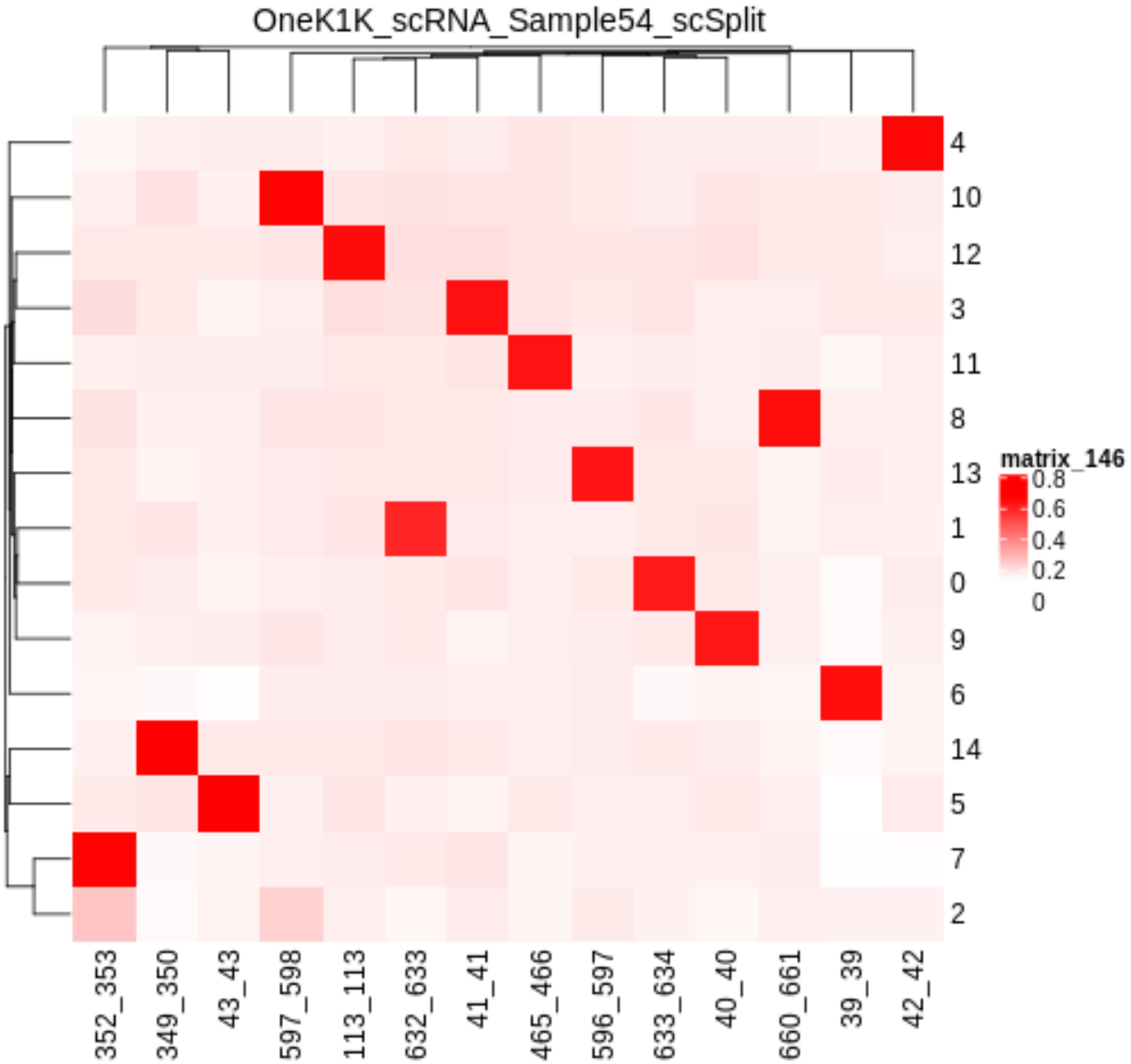
Barcode	Cluster
AAACCTGTCCGAATGT-1	SNG-0
AAACGGGAGTTGAGAT-1	SNG-0
AAACGGGCATGTCTCC-1	SNG-0
AAACGGGTCCACGAAT-1	SNG-0
AAACGGGTCCAGTAGT-1	SNG-0
AAACGGGTCGGCTTGG-1	SNG-0
AAAGATGTCCGAACGC-1	SNG-0
AAAGATGTCCGTCAAA-1	SNG-0
AAAGTAGCATCACGTA-1	SNG-0
...	...

If you ran the `Assign_Indiv_by_Geno.R` script, you will also have the following files:

- `Genotype_ID_key.txt`
  - Key of the cluster and assignments for each individual and the Pearson correlation coefficient.

Genotype_ID	Cluster_ID	Correlation
113_113	12	0.6448151
349_350	14	0.6663323
352_353	7	0.6596409
39_39	6	0.6398297
40_40	9	0.6191905
41_41	3	0.6324396
42_42	4	0.6560180
43_43	5	0.6672336
465_466	11	0.6297396
596_597	13	0.6273717
597_598	10	0.6627428
632_633	1	0.5899685
633_634	0	0.6157936
660_661	8	0.6423770

- `ref_clust_pearson_correlation.png`
  - Figure of the Pearson correlation coefficients for each cluster-individual pair.
- `ref_clust_pearson_correlations.tsv`
  - All of the Pearson correlation coefficients between the clusters and the individuals


OneK1K_scRNA_Sample54_scSplit
4
10
12
3
11
8
13
1
0
9
6
14
5
7
2
352_353
349_350
43_43
597_598
113_113
632_633
41_41
465_466
596_597
633_634
40_40
660_661
39_39
42_42
matrix_146
0.8
0.6
0.4
0.2
0

Cluster	113_113	349_350	352_353	39_39	40_40	...
0	0.18419103983986868	0.16532823032069312	0.12917627297302155	0.15376916805897994	0.19410752490893462	...623
1	0.19853015287740038	0.19816220749550049	0.14924584028347832	0.17855748333388519	0.13455433395443292	...292
2	0.17993959098414505	0.15477058833898626	0.16412833664920995	0.17360648445020146	0.11637461516087665	...657
3	0.21286169961533517	0.17932514814809528	0.18472899166808817	0.19346574998787022	0.17292165137921108	...084
4	0.17573820413419837	0.16295040873107164	0.17642615665946531	0.17427996983606918	0.18322785415879167	...167
...	...	...	...	...	...	...

2.12.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.12.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as ScSplit.

2.13 Souporcell

Souporcell is a genotype-free demultiplexing software that does not require you to have SNP genotypes the donors in your multiplexed capture. However, it can natively integrate SNP genotypes into the demultiplexing if you have them available for **all** the donors in your pool. If you don't have the reference SNP genotypes for all the donors in your multiplexed pool, we have provided some scripts that will help identify clusters from given donors after running Souporcell without the SNP genotypes. Depending on your downstream analyses, if you have reference SNP genotypes for donors in your pool, you could also use *Demuxlet*, or *Vireo*.

One advantage that we have found immensely helpful about Souporcell is that it provides an ambient RNA estimate for the pool. This can be helpful to identify samples that may have high ambient RNA estimates early in the analysis pipeline so that it can be accounted for throughout downstream analyses.

2.13.1 Data

This is the data that you will need to have prepare to run Souporcell:

Required

- Common SNP genotypes vcf (`$VCF`)
  - While not exactly required, using common SNP genotype locations enhances accuracy
    - If you have reference SNP genotypes for individuals in your pool, you can use those
    - If you do not have reference SNP genotypes, they can be from any large population resource (i.e. 1000 Genomes or HRC)
  - Filter for common SNPs (> 5% minor allele frequency) and SNPs overlapping genes
- Barcode file (`$BARCODES`)
- Number of samples in pool (`$N`)

- Bam file (`$BAM`)
  - Aligned single cell reads
- Reference fasta (`$FASTA`)
  - that your reads were aligned to (or at least the same genome)
- Output directory (`$SOUPORCELL_OUTDIR`)

2.13.2 Run Souporcell

You can run Souporcell with or without reference SNP genotypes - follow the instructions for each bellow:

Without Reference SNP Genotypes

With Reference SNP Genotypes

If you don't have reference SNP genotypes for all of your donors, you can run souporcell with the following command, providing an appropriate thread number (`$THREADS`) for your system . Don't worry if you only have reference SNP genotypes for a subset of your donors, we have a script that will correlate the cluster and reference SNP genotypes.

```
singularity exec Demuxafy.sif souporcell_pipeline.py -i $BAM -b $BARCODES -f $FASTA -
↪t $THREADS -o $SOUPORCELL_OUTDIR -k $N --common_variants $VCF
```

If you have reference SNP genotypes for **all** of your donors, you can run souporcell with the following command, providing an appropriate thread number (`$THREADS`) for your system and listing the donor ids that correspond in the `$VCF` file

```
singularity exec Demuxafy.sif souporcell_pipeline.py -i $BAM -b $BARCODES -f $FASTA -
↪t $THREADS -o $SOUPORCELL_OUTDIR -k $N --known_genotypes $VCF --known_genotypes_
↪sample_names donor1 donor donor3 donor4
```

Note

Souporcell can currently only be executed when either **all** or **none** of the individuals that have been pooled have SNP genotypes. Further, the output still has cluster numbers but they should correspond to the order that you listed your individuals. For example, if you have two individuals in your pool (donorA and donorB) and input them as `--known_genotypes_sample_names donorA donorB`, then the output will have two clusters: 0 and 1. donorA will correspond to 0 and donorB will correspond to 1.

Even when we have reference SNP genotypes, we typically runn Souporcell without reference SNP genotypes and then use the cluster vs individual correlations (below) to assign clusters to individuals.

If Souporcell is successful, you will have these files in your `$SOUPORCELL_OUTDIR`:

```
.
├── alt.mtx
├── ambient_rna.txt
├── cluster_genotypes.vcf
├── clustering.done
├── clusters.err
├── clusters_tmp.tsv
├── clusters.tsv
├── common_variants_covered_tmp.vcf
├── common_variants_covered.vcf
```

(continues on next page)

(continued from previous page)

```
├── consensus.done
├── depth_merged.bed
├── doublets.err
├── fastqs.done
├── minimap.err
├── ref.mtx
├── remapping.done
├── retag.err
├── retagging.done
├── souporcell_minimap_tagged_sorted.bam
├── souporcell_minimap_tagged_sorted.bam.bai
├── troublet.done
├── variants.done
└── vartrix.done
```

Additional details about outputs are available below in the *Souporcell Results and Interpretation*.

Souporcell Summary

We have provided a script that will provide a summary of the number of droplets classified as doublets, ambiguous and assigned to each cluster by Souporcell. You can run this to get a fast and easy summary of your results by providing the souporcell result file:

```
singularity exec Demuxafy.sif bash souporcell_summary.sh $SOUPORCELL_OUTDIR/clusters.
↪tsv
```

which should print:

Classification	Assignment N
0	1441
1	980
10	1285
11	1107
12	1315
13	1529
2	1629
3	1473
4	1381
5	1360
6	1157
7	892
8	1111
9	1565
doublet	2757

or you can write the results to file:

```
singularity exec Demuxafy.sif bash souporcell_summary.sh $SOUPORCELL_OUTDIR/clusters.
↪tsv > $SOUPORCELL_OUTDIR/souporcell_summary.tsv
```

Note

To check if these numbers are consistent with the expected doublet rate in your dataset, you can use our Doublet Estimation Calculator.

If the souporcell summary is successful, you will have this new file in your `$SOUPORCELL_OUTDIR`:

```
.
├── alt.mtx
├── ambient_rna.txt
├── cluster_genotypes.vcf
├── clustering.done
├── clusters.err
├── clusters_tmp.tsv
├── clusters.tsv
├── common_variants_covered_tmp.vcf
├── common_variants_covered.vcf
├── consensus.done
├── depth_merged.bed
├── doublets.err
├── fastqs.done
├── minimap.err
├── ref.mtx
├── remapping.done
├── retag.err
├── retagging.done
├── souporcell_minimap_tagged_sorted.bam
├── souporcell_summary.tsv
├── troublet.done
├── variants.done
└── vartrix.done
```

Additional details about outputs are available below in the *Souporcell Results and Interpretation*.

Correlating Cluster to Donor Reference SNP Genotypes (optional)

If you have reference SNP genotypes for some or all of the donors in your pool, you can identify which cluster is best correlated with each donor in your reference SNP genotypes. We have provided a script that will do this and provide a heatmap correlation figure and the predicted individual that should be assigned for each cluster. You can either run it with the script by providing the reference SNP genotypes (`$VCF`), the cluster SNP genotypes (`$SOUPORCELL_OUTDIR/cluster_genotypes.vcf`) and the output directory (`$SOUPORCELL_OUTDIR`) You can run this script with:

Note

In order to do this, your $VCF must be reference SNP genotypes for the individuals in the pool and cannot be a general vcf with common SNP genotype locations from 1000 Genomes or HRC.

With Script

Run in R

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -r $VCF -c $SOUPORCELL_OUTDIR/
↪cluster_genotypes.vcf -o $SOUPORCELL_OUTDIR
```

To see the parameter help menu, type:

```
singularity exec Demuxafy.sif Assign_Indiv_by_Geno.R -h
```

Which will print:

```
usage: Assign_Indiv_by_Geno.R [-h] -r REFERENCE_VCF -c CLUSTER_VCF -o OUTDIR

optional arguments:
-h, --help            show this help message and exit
-r REFERENCE_VCF, --reference_vcf REFERENCE_VCF
                                                    The output directory where results␣
↪will be saved
-c CLUSTER_VCF, --cluster_vcf CLUSTER_VCF
                                                    A QC, normalized seurat object with
                                                    classifications/clusters as Idents().
-o OUTDIR, --outdir OUTDIR
                                                    Number of genes to use in
                                                    'Improved_Seurat_Pre_Process'␣
↪function.
```

You can run the reference vs cluster genotypes manually (possibly because your data doesn't have GT, DS or GP genotype formats) or because you would prefer to alter some of the steps. To run the correlations manually, simply start R from the singularity image:

```
singularity exec Demuxafy.sif R
```

Once, R has started, you can load the required libraries (included in the singularity image) and run the code.

```
.libPaths("/usr/local/lib/R/site-library") ### Required so that libraries are loaded␣
↪from the image instead of locally
library(tidyr)
library(tidyverse)
library(dplyr)
library(vcfR)
library(lsa)
library(ComplexHeatmap)


########## Set up paths and variables ##########

reference_vcf <- "/path/to/reference.vcf"
cluster_vcf <- "/path/to/souporcell/out/cluster_genotypes.vcf"
outdir <- "/path/to/souporcell/out/"


########## Set up functions ##########
##### Calculate DS from GP if genotypes in that format #####
calculate_DS <- function(GP_df){
    columns <- c()
    for (i in 1:ncol(GP_df)){
        columns <- c(columns, paste0(colnames(GP_df)[i],"-0"), paste0(colnames(GP_
↪df)[i],"-1"), paste0(colnames(GP_df)[i],"-2"))
    }
    df <- GP_df
    colnames(df) <- paste0("c", colnames(df))
    colnames_orig <- colnames(df)
    for (i in 1:length(colnames_orig)){
        df <- separate(df, sep = ",", col = colnames_orig[i], into = columns[(1+(3*(i-
↪1))):(3+(3*(i-1)))])
```

(continues on next page)

(continued from previous page)

```
    }
    df <- mutate_all(df, function(x) as.numeric(as.character(x)))
    for (i in 1: ncol(GP_df)){
        GP_df[,i] <- df[,(2+((i-1)*3))] + 2* df[,(3+((i-1)*3))]
    }
    return(GP_df)
}

pearson_correlation <- function(df, ref_df, clust_df){
    for (col in colnames(df)){
        for (row in rownames(df)){
            df[row,col] <- cor(as.numeric(pull(ref_df, col)), as.numeric(pull(clust_
↪df, row)), method = "pearson", use = "complete.obs")
        }
    }
    return(df)
}


########## Read in vcf files for each of three non-reference genotype softwares ######
↪####
ref_geno <- read.vcfR(reference_vcf)
cluster_geno <- read.vcfR(cluster_vcf)



########## Convert to tidy data frame ##########
####### Identify which genotype FORMAT to use #######
##### Cluster VCF #####
### Check for each of the different genotype formats ##
## DS ##
format_clust=NA
cluster_geno_tidy <- as_tibble(extract.gt(element = "DS",cluster_geno, IDtoRowNames =␣
↪F))
if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
 ␣
↪message("Found DS genotype format in cluster vcf. Will use that metric for cluster correlation.")
  format_clust = "DS"
}

## GT ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GT",cluster_geno, IDtoRowNames␣
↪= F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
   ␣
↪message("Found GT genotype format in cluster vcf. Will use that metric for cluster correlation.")
    format_clust = "GT"

    if (any(grepl("\\|",cluster_geno_tidy[1,]))){
      separator = "|"
      message("Detected | separator for GT genotype format in cluster vcf")
    } else if (any(grepl("/",cluster_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in cluster vcf")
    } else {
      format_clust = NA
```

(continues on next page)

(continued from previous page)

```
↪message("Can't identify a separator for the GT field in cluster vcf, moving on to using GP.")
    }

    cluster_geno_tidy <- as_tibble(lapply(cluster_geno_tidy, function(x)
↪{gsub(paste0("0",separator,"0"),0, x)}) %>%
                                   lapply(., function(x) {gsub(paste0("0",separator,"1"),1,
↪x)}) %>%
                                   lapply(., function(x) {gsub(paste0("1",separator,"0"),1,
↪x)}) %>%
                                   lapply(., function(x) {gsub(paste0("1",separator,"1"),2,
↪x)}))

  }
}

## GP ##
if (is.na(format_clust)){
  cluster_geno_tidy <- as_tibble(extract.gt(element = "GP",cluster_geno, IDtoRowNames
↪=F))
  if (!all(colSums(is.na(cluster_geno_tidy)) == nrow(cluster_geno_tidy))){
    format_clust = "GP"
    cluster_geno_tidy <- calculate_DS(cluster_geno_tidy)

↪message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")

  } else {

↪print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. P
    q()
  }
}




### Reference VCF ###
### Check for each of the different genotype formats ##
## DS ##
format_ref = NA
ref_geno_tidy <- as_tibble(extract.gt(element = "DS",ref_geno, IDtoRowNames = F))
if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){

↪message("Found DS genotype format in reference vcf. Will use that metric for cluster correlation.")
  format_ref = "DS"
}

## GT ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GT",ref_geno, IDtoRowNames = F))
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){

↪message("Found GT genotype format in reference vcf. Will use that metric for cluster correlation.")
    format_ref = "GT"

    if (any(grepl("\\|",ref_geno_tidy[1,]))){
```

(continues on next page)

(continued from previous page)

```
      separator = "|"
      message("Detected | separator for GT genotype format in reference vcf")
    } else if (any(grepl("/",ref_geno_tidy[1,]))) {
      separator = "/"
      message("Detected / separator for GT genotype format in reference vcf")
    } else {
      format_ref = NA
      message("Can't identify a separator for the GT field in reference vcf, moving on to using GP.")
    }

    ref_geno_tidy <- as_tibble(lapply(ref_geno_tidy, function(x) {gsub(paste0("0",separator,"0"),0, x)}) %>%
                               lapply(., function(x) {gsub(paste0("0",separator,"1"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"0"),1, x)}) %>%
                               lapply(., function(x) {gsub(paste0("1",separator,"1"),2, x)}))

  }
}

## GP ##
if (is.na(format_ref)){
  ref_geno_tidy <- as_tibble(extract.gt(element = "GP",ref_geno, IDtoRowNames = F))
  if (!all(colSums(is.na(ref_geno_tidy)) == nrow(ref_geno_tidy))){
    format_clust = "GP"
    ref_geno_tidy <- calculate_DS(ref_geno_tidy)
    message("Found GP genotype format in cluster vcf. Will use that metric for cluster correlation.")

  } else {
    print("Could not identify the expected genotype format fields (DS, GT or GP) in your cluster vcf. E
    q()
  }
}



### Get SNP IDs that will match between reference and cluster ###
## Account for possibility that the ref or alt might be missing
if ((all(is.na(cluster_geno@fix[,'REF'])) & all(is.na(cluster_geno@fix[,'ALT']))) |(all(is.na(ref_geno@fix[,'REF'])) & all(is.na(ref_geno@fix[,'ALT'])))){
  message("The REF and ALT categories are not provided for the reference and/or the cluster vcf. Will
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,'POS'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'])
} else if (all(is.na(cluster_geno@fix[,'REF'])) | all(is.na(ref_geno@fix[,'REF']))){
  message("The REF categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,'POS'],"_", cluster_geno@fix[,'REF'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",ref_geno@fix[,'REF'])
```

(continues on next page)

(continued from previous page)

```
} else if (all(is.na(cluster_geno@fix[,'ALT'])) | all(is.na(ref_geno@fix[,'ALT']))){
 ␣
↪message("The ALT categories are not provided for the reference and/or the cluster vcf. Will use the
  cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
↪'POS'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",␣
↪ref_geno@fix[,'ALT'])
} else {
 ␣
↪message("Found REF and ALT in both cluster and reference genotype vcfs. Will use chromosome, positi
    cluster_geno_tidy$ID <- paste0(cluster_geno@fix[,'CHROM'],":", cluster_geno@fix[,
↪'POS'],"_", cluster_geno@fix[,'REF'],"_", cluster_geno@fix[,'ALT'])
  ref_geno_tidy$ID <- paste0(ref_geno@fix[,'CHROM'],":", ref_geno@fix[,'POS'],"_",␣
↪ref_geno@fix[,'REF'],"_", ref_geno@fix[,'ALT'])
}


### Update the vcf dfs to remove SNPs with no genotyopes
cluster_geno_tidy <- cluster_geno_tidy[colSums(!is.na(cluster_geno_tidy)) > 0]
ref_geno_tidy <- ref_geno_tidy[colSums(!is.na(ref_geno_tidy)) > 0]



########## Get a unique list of SNPs that is in both the reference and cluster␣
↪genotypes ##########
locations  <- inner_join(ref_geno_tidy[,"ID"],cluster_geno_tidy[,"ID"])
locations <- locations[!(locations$ID %in% locations[duplicated(locations),]$ID),]

########## Keep just the SNPs that overlap ##########
ref_geno_tidy <- left_join(locations, ref_geno_tidy)
cluster_geno_tidy <- left_join(locations, cluster_geno_tidy)

########## Correlate all the cluster genotypes with the individuals genotyped ########
↪##
##### Make a dataframe that has the clusters as the row names and the individuals as␣
↪the column names #####
pearson_correlations <- as.data.frame(matrix(nrow = (ncol(cluster_geno_tidy) -1),␣
↪ncol = (ncol(ref_geno_tidy) -1)))
colnames(pearson_correlations) <- colnames(ref_geno_tidy)[2:(ncol(ref_geno_tidy))]
rownames(pearson_correlations) <- colnames(cluster_geno_tidy)[2:(ncol(cluster_geno_
↪tidy))]
pearson_correlations <- pearson_correlation(pearson_correlations, ref_geno_tidy,␣
↪cluster_geno_tidy)
cluster <- data.frame("Cluster" = rownames(pearson_correlations))
pearson_correlations_out <- cbind(cluster, pearson_correlations)

########## Save the correlation dataframes ##########
write_delim(pearson_correlations_out, file = paste0(outdir,
↪"/ref_clust_pearson_correlations.tsv"), delim = "\t" )


########## Create correlation figures ##########
col_fun = colorRampPalette(c("white", "red"))(101)
pPearsonCorrelations <- Heatmap(as.matrix(pearson_correlations), cluster_rows = T,␣
↪col = col_fun)

########## Save the correlation figures ##########
```

(continues on next page)

(continued from previous page)

```
png(filename = paste0(outdir,"/ref_clust_pearson_correlation.png"), width = 500)
print(pPearsonCorrelations)
dev.off()

########## Assign individual to cluster based on highest correlating individual ######
↪####
key <- as.data.frame(matrix(nrow = ncol(pearson_correlations), ncol = 3))
colnames(key) <- c("Genotype_ID","Cluster_ID","Correlation")
key$Genotype_ID <- colnames(pearson_correlations)
for (id in key$Genotype_ID){
    if (max(pearson_correlations[,id]) == max(pearson_correlations[rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])],])){
        key$Cluster_ID[which(key$Genotype_ID == id)] <- rownames(pearson_
↪correlations)[which.max(pearson_correlations[,id])]
        key$Correlation[which(key$Genotype_ID == id)] <- max(pearson_correlations[,
↪id])
    } else {
        key$Cluster_ID[which(key$Genotype_ID == id)] <- "unassigned"
        key$Correlation[which(key$Genotype_ID == id)] <- NA
    }
}

write_delim(key, file = paste0(outdir,"/Genotype_ID_key.txt"), delim = "\t")
```

After correlating the cluster and reference donor SNP genotypes, you should have the new results in your directory:

If the souporcell summary is successful, you will have this new file in your `$SOUPORCELL_OUTDIR`:

```
.
├── alt.mtx
├── ambient_rna.txt
├── cluster_genotypes.vcf
├── clustering.done
├── clusters.err
├── clusters_tmp.tsv
├── clusters.tsv
├── common_variants_covered_tmp.vcf
├── common_variants_covered.vcf
├── consensus.done
├── depth_merged.bed
├── doublets.err
├── fastqs.done
├── Genotype_ID_key.txt
├── Individual_genotypes_subset.vcf.gz
├── minimap.err
├── ref.mtx
├── ref_clust_pearson_correlation.png
├── ref_clust_pearson_correlations.tsv
├── remapping.done
├── retag.err
├── retagging.done
├── souporcell_minimap_tagged_sorted.bam
├── souporcell_summary.tsv
├── troublet.done
├── variants.done
└── vartrix.done
```

Additional details about outputs are available below in the *Souporcell Results and Interpretation*.

2.13.3 Souporcell Results and Interpretation

After running the Souporcell steps and summarizing the results, you will have a number of files from some of the intermediary steps. These are the files that most users will find the most informative:

- To check if these numbers are consistent with the expected doublet rate in your dataset, you can use our Expected Doublet Estimation Calculator.
- `clusters.tsv`
  - The Souporcell droplet classifications with the log probabilities of each donor and doublet vs singlet.

barcode	status	assignment	log_prob_singleton	log_prob_doublet	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5	cluster6	cluster7	cluster8	cluster9	cluster10	cluster11	cluster12	cluster13
AAACCTGAGATAGCAT-1	singlet	6	-47.49[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
AAACCTGAGCAGCGTA-1	singlet	[illegible]	-102.8[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
AAACCTGAGCGATGAC-1	singlet	5	-39.97[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
AAACCTGAGCGTAGTG-1	singlet	3	-66.73[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
…	…		…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

- `ambient_rna.txt`
  - The estimated ambient RNA percent in the pool. We typically see < 5% for scRNA-seq PBMCs and < 10% for other scRNA-seq cell types.

```
ambient RNA estimated as 4.071468697320357%
```

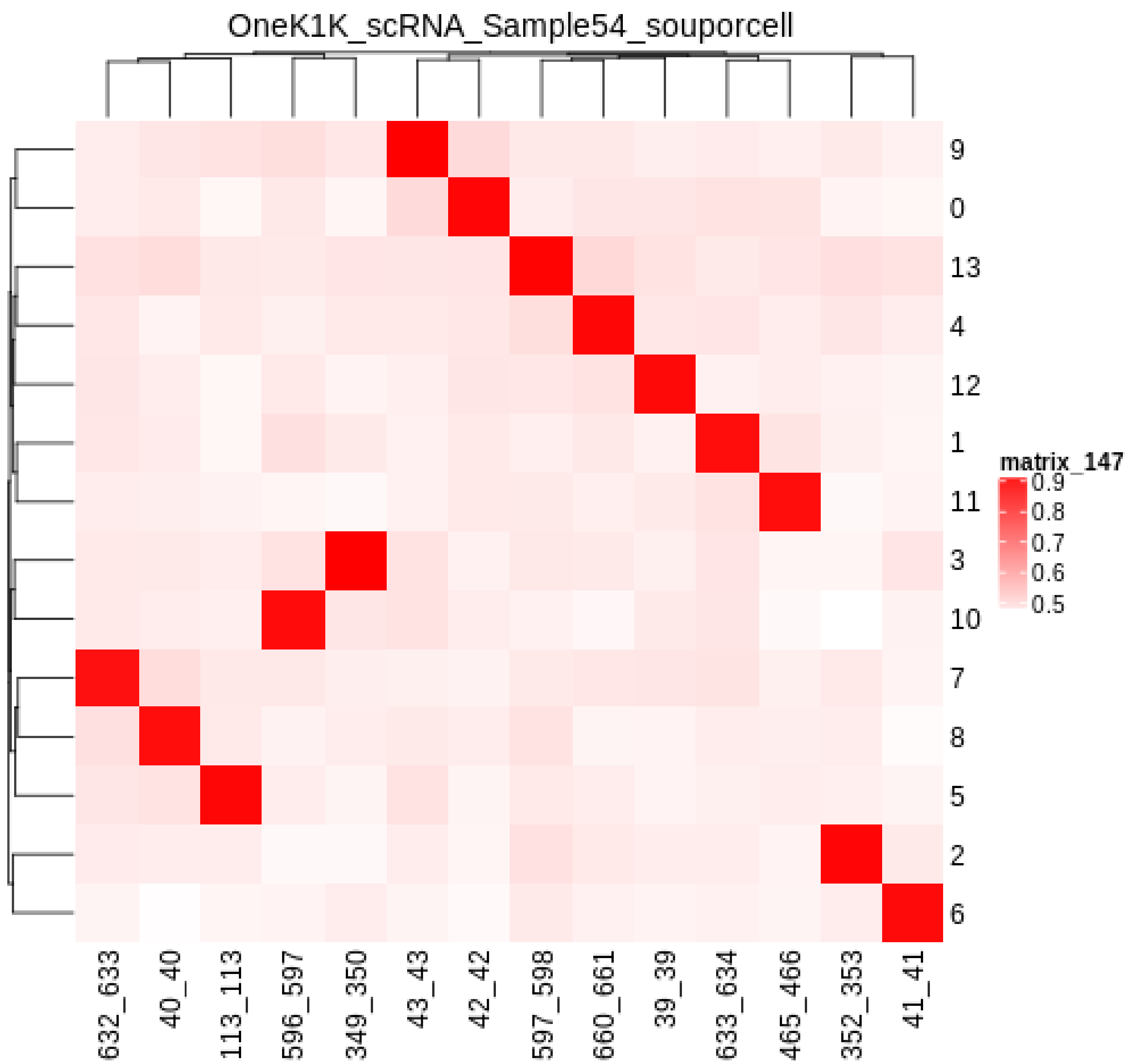
If you ran the `Assign_Indiv_by_Geno.R` script, you will also have the following files:

- `Genotype_ID_key.txt`
  - Key of the cluster and assignments for each individual and the Pearson correlation coefficient.

Genotype_ID	Cluster_ID	Correlation
113_113	5	0.9365902
349_350	3	0.9484794
352_353	2	0.9385500
39_39	12	0.9325007
40_40	8	0.9252865
41_41	6	0.9282633
42_42	0	0.9387788
43_43	9	0.9497327
465_466	11	0.9234109
596_597	10	0.9277824
597_598	13	0.9435752
632_633	7	0.9179054
633_634	1	0.9222734
660_661	4	0.9368751

- `ref_clust_pearson_correlation.png`

- Figure of the Pearson correlation coefficients for each cluster-individual pair.



- `ref_clust_pearson_correlations.tsv`
  - All of the Pearson correlation coefficients between the clusters and the individuals

Cluster	113_113	349_350	352_353	39_39	40_40	…
0	0.4578087241392145	0.458957333501781	0.46351292453350	0.48926720614880	0.48418714411037	91
1	0.4570643404384285	0.452804452734627	0.45026187973225	0.48678187806965	0.49801164797099	736
2	0.47601768323080645	0.452814886065	0.49385500036660	0.4770382927947	0.46763977156991	7855
3	0.477170980829932	0.4847943520673	0.4598361363766	0.4769883259382	0.4982277958757	9728
4	0.48518729333467	0.5248063786743	0.47508275654324	0.42890059449180	0.46471006755998	44
…	…	…	…	…	…	…

2.13.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.13.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as Souporcell.

2.14 Vireo

Vireo is a flexible demultiplexing software that can demutliplex without any reference SNP genotypes, with reference SNP genotypes for a subset of the donors in the pool or no reference SNP genotypes. If you have reference SNP genotypes for **all** of the donors in your pool, you could also use *Demuxlet* or *Souporcell*. If you don't have reference SNP genotypes, you could alternatively use *Freemuxlet* or *ScSplit*.

2.14.1 Data

This is the data that you will need to have preparede to run Vireo:

Required

- Common SNP genotypes vcf (`$VCF`)
  - If you have reference SNP genotypes for individuals in your pool, you can use those
    - For Vireo you should only have the donors that are in this pool in the vcf file
  - If you do not have reference SNP genotypes, they can be from any large population resource (i.e. 1000 Genomes or HRC)
  - Filter for common SNPs (> 5% minor allele frequency) and SNPs overlapping genes
- Barcode file (`$BARCODES`)
- Number of samples in pool (`$N`)
- Bam file (`$BAM`)
  - Aligned single cell reads
- Output directory (`$VIREO_OUTDIR`)

2.14.2 Run Vireo

CellSNP Pileup

First, you need to count the number of alleles at each SNP in each droplet using cellSNP-lite:

```
singularity exec Demuxafy.sif cellsnp-lite -s $BAM -b $BARCODES -O $VIREO_OUTDIR -R
↪$VCF -p 20 --minMAF 0.1 --minCOUNT 20 --gzip
```

You can alter the `-p`, `--minMAF` and `--minCOUNT` parameters to fit your data and your needs. We have found these settings to work well with our data.

If the pileup is successful, you will have this new file in your `$VIREO_OUTDIR`:

```
.
├── cellSNP.base.vcf.gz
├── cellSNP.samples.tsv
├── cellSNP.tag.AD.mtx
├── cellSNP.tag.DP.mtx
└── cellSNP.tag.OTH.mtx
```

Additional details about outputs are available below in the *Vireo Results and Interpretation*.

Demultiplex with Vireo

Next, we can use the cellSNP results to demultiplex the data with Vireo. As already mentioned, you can use Vireo with multiple different levels of reference SNP genotypes. We've provided an example command for each of these differing amounts of donor SNP genotype data.

With SNP Genotype Data for All Donors

With SNP Genotype Data for Some Donors

Without Donor SNP Genotype Data

You will need to provide which genotype measure (`$FIELD`) is provided in your donor SNP genotype file (GT, GP, or PL); default is PL.

STRONGLY Recommended

For Vireo you should only have the donors that are in this pool in the vcf file. Vireo assumes all the individuals in your vcf are in the pool - so if left unfiltered, it will check for all the individuals in the reference SNP genotype file.

Vireo also runs more efficiently when the SNPs from the donor `$VCF` have been filtered for the SNPs identified by `cellSNP-lite`. Therefore, it is highly recommended subset the vcf first.

We can do both of these filtering actions at the same time with *bcftools*:

> **Note:** If your reference SNP genotype `$VCF` is bgzipped (*i.e.* ends in `.vcf.gz`), you should first bgzip and index your file with:
>
> ```
> singularity exec Demuxafy.sif bgzip -c $VCF > $VCF.gz
> singularity exec Demuxafy.sif tabix -p vcf $VCF.gz
> ```

```
singularity exec Demuxafy.sif bcftools view $VCF -R $VIREO_OUTDIR/cellSNP.
↪base.vcf.gz -s sample1,sample2 -Ov -o $VIREO_OUTDIR/donor_subset.vcf
```

> Alternatively, if you have the individuals from the pool in a file with each individuals separated by a new line (`individual_file.tsv`), then you can use `-S individual_file.tsv`.

To run Vireo with reference SNP genotype data for your donors (ideally filtered as shown above):

```
singularity exec Demuxafy.sif vireo -c $VIREO_OUTDIR -d $VIREO_OUTDIR/donor_subset.
↪vcf -o $VIREO_OUTDIR -t $FIELD
```

STRONGLY Recommended

For Vireo you should only have the donors that are in this pool in the reference SNP genotype vcf file. Vireo assumes all the individuals in your vcf are in the pool - so if left unfiltered, it will check for all the individuals in the reference SNP genotype file. It assumes that `$N` is larger than the number of donors in the `$VCF`

Vireo also runs more efficiently when the SNPs from the donor `$VCF` have been filtered for the SNPs identified by `cellSNP-lite`. Therefore, it is highly recommended to subset the vcf first.

We can do both of these filtering actions at the same time with *bcftools*:

> **Note:** If your reference SNP genotype `$VCF` is bgzipped (*i.e.* ends in `.vcf.gz`), you should first bgzip and index your file with:
>
> ```
> singularity exec Demuxafy.sif bgzip -c $VCF > $VCF.gz
> singularity exec Demuxafy.sif tabix -p vcf $VCF.gz
> ```

```
singularity exec Demuxafy.sif bcftools view $VCF -R $VIREO_OUTDIR/cellSNP.
↪base.vcf.gz -s sample1,sample2 -Ov -o $VIREO_OUTDIR/donor_subset.vcf
```

> Alternatively, if you have the individuals from the pool in a file with each individuals separated by a new line (`individual_file.tsv`), then you can use `-S individual_file.tsv`.

Recommended

Vireo runs more efficiently when the SNPs from the donor `$VCF` have been filtered for the SNPs identified by `cellSNP-lite`. Therefore, it is highly recommended subset the vcf as follows first:

```
singularity exec Demuxafy.sif bcftools view $VCF -R $VIREO_OUTDIR/cellSNP.
↪base.vcf.gz -Oz -o $VIREO_OUTDIR/donor_subset.vcf
```

```
singularity exec Demuxafy.sif vireo -c $VIREO_OUTDIR -d $VIREO_OUTDIR/donor_subset.
↪vcf.gz -o $VIREO_OUTDIR -t $FIELD -N $N
```

```
singularity exec Demuxafy.sif vireo -c $VIREO_OUTDIR -o $VIREO_OUTDIR -N $N
```

If Vireo is successful, you will have these new files in your `$VIREO_OUTDIR`:

```
.
├── cellSNP.base.vcf
├── cellSNP.samples.tsv
├── cellSNP.tag.AD.mtx
├── cellSNP.tag.DP.mtx
├── cellSNP.tag.OTH.mtx
├── donor_ids.tsv
├── donor_subset.vcf
├── fig_GT_distance_estimated.pdf
├── _log.txt
├── prob_doublet.tsv.gz
├── prob_singlet.tsv.gz
└── summary.tsv
```

Additional details about outputs are available below in the *Vireo Results and Interpretation*.

2.14.3 Vireo Results and Interpretation

After running the Vireo steps, you will have a number of files in your `$VIREO_OUTDIR`. These are the files that most users will find the most informative:

- `summary.tsv`
  - A summary of the droplets assigned to each donor, doublets and unassigned.

Var1	Freq
113_113	1342
349_350	1475
352_353	1619
39_39	1309
40_40	1097
41_41	1144
42_42	1430
43_43	1561
465_466	1104
596_597	1271
597_598	1532
632_633	871
633_634	967
660_661	1377
doublet	2770
unassigned	113

    - To check whether the number of doublets identified by Vireo is consistent with the expected doublet rate based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.

- `donor_ids.tsv`
  - The classification of each droplet, and some droplet metrics.

cell	donor_id	prob_max	prob_doublet	n_vars	best_singlet	best_doublet
AAACCTGAGATAGCAT-1	41_41	1.00e+00	9.13e-09	115	41_41	40_40,41_41
AAACCTGAGCAGCGTA-1	465_466	1.00e+00	5.03e-17	239	465_466	349_350,43_43
AAACCTGAGCGATGAC-1	113_113	1.00e+00	7.57e-07	98	113_113	113_113,633_634
AAACCTGAGCGTAGTG-1	349_350	1.00e+00	8.07e-07	140	349_350	349_350,597_598
AAACCTGAGGAGTTTA-1	632_633	1.00e+00	5.99e-11	177	632_633	40_40,113_113
AAACCTGAGGCTCATT-1	39_39	1.00e+00	4.44e-06	110	39_39	39_39,40_40

2.14.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.14.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as Vireo.

2.15 Overview of Doublet Detecting Softwares

Transcription-based doublet detection softwares use the transcriptomic profiles in each cell to predict whether that cell is a singlet or doublet. Most methods simulate doublets by adding the transcriptional profiles of two droplets in your pool together. Therefore, these approaches assume that only a small percentage of the droplets in your dataset are doublets. The table bellow provides a comparison of the different methods.

If you don't know which demultiplexing software(s) to run, take a look at our *Software Selection Recommendations* based on your dataset.

2.16 DoubletDecon

DoubletDecon is a transcription-based doublet detection software that uses deconvolution to identify doublets using the *R* statistical software. We have provided a wrapper script that takes common arguments for DoubletDecon and also provide example code for you to run manually if you prefer.

2.16.1 Data

This is the data that you will need to have prepare to run DoubletDecon:

Required

- A QC-filtered and normalized seurat object saved as an `rds` object (`$SEURAT_RDS`)
  - For example, using the Seurat Vignette
  - If you run DoubletDecon manually, you can use any data format of interest and read in with a method that works for your data.
- Output directory (`$DOUBLETDECON_OUTDIR`)

2.16.2 Run DoubletDecon

You can either run DoubletDecon with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

With Wrapper Script

Run in R

Note

Since it is hard to predict the correct *rhop* to use for each dataset, we typically run a range. For example: 0.6, 0.7, 0.8, 0.9, 1, and 1.1. Then we select the results that predict the number of doublets closest to the expected doublet number. You can estimate that number with our **doublet calculator** The *rhop* parameter can be set with `-r` or `--rhop` in the command below.

```
singularity exec Demuxafy.sif DoubletDecon.R -o $DOUBLETDECON_OUTDIR -s $SEURAT_RDS
```

You can provide many other parameters as well which can be seen from running a help request:

```
singularity exec image DoubletDecon.R -h

usage: DoubletDecon.R [-h] -o OUT -s SEURAT_OBJECT [-g NUM_GENES] [-r RHOP]
                      [-p SPECIES] [-n NCORES] [-c REMOVECC] [-m PMF]
                      [-f HEATMAP] [-t CENTROIDS] [-d NUM_DOUBS] [-5 ONLY50]
                      [-u MIN_UNIQ]

optional arguments:
  -h, --help            show this help message and exit
  -o OUT, --out OUT     The output directory where results will be saved
  -s SEURAT_OBJECT, --seurat_object SEURAT_OBJECT
                        A QC, normalized seurat object with classifications/clusters␣
↪as Idents() saved as an rds object.
  -g NUM_GENES, --num_genes NUM_GENES
                        Number of genes to use in 'Improved_Seurat_Pre_Process'␣
↪function.
  -r RHOP, --rhop RHOP  rhop to use in DoubletDecon - the number of SD from the mean␣
↪to identify upper limit to blacklist
  -p SPECIES, --species SPECIES
                        The species of your sample. Can be scientific species name,␣
↪KEGG ID, three letter species abbreviation, or NCBI ID.
  -n NCORES, --nCores NCORES
                        The number of unique cores you would like to use to run␣
↪DoubletDecon. By default, uses one less than available detected.
  -c REMOVECC, --removeCC REMOVECC
                        Whether to remove clusters enriched in cell cycle genes.
  -m PMF, --pmf PMF     Whether to use unique gene expression in doublet␣
↪determination.
  -f HEATMAP, --heatmap HEATMAP
                        Whether to generate heatmaps.
  -t CENTROIDS, --centroids CENTROIDS
                        Whether to use centroids instead of medoids for doublet␣
↪detecting.
  -d NUM_DOUBS, --num_doubs NUM_DOUBS
                        The number of doublets to simulate for each cluster pair.
  -5 ONLY50, --only50 ONLY50
                        Whether to only compute doublets as 50:50 ratio. Default is␣
↪to use other ratios as well.
```

(continues on next page)

(continued from previous page)

```
  -u MIN_UNIQ, --min_uniq MIN_UNIQ
                        Minimum number of unique genes to rescue a cluster identified␣
↪as doublets.
```

First, you will have to start R. We have built R and all the required software to run DoubletDecon into the singularity image so you can run it directly from the image.

```
singularity exec Demuxafy.sif R
```

That will open R in your terminal. Next, you can load all the libraries and run DoubletDecon.

```
.libPaths("/usr/local/lib/R/site-library") ### This is required so that R uses the␣
↪libraries loaded in the image and not any local libraries
library(DoubletDecon)
library(tidyverse)
library(Seurat)
library(ggplot2)
library(data.table)

## Set up variables ##
out <- "/path/to/doubletdecon/outdir"
SEURAT_RDSect <- "/path/to/preprocessed/SEURAT_RDSect.rds"




## make sure the directory exists ###
dir.create(out, recursive = TRUE)

## Read in Data ##
seurat <- readRDS(SEURAT_RDSect)

## Preprocess ##
processed <- Improved_Seurat_Pre_Process(seurat, num_genes=50, write_files=FALSE)

## Run Doublet Decon ##
results <- Main_Doublet_Decon(rawDataFile = processed$newExpressionFile,
  groupsFile = processed$newGroupsFile,
  filename = "DoubletDecon_results",
  location = paste0(out, "/"),
  fullDataFile = NULL,
  removeCC = FALSE,
  species = "hsa",
  rhop = 0.9,                          ## We recommend testing multiple rhop␣
↪parameters to find which fits your data the best
  write = TRUE,
  PMF = TRUE,
  useFull = FALSE,
  heatmap = FALSE,
  centroids=FALSE,
  num_doubs=100,
  only50=FALSE,
  min_uniq=4,
  nCores = 1)
```

(continues on next page)

(continued from previous page)

```
doublets <- read.table(paste0(out, "/Final_doublets_groups_DoubletDecon_results.txt"))
doublets$Barcode <- gsub("\\.", "-",rownames(doublets))
doublets$DoubletDecon_DropletType <- "doublet"
doublets$V1 <- NULL
doublets$V2 <- NULL


singlets <- read.table(paste0(out,␣
↪"/Final_nondoublets_groups_DoubletDecon_results.txt"))
singlets$Barcode <- gsub("\\.", "-",rownames(singlets))
singlets$DoubletDecon_DropletType <- "singlet"
singlets$V1 <- NULL
singlets$V2 <- NULL

doublets_singlets <- rbind(singlets,doublets)

fwrite(doublets_singlets, paste0(out, "/DoubletDecon_doublets_singlets.tsv"), sep =␣
↪"\t", append = FALSE)


### Make a summary of the number of singlets and doublets
summary <- as.data.frame(table(doublets_singlets$DoubletDecon_DropletType))
colnames(summary) <- c("Classification", "Droplet N")
fwrite(summary, paste0(out,"/DoubletDecon_doublet_summary.tsv"), sep = "\t", append =␣
↪FALSE)
```

2.16.3 DoubletDecon Results and Interpretation

After running the DoubletDecon, you will have multiple files in the `$DOUBLETDECON_OUTDIR`:

```
.
├── data_processed_DoubletDecon_results.txt
├── data_processed_reclust_DoubletDecon_results.txt
├── DoubletDecon_doublets_singlets.tsv
├── DoubletDecon_doublet_summary.tsv
├── DoubletDecon_results.log
├── DRS_doublet_table_DoubletDecon_results.txt
├── DRS_results_DoubletDecon_results.txt
├── Final_doublets_exp_DoubletDecon_results.txt
├── Final_doublets_groups_DoubletDecon_results.txt
├── Final_nondoublets_exp_DoubletDecon_results.txt
├── Final_nondoublets_groups_DoubletDecon_results.txt
├── groups_processed_DoubletDecon_results.txt
├── groups_processed_reclust_DoubletDecon_results.txt
├── new_PMF_results_DoubletDecon_results.txt
├── resultsreadable_synths.txt
└── Synth_doublet_info_DoubletDecon_results.txt
```

DoubletDecon puts most of the results in multiple separate files. However, the wrapper script and the example code has some steps to combine these results together into a single file, which will likely be the most informative output. These are the files that we think will be the most helpful for users:

- `DoubletDecon_doublet_summary.tsv`

  - A summary of the number of singlets and doublets predicted by DoubletDecon.

Classification	Droplet N
doublet	1510
singlet	19470

    * To check whether the numbe of doublets identified by DoubletDecon is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.

- `DoubletDecon_doublets_singlets.tsv`
  - The per-barcode singlet and doublet classification from DoubletDecon.

Barcode	DoubletDecon_DropletType
AAACCTGAGCAGCGTA-1	singlet
AAACCTGAGCGATGAC-1	singlet
AAACCTGAGCGTAGTG-1	singlet
AAACCTGAGGCTCATT-1	singlet
AAACCTGAGTAGCCGA-1	singlet
...	...

2.16.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.16.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as DoubletDecon.

2.17 DoubletDetection

DoubletDetection is a transcription-based doublet detection software. This was one of the better-performing doublet detecting softwares that we identified in our paper (**CITE**) and it is also relatively fast to run. We have provided a wrapper script that enables DoubletDetection to be easily run from the command line but we also provide example code so that users can run manually as well depending on their data.

2.17.1 Data

This is the data that you will need to have prepare to run DoubletDetection:

Required

- A counts matrix (`$COUNTS`)
  - DoubletDetection expects counts to be in the cellranger output format either as
    * h5 file (`filtered_feature_bc_matrix.h5`)

      or

    * matrix directory (directory containing `barcodes.tsv`, `genes.tsv` and `matrix.mtx` **or** `barcodes.tsv.gz`, `features.tsv.gz` and `matrix.mtx.gz`)
    * If you don't have your data in either of these formats, you can run DoubletDetection manually in python and load the data in using a method of your choosing.

Optional

- Output directory (`$DOUBLETDETECTION_OUTDIR`)
  - If you don't provide an `$DOUBLETDETECTION_OUTDIR`, the results will be written to the present working directory.

2.17.2 Run DoubletDetection

You can either run DoubletDetection with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

With Wrapper Script

Run in python

```
singularity exec Demuxafy.sif DoubletDetection.py -m $COUNTS -o $DOUBLETDETECTION_
↪OUTDIR
```

To see all the parameters that this wrapper script will accept, run:

```
singularity exec Demuxafy.sif DoubletDetection.py -h

usage: DoubletDetection.py [-h] -m COUNTS_MATRIX [-b BARCODES] [-o OUTDIR] [-i N_
↪ITERATIONS] [-p PHENOGRAPH] [-s STANDARD_SCALING] [-t P_THRESH] [-v VOTER_THRESH]

wrapper for DoubletDetection for doublet detection from transcriptomic data.

optional arguments:
  -h, --help            show this help message and exit
  -m COUNTS_MATRIX, --counts_matrix COUNTS_MATRIX
                        cell ranger counts matrix directory containing matrix files␣
↪or full path to matrix.mtx. Can also also provide the 10x h5.
  -b BARCODES, --barcodes BARCODES
                        File containing droplet barcodes. Use barcodes from provided␣
↪10x dir by default.
  -o OUTDIR, --outdir OUTDIR
                        The output directory; default is current working directory
  -i N_ITERATIONS, --n_iterations N_ITERATIONS
                        Number of iterations to use; default is 50
  -p PHENOGRAPH, --phenograph PHENOGRAPH
                        Whether to use phenograph (True) or not (False); default is␣
↪False
  -s STANDARD_SCALING, --standard_scaling STANDARD_SCALING
                        Whether to use standard scaling of normalized count matrix␣
↪prior to PCA (True) or not (False); default is True
  -t P_THRESH, --p_thresh P_THRESH
                        P-value threshold for doublet calling; default is 1e-16
  -v VOTER_THRESH, --voter_thresh VOTER_THRESH
                        Voter threshold for doublet calling; default is 0.5
```

To run DoubletDetection manually, first start python from the singularity image (all the required software have been provided in the image)

```
singularity exec Demuxafy.sif python
```

Now, python will open in your terminal and you can run the DoubletDetection code. Here is an example:

```
import os
import numpy as np
import doubletdetection
import tarfile
import matplotlib
matplotlib.use('PDF')
import matplotlib.pyplot as plt
import sys
import pandas as pd

# Load read10x function from mods directory

mods_path = "/opt/Demultiplexing_Doublet_Detecting_Docs/mods" ## custom script for␣
↪loading 10x data in python
sys.path.append(mods_path)
import read10x

### Set up parameters and variables ###
counts_matrix = "/path/to/counts/matrix.mtx"
outdir = "/path/to/doublet/detection/outdir"


if not os.path.isdir(outdir):
  os.mkdir(outdir)


### Read in data ###
raw_counts = read10x.import_cellranger_mtx(counts_matrix)

try:
  barcodes_df = read10x.read_barcodes(counts_matrix + "/barcodes.tsv.gz")
except:
  try:
    barcodes_df = read10x.read_barcodes(counts_matrix + "/barcodes.tsv")
  except:
   ␣
↪print("No barcode file in provided counts matrix directory. Please double check the directory or pr

print('Counts matrix shape: {} rows, {} columns'.format(raw_counts.shape[0], raw_
↪counts.shape[1]))

# Remove columns with all 0s
zero_genes = (np.sum(raw_counts, axis=0) == 0).A.ravel()
raw_counts = raw_counts[:, ~zero_genes]
print('Counts matrix shape after removing unexpressed genes: {} rows, {} columns'.
↪format(raw_counts.shape[0], raw_counts.shape[1]))

clf = doubletdetection.BoostClassifier(n_iters=50, use_phenograph=True, standard_
↪scaling=False, verbose = True)
doublets = clf.fit(raw_counts).predict(p_thresh=1e-16, voter_thresh=50)
```

(continues on next page)

(continued from previous page)

```
results = pd.Series(doublets, name="DoubletDetection_DropletType")
dataframe = pd.concat([barcodes_df, results], axis=1)
dataframe.DoubletDetection_DropletType = dataframe.DoubletDetection_DropletType.
↪replace(1.0, "doublet")
dataframe.DoubletDetection_DropletType = dataframe.DoubletDetection_DropletType.
↪replace(0.0, "singlet")

dataframe.to_csv(os.path.join(outdir,'DoubletDetection_doublets_singlets.tsv'), sep =␣
↪"\t", index = False)


### Figures ###
doubletdetection.plot.convergence(clf, save=os.path.join(outdir,
↪'convergence_test.pdf'), show=False, p_thresh=1e-16, voter_thresh=0.5)

f3 = doubletdetection.plot.threshold(clf, save=os.path.join(outdir,
↪'threshold_test.pdf'), show=False, p_step=6)


### Make summary of singlets and doublets and write to file ###
summary = pd.DataFrame(dataframe.DoubletDetection_DropletType.value_counts())
summary.index.name = 'Classification'
summary.reset_index(inplace=True)
summary = summary.rename({'DoubletDetection_DropletType': 'Droplet N'}, axis=1)

summary.to_csv(os.path.join(outdir,'DoubletDetection_summary.tsv'), sep = "\t", index␣
↪= False)
```

2.17.3 DoubletDetection Results and Interpretation

After running the DoubletDetection, you will have multiple files in the `$DOUBLETDETECTION_OUTDIR`:

```
.
├── convergence_test.pdf
├── DoubletDetection_doublets_singlets.tsv
├── DoubletDetection_summary.tsv
└── threshold_test.pdf
```

We have found these to be the most helpful:

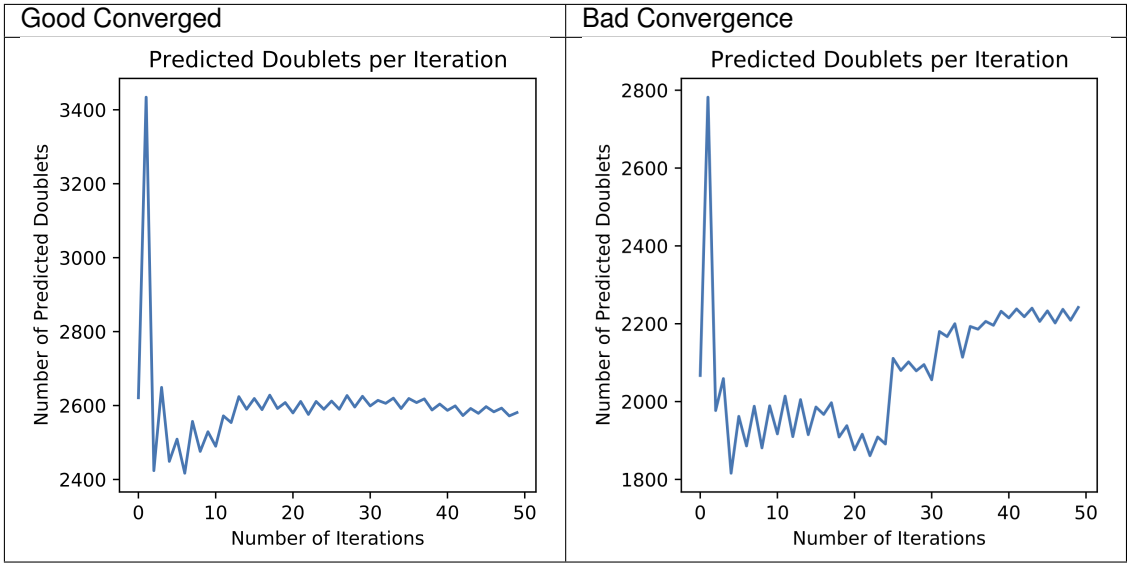
- `DoubletDetection_summary.tsv`
  - A summary of the number of singlets and doublets predicted by DoubletDetection.

DoubletDetection_DropletType	Droplet N
doublet	2594
singlet	18388

  - To check whether the number of doublets identified by DoubletDetection is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.
- `DoubletDetection_doublets_singlets.tsv`
  - The per-barcode singlet and doublet classification from DoubletDetection.

Barcode	DoubletDetection_DropletType
AAACCTGAGATAGCAT-1	singlet
AAACCTGAGCAGCGTA-1	singlet
AAACCTGAGCGATGAC-1	singlet
AAACCTGAGCGTAGTG-1	singlet
AAACCTGAGGAGTTTA-1	singlet
AAACCTGAGGCTCATT-1	singlet
AAACCTGAGGGCACTA-1	singlet
...	...

- `convergence_test.pdf`
  - The expectation is that after multiple rounds, the expected number of doublets will converge. If that is not the case, we suggest that you run DoubletDetection for more iterations (try 150, or even 250 if that isn't convincing).
  - Here are two figures - one of a sample that came to convergence after 50 iterations (left) and one that did not (right)



2.17.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.17.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as DoubletDetection.

2.18 DoubletFinder

DoubletFinder is a transcription-based doublet detection software that uses simulated doublets to find droplets that has a high proportion of neighbors that are doublets. We have provided a wrapper script that takes common arguments for DoubletFinder and we also provide an example script that you can run manually in R if you prefer.

2.18.1 Data

This is the data that you will need to have preparede to run DoubletFinder:

Required

- A QC-filtered and normalized seurat object saved as an `rds` object (`$SEURAT_RDS`)
  - For example, using the Seurat Vignette
  - If you run DoubletFinder manually, you can use any data format of interest and read in with a method that works for your data.
- Output directory (`$DOUBLETFINDER_OUTDIR`)
- Expected number of doublets (`$DOUBLETS`)
  - This can be calculated based on the number of droplets captured using our **doublet calculator**

2.18.2 Run DoubletFinder

You can either run DoubletFinder with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

With Wrapper Script

Run in R

```
singularity exec Demuxafy.sif DoubletFinder.R -o $DOUBLETFINDER_OUTDIR -s $SEURAT_RDS␣
↪-c TRUE -d $DOUBLETS
```

You can provide many other parameters as well which can be seen from running a help request:

```
singularity exec Demuxafy.sif DoubletFinder.R -h


usage: DoubletFinder.R [-h] -o OUT -s SEURAT_OBJECT -c SCT -d DOUBLET_NUMBER [-p PCS]␣
↪[-n PN]

optional arguments:
  -h, --help            show this help message and exit
  -o OUT, --out OUT     The output directory where results will be saved
  -s SEURAT_OBJECT, --seurat_object SEURAT_OBJECT
                        A QC, normalized seurat object with classifications/clusters␣
↪as Idents() saved as an rds object.
  -c SCT, --sct SCT     Whether sctransform was used for normalization.
  -d DOUBLET_NUMBER, --doublet_number DOUBLET_NUMBER
                        Number of expected doublets based on droplets captured.
  -p PCS, --PCs PCS     Number of PCs to use for 'doubletFinder_v3' function.
  -n PN, --pN PN        Number of doublets to simulate as a proportion of the pool␣
↪size.
```

First, you will have to start R. We have built R and all the required software to run DoubletFinder into the singularity image so you can run it directly from the image.

```
singularity exec Demuxafy.sif R
```

That will open R in your terminal. Next, you can load all the libraries and run DoubletFinder.

```
.libPaths("/usr/local/lib/R/site-library") ### This is required so that R uses the␣
↪libraries loaded in the image and not any local libraries
library(Seurat)
library(ggplot2)
library(DoubletFinder)
library(dplyr)
library(tidyr)
library(tidyverse)

## Set up parameters ##
out <- "/path/to/doubletfinder/outdir"
SEURAT_RDSect <- "/path/to/preprocessed/SEURAT_RDSect.rds"
doublet_number <- 3200

## make sure the directory exists ###
dir.create(out, recursive = TRUE)

## Add max future globals size for large pools
options(future.globals.maxSize=(850*1024^2))

### Read in the data
seurat <- readRDS(SEURAT_RDSect)


## pK Identification (no ground-truth) ---------------------------------------------
↪---------------------------------------
sweep.res.list <- paramSweep_v3(seurat, PCs = 1:10, sct = TRUE)
sweep.stats <- summarizeSweep(sweep.res.list, GT = FALSE)
bcmvn <- find.pK(sweep.stats)
plot <- ggplot(bcmvn, aes(pK, BCmetric)) +
    geom_point()
```

(continues on next page)

(continued from previous page)

```
ggsave(plot, filename = paste0(out,"/pKvBCmetric.png"))

## Homotypic Doublet Proportion Estimate -------------------------------------------
↪---------------------------------------
annotations <- Idents(seurat)
homotypic.prop <- modelHomotypic(annotations)
nExp_poi <- doublet_number
print(paste0("Expected number of doublets: ", doublet_number))
nExp_poi.adj <- round(doublet_number*(1-homotypic.prop))

## Run DoubletFinder with varying classification stringencies ----------------------
↪---------------------------------------
seurat <- doubletFinder_v3(seurat, PCs = 1:10, pN = 0.25, pK = as.numeric(as.
↪character(bcmvn$pK[which(bcmvn$BCmetric == max(bcmvn$BCmetric))])), nExp = nExp_poi.
↪adj, reuse.pANN = FALSE, sct = TRUE)
doublets <- as.data.frame(cbind(colnames(seurat), seurat@meta.data[,
↪grepl(paste0("pANN_0.25_",as.numeric(as.character(bcmvn$pK[which(bcmvn$BCmetric ==␣
↪max(bcmvn$BCmetric))]))), colnames(seurat@meta.data))], seurat@meta.data[,
↪grepl(paste0("DF.classifications_0.25_",as.numeric(as.character(bcmvn$pK[which(bcmvn
↪$BCmetric == max(bcmvn$BCmetric))]))), colnames(seurat@meta.data))]))
colnames(doublets) <-  c("Barcode","DoubletFinder_score","DoubletFinder_DropletType")
doublets$DoubletFinder_DropletType <- gsub("Singlet","singlet",doublets$DoubletFinder_
↪DropletType) %>% gsub("Doublet","doublet",.)

write_delim(doublets, file = paste0(out,"/DoubletFinder_doublets_singlets.tsv"),␣
↪delim = "\t")

### Calculate number of doublets and singlets ###
summary <- as.data.frame(table(doublets$DoubletFinder_DropletType))
colnames(summary) <- c("Classification", "Droplet N")
write_delim(summary, paste0(out,"/DoubletFinder_doublet_summary.tsv"), "\t")
```

2.18.3 DoubletFinder Results and Interpretation

After running the DoubletFinder, you will have multiple files in the `$DOUBLETFINDER_OUTDIR`:

```
.
├── DoubletFinder_doublets_singlets.tsv
├── DoubletFinder_doublet_summary.tsv
└── pKvBCmetric.png
```

Here's a more detailed description of the contents of each of those files:

- `DoubletFinder_doublet_summary.tsv`
  - A sumamry of the number of singlets and doublets predicted by DoubletFinder.

Classification	Droplet N
doublet	3014
singlet	16395

    * To check whether the numbe of doublets identified by DoubletFinder is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.

- `DoubletFinder_doublets_singlets.tsv`

  - The per-barcode singlet and doublet classification from DoubletFinder.

Barcode	DoubletFinder_score	DoubletFinder_DropletType
AAACCTGAGATAGCAT-1	0.206401766004415	singlet
AAACCTGAGCAGCGTA-1	0.144039735099338	singlet
AAACCTGAGCGATGAC-1	0.191501103752759	singlet
AAACCTGAGCGTAGTG-1	0.212472406181015	singlet
AAACCTGAGGAGTTTA-1	0.242273730684327	singlet
AAACCTGAGGCTCATT-1	0.211368653421634	singlet
AAACCTGAGGGCACTA-1	0.626379690949227	doublet
...	...	...

- `pKvBCmetric.png`
  - This is the metric that DoubletFinder uses to call doublets and singlets. Typically the `pK` value at the maximum `BC` value is the best doublet calling threshold.
  - If you do not have a clear `BC` maximum, see responses from the DoubletFinder developer here and here for possible solutions.

2.18.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.18.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as DoubletFinder.

2.19 ScDblFinder

scDblFinder is a transcriptome-based doublet detecting method that uses doublet simulation from droplets in the dataset to identify doublets. We have provided a wrapper script that takes common arguments for ScDblFinder and also provide example code for you to run manually if you prefer.

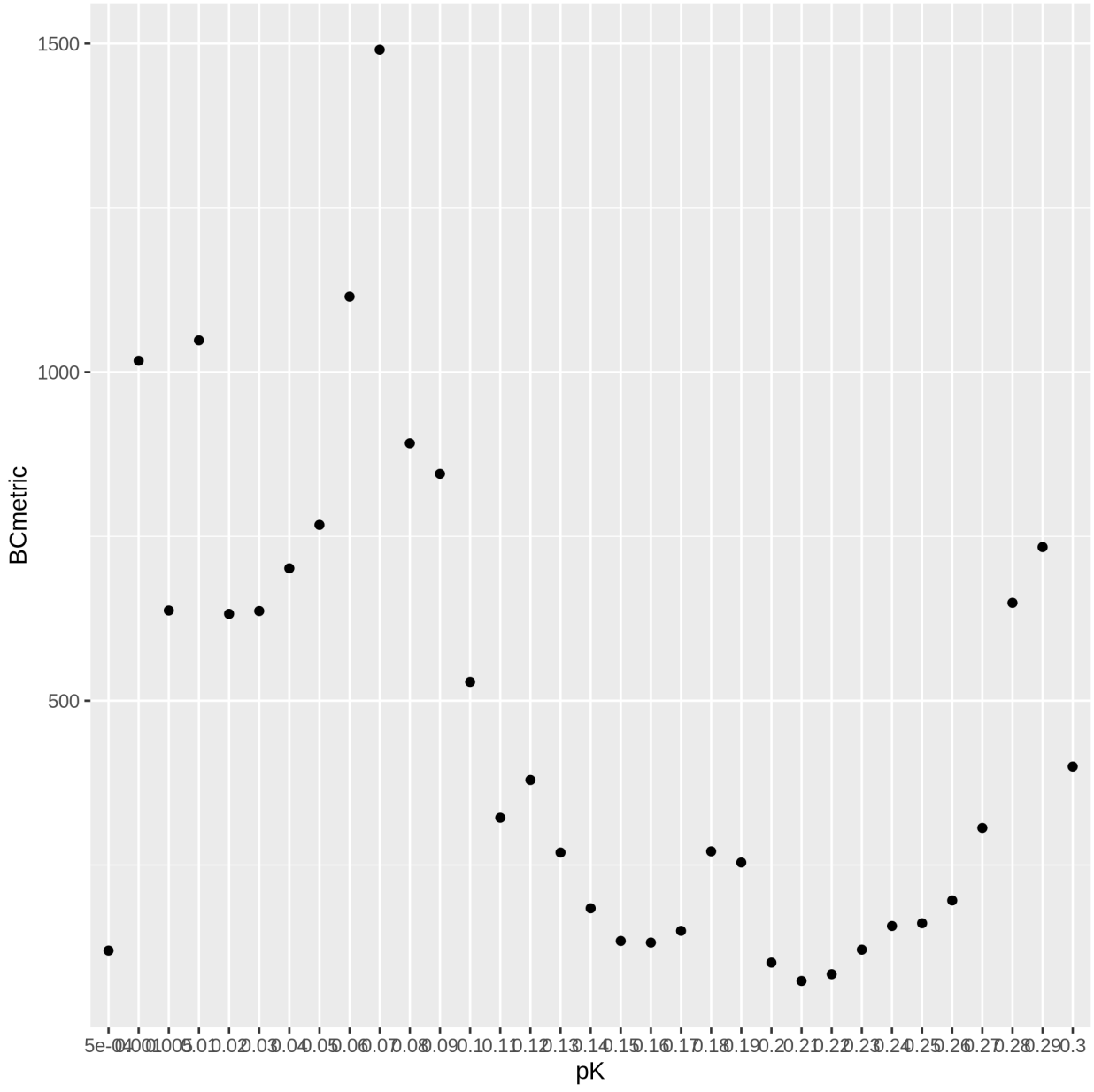
2.19.1 Data

This is the data that you will need to have prepare to run ScDblFinder:

Required

- A counts matrix (`$COUNTS`)
  - The directory path containing your cellranger counts matrix files (directory containing `barcodes.tsv`, `genes.tsv` and `matrix.mtx` **or** `barcodes.tsv.gz`, `features.tsv.gz` and `matrix.mtx.gz`)

    or
  - h5 file (`filtered_feature_bc_matrix.h5`)


1500
1000
500
BCmetric
pK

    * If you don't have your data in this format, you can run ScDblFinder manually in R and load the data in using a method of your choosing.

- Output directory (`$SCDBLFINDER_OUTDIR`)
  - If you don't provide an `$SCDBLFINDER_OUTDIR`, the results will be written to the present working directory.

2.19.2 Run ScDblFinder

You can either run ScDblFinder with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

With Wrapper Script

Run in R

```
singularity exec Demuxafy.sif scDblFinder.R -o $SCDBLFINDER_OUTDIR -t $COUNTS
```

First, you will have to start R. We have built R and all the required software to run ScDblFinder into the singularity image so you can run it directly from the image.

```
singularity exec Demuxafy.sif R
```

That will open R in your terminal. Next, you can load all the libraries and run ScDblFinder.

```
.libPaths("/usr/local/lib/R/site-library") ### This is required so that R uses the␣
↪libraries loaded in the image and not any local libraries
library(scDblFinder)
library(Seurat)
library(SingleCellExperiment)
library(tidyverse)

## Set up variables and parameters ##
out <- "/path/to/scds/outdir/"
tenX_matrix <- "/path/to/counts/matrix/dir/"

dir.create(out, recursive = TRUE)
print(paste0("Using the following counts directory: ", tenX_matrix))



### Read in data as an sce object ###
counts <- Read10X(tenX_matrix, gene.column = 1) ## or Read10X_h5 if using h5 file as␣
↪input
sce <- SingleCellExperiment(list(counts=counts))


## Calculate doublet ratio ###
doublet_ratio <- ncol(sce)/1000*0.008


### Calculate Singlets and Doublets ###
sce <- scDblFinder(sce, dbr=doublet_ratio)
```

(continues on next page)

(continued from previous page)

```
### Make a dataframe of the results ###
results <- data.frame("Barcode" = rownames(colData(sce)), "scDblFinder_DropletType" =␣
↪sce$scDblFinder.class, "scDblFinder_Score" = sce$scDblFinder.score)


write_delim(results, path = paste0(out,"/scDblFinder_doublets_singlets.tsv"), delim =␣
↪"\t")

### Calculate number of doublets and singlets ###
summary <- as.data.frame(table(results$scDblFinder_DropletType))
colnames(summary) <- c("Classification", "Droplet N")
write_delim(summary, paste0(out,"/scDblFinder_doublet_summary.tsv"), "\t")
```

2.19.3 ScDblFinder Results and Interpretation

After running the ScDblFinder with the wrapper script or manually you should have two files in the `$SCDBLFINDER_OUTDIR`:

```
.
├── scDblFinder_doublets_singlets.tsv
└── scDblFinder_doublet_summary.tsv
```

Here's a more detaild description of each of those files:

- `scDblFinder_doublet_summary.tsv`
  - A sumamry of the number of singlets and doublets predicted by ScDblFinder.

Classification	Droplet N
doublet	3323
singlet	17659

    * To check whether the numbe of doublets identified by ScDblFinder is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.

- `scDblFinder_doublets_singlets.tsv`
  - The per-barcode singlet and doublet classification from ScDblFinder.

Barcode	scDblFinder_DropletType	scDblFinder_Score
AAACCTGAGATAGCAT-1	singlet	0.0033526041079312563
AAACCTGAGCAGCGTA-1	doublet	0.9937564134597778
AAACCTGAGCGATGAC-1	singlet	5.045032594352961e-
AAACCTGAGCGTAGTG-1	singlet	0.007504515815526247
AAACCTGAGGAGTTTA-1	singlet	0.00835108570754528
AAACCTGAGGCTCATT-1	singlet	0.028838597238063812
AAACCTGAGGGCACTA-1	doublet	0.9985504746437073
AAACCTGAGTAATCCC-1	singlet	0.005869860760867596
…	…	…

2.19.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.19.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as ScDblFinder.

2.20 Scds

Scds is a transcription-based doublet detection software that uses two different methods to detect doublets - cxds and bcds. The cxds method uses marker genes that are not co-expressed to identify droplets that are likely doublets. bcds simulates doublet by adding droplet transcriptomes together and then uses variable genes to identify the probability a droplet is a doublet with a binary classification algorithm. We typically use the combined score of these two methods but they can be use separately as well. We have provided a wrapper script that takes common arguments for Scds and we also provide an example script that you can run manually in R if you prefer.

2.20.1 Data

This is the data that you will need to have prepare to run Scds:

Required

- A counts matrix (`$COUNTS`)
  - The directory path containing your cellranger counts matrix files (directory containing `barcodes.tsv`, `genes.tsv` and `matrix.mtx` **or** `barcodes.tsv.gz`, `features.tsv.gz` and `matrix.mtx.gz`)

    or
  - h5 file (`filtered_feature_bc_matrix.h5`)
    - If you don't have your data in this format, you can run Scds manually in R and load the data in using a method of your choosing.
- Output directory (`$SCDS_OUTDIR`)
  - If you don't provide an `$SCDS_OUTDIR`, the results will be written to the present working directory.

2.20.2 Run Scds

You can either run Scds with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

With Wrapper Script

Run in R

```
singularity exec Demuxafy.sif scds.R -o $SCDS_OUTDIR -t $COUNTS
```

First, you will have to start R. We have built R and all the required software to run Scds into the singularity image so you can run it directly from the image.

```
singularity exec Demuxafy.sif R
```

That will open R in your terminal. Next, you can load all the libraries and run Scds.

```
.libPaths("/usr/local/lib/R/site-library") ### This is required so that R uses the␣
↪libraries loaded in the image and not any local libraries
library(dplyr)
library(tidyr)
library(tidyverse)
library(scds)
library(Seurat)
library(SingleCellExperiment)

## Set up variables and parameters ##
out <- "/path/to/scds/outdir/"
tenX_matrix <- "/path/to/counts/matrix/dir/"

## Read in data
counts <- Read10X(as.character(tenX_matrix), gene.column = 1) ## or Read10X_h5 if␣
↪using h5 file as input

## Account for possibility that not just single cell data
if (is.list(counts)){
  sce <- SingleCellExperiment(list(counts=counts[[grep("Gene", names(counts))]]))
} else {
  sce <- SingleCellExperiment(list(counts=counts))
}

## Annotate doublet using binary classification based doublet scoring:
sce = bcds(sce, retRes = TRUE, estNdbl=TRUE)

## Annotate doublet using co-expression based doublet scoring:
try({
    sce = cxds(sce, retRes = TRUE, estNdbl=TRUE)
})

### If cxds worked, run hybrid, otherwise use bcds annotations
if ("cxds_score" %in% colnames(colData(sce))) {
    ## Combine both annotations into a hybrid annotation
    sce = cxds_bcds_hybrid(sce, estNdbl=TRUE)
    Doublets <- as.data.frame(cbind(rownames(colData(sce)), colData(sce)$hybrid_score,
↪ colData(sce)$hybrid_call))
} else {
    print("this pool failed cxds so results are just the bcds calls")
    Doublets <- as.data.frame(cbind(rownames(colData(sce)), colData(sce)$bcds_score,␣
↪colData(sce)$bcds_call))
}

## Doublet scores are now available via colData:
colnames(Doublets) <- c("Barcode","scds_score","scds_DropletType")
Doublets$scds_DropletType <- gsub("FALSE","singlet",Doublets$scds_DropletType)
Doublets$scds_DropletType <- gsub("TRUE","doublet",Doublets$scds_DropletType)

message("writing output")
write_delim(Doublets, paste0(out,"/scds_doublets_singlets.tsv"), "\t")
```

(continues on next page)

(continued from previous page)

```
summary <- as.data.frame(table(Doublets$scds_DropletType))
colnames(summary) <- c("Classification", "Droplet N")
write_delim(summary, paste0(out,"/scds_doublet_summary.tsv"), "\t")
```

2.20.3 Scds Results and Interpretation

After running the Scds with the wrapper script or manually you should have two files in the `$SCDS_OUTDIR`:

```
.
├── scds_doublets_singlets.tsv
└── scds_doublet_summary.tsv
```

- `scds_doublet_summary.tsv`
  - A summary of the number of singlets and doublets predicted by Scds.

Classification	Droplet N
doublet	2771
singlet	18211

    * To check whether the number of doublets identified by Scds is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.

- `scds_doublets_singlets.tsv`
  - The per-barcode singlet and doublet classification from Scds.

Barcode	scds_score	scds_DropletType
AAACCTGAGATAGCAT-1	0.116344358493288	singlet
AAACCTGAGCAGCGTA-1	0.539856378453988	singlet
AAACCTGAGCGATGAC-1	0.0237184380134577	singlet
AAACCTGAGCGTAGTG-1	0.163695865366576	singlet
AAACCTGAGGAGTTTA-1	0.11591462421927	singlet
AAACCTGAGGCTCATT-1	0.0479944175570073	singlet
AAACCTGAGGGCACTA-1	0.374426050641161	singlet
AAACCTGAGTAATCCC-1	0.247842972104563	singlet
...	...	...

2.20.4 Merging Results with Other Software Retults

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.20.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as scds.

2.21 Scrublet

Scrublet is a transcription-based doublet detecting software. We have provided a wrapper script that enables Scrublet to be easily run from the command line but we also provide example code so that users can run manually as well depending on their data.

2.21.1 Data

This is the data that you will need to have prepare to run Scrublet:

Required

- A counts matrix (`$COUNTS`)
  - Scrublet expects counts to be in the cellranger output format either as
    - h5 file (`filtered_feature_bc_matrix.h5`)

      or

    - matrix directory (directory containing `barcodes.tsv`, `genes.tsv` and `matrix.mtx` **or** `barcodes.tsv.gz`, `features.tsv.gz` and `matrix.mtx.gz`)
    - If you don't have your data in this format, you can run Scrublet manually in python and load the data in using a method of your choosing.

Optional

- Output directory (`$SCRUBLET_OUTDIR`)
  - If you don't provide an `$SCRUBLET_OUTDIR`, the results will be written to the present working directory.

2.21.2 Run Scrublet

You can either run Scrublet with the wrapper script we have provided or you can run it manually if you would prefer to alter more parameters.

Note

It is a good idea to try multiple different percentile variable numbers. We typically try, 80, 85, 90 and 95. Then we choose the one that has the best defined bimodal distribution based on the `doublet_score_histogram.png` (see *Scrublet Results and Interpretation* for details).

With Wrapper Script

Run in python

```
singularity exec Demuxafy.sif Scrublet.py -m $COUNTS -o $SCRUBLET_OUTDIR
```

To see all the parameters that this wrapper script will accept, run:

```
singularity exec Demuxafy.sif Scrublet.py -h



usage: Scrublet.py [-h] -m COUNTS_MATRIX [-b BARCODES] [-r SIM_DOUBLET_RATIO] [-c MIN_
↪COUNTS] [-e MIN_CELLS] [-v MIN_GENE_VARIABILITY_PCTL] [-p N_PRIN_COMPS] [-t␣
↪SCRUBLET_DOUBLET_THRESHOLD] [-o OUTDIR]

wrapper for scrublet for doublet detection of transcriptomic data.

optional arguments:
  -h, --help            show this help message and exit
  -m COUNTS_MATRIX, --counts_matrix COUNTS_MATRIX
                        cell ranger counts matrix directory containing matrix files␣
↪or full path to matrix.mtx. Can also also provide the 10x h5.
  -b BARCODES, --barcodes BARCODES
                        barcodes.tsv or barcodes.tsv.gz from cellranger
  -r SIM_DOUBLET_RATIO, --sim_doublet_ratio SIM_DOUBLET_RATIO
                        Number of doublets to simulate relative to the number of␣
↪observed transcriptomes.
  -c MIN_COUNTS, --min_counts MIN_COUNTS
                        Used for gene filtering prior to PCA. Genes expressed at␣
↪fewer than min_counts in fewer than min_cells are excluded.
  -e MIN_CELLS, --min_cells MIN_CELLS
                        Used for gene filtering prior to PCA. Genes expressed at␣
↪fewer than min_counts in fewer than are excluded.
  -v MIN_GENE_VARIABILITY_PCTL, --min_gene_variability_pctl MIN_GENE_VARIABILITY_PCTL
                        Used for gene filtering prior to PCA. Keep the most highly␣
↪variable genes in the top min_gene_variability_pctl percentile), as measured by the␣
↪v-statistic [Klein et al., Cell 2015].
  -p N_PRIN_COMPS, --n_prin_comps N_PRIN_COMPS
                        Number of principal components used to embed the␣
↪transcriptomes priorto k-nearest-neighbor graph construction.
  -t SCRUBLET_DOUBLET_THRESHOLD, --scrublet_doublet_threshold SCRUBLET_DOUBLET_
↪THRESHOLD
                        Manually Set the scrublet doublet threshold location. For␣
↪running a second time if scrublet incorrectly places the threshold the first time
  -o OUTDIR, --outdir OUTDIR
                        The output directory
```

To run Scrublet manually, first start python from the singularity image (all the required software have been provided in the image)

```
singularity exec Demuxafy.sif python
```

Now, python will open in your terminal and you can run the Scrublet code. Here is an example:

```
import sys
import os
import scrublet as scr
import scipy.io
import matplotlib
matplotlib.use('AGG')
```

(continues on next page)

(continued from previous page)

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import umap
import numba
import numba.typed

# Get path of mods directory from current script directory
mods_path = "/opt/Demultiplexing_Doublet_Detecting_Docs/mods"
sys.path.append(mods_path)
import read10x

## Set up parameters and variables ##
counts_matrix_dir = "/path/to/counts/matrix/dir/"
outdir = "/path/to/doublet/detection/outdir"

if not os.path.isdir(outdir):
  os.mkdir(outdir)


plt.rc('font', size=14)
plt.rcParams['pdf.fonttype'] = 42

## Basic run with scrublet
counts_matrix = read10x.import_cellranger_mtx(counts_matrix_dir) ## or scanpy.read_
↪10x_h5(counts_matrix_dir)

try:
  barcodes_df = read10x.read_barcodes(counts_matrix_dir + "/barcodes.tsv.gz")
except:
  try:
    barcodes_df = read10x.read_barcodes(counts_matrix_dir + "/barcodes.tsv")
  except:
    ␣
↪print("No barcode file in provided counts matrix directory. Please double check the directory or pr



dbl_rate = counts_matrix.shape[0]/1000 * 0.008
print('Counts matrix shape: {} rows, {} columns'.format(counts_matrix.shape[0],␣
↪counts_matrix.shape[1]))
scrub = scr.Scrublet(counts_matrix, expected_doublet_rate=dbl_rate, sim_doublet_ratio␣
↪= 2)
doublet_scores, predicted_doublets = scrub.scrub_doublets(min_counts=3,
                                                          min_cells=3,
                                                          min_gene_variability_
↪pctl=85,
                                                          n_prin_comps=30)


### Plotting and saving
scrub.plot_histogram();
plt.savefig(os.path.join(outdir,'doublet_score_histogram.png'))
print('Running UMAP...')
scrub.set_embedding('UMAP', scr.get_umap(scrub.manifold_obs_, 10, min_dist=0.3))
print('Done.')
scrub.plot_embedding('UMAP', order_points=True);
```

(continues on next page)

(continued from previous page)

```
plt.savefig(os.path.join(outdir,'UMAP.png'))

results = pd.Series(scrub.predicted_doublets_, name="scrublet_DropletType")
scores = pd.Series(scrub.doublet_scores_obs_, name="scrublet_Scores")
dataframe = pd.concat([barcodes_df, results, scores], axis=1)
dataframe.scrublet_DropletType = dataframe.scrublet_DropletType.replace(True,␣
↪"doublet")
dataframe.scrublet_DropletType = dataframe.scrublet_DropletType.replace(False,␣
↪"singlet")

dataframe.to_csv(os.path.join(outdir,'scrublet_results.tsv'), sep = "\t", index =␣
↪False)


### Make summary of singlets and doublets and write to file ###
summary = pd.DataFrame(dataframe.scrublet_DropletType.value_counts())
summary.index.name = 'Classification'
summary.reset_index(inplace=True)
summary = summary.rename({'scrublet_DropletType': 'Droplet N'}, axis=1)

summary.to_csv(os.path.join(outdir,'scrublet_summary.tsv'), sep = "\t", index = False)
```

2.21.3 Scrublet Results and Interpretation

After running the Scrublet, you will have four files in the `$SCRUBLET_OUTDIR`:

```
.
├── doublet_score_histogram.png
├── scrublet_results.tsv
├── scrublet_summary.tsv
└── UMAP.png
```

We have found these to be the most helpful:

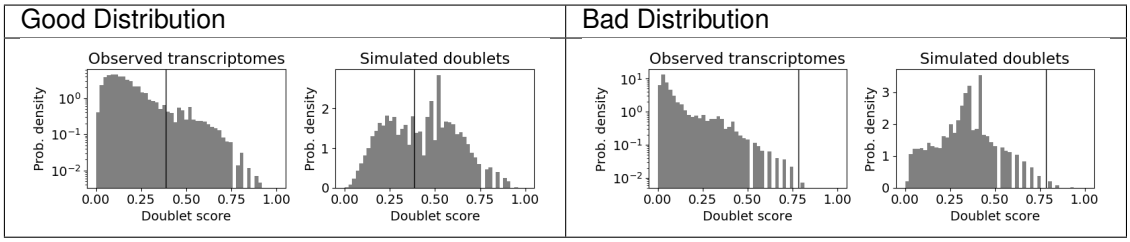
- `scrublet_summary.tsv`
  - A summary of the number of singlets and doublets predicted by Scrublet.

scrublet_DropletType	Droplet N
doublet	1851
singlet	19131

  - To check whether the number of doublets identified by Scrublet is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.
- `scrublet_results.tsv`

Barcode	scrublet_DropletType	scrublet_Scores
AAACCTGAGATAGCAT-1	singlet	0.0545
AAACCTGAGCAGCGTA-1	singlet	0.1179
AAACCTGAGCGATGAC-1	singlet	0.1356
AAACCTGAGCGTAGTG-1	singlet	0.0844
AAACCTGAGGAGTTTA-1	singlet	0.0958
AAACCTGAGGCTCATT-1	singlet	0.1329
AAACCTGAGGGCACTA-1	doublet	0.4474
...	...	...

- `doublet_score_histogram.png`
  - This is the method that Scrublet uses to identify doublets - it assumes a bimodal distribution of doublet scores. Those droplets with lower scores should be singlets and those with higher scores should be doublets. It identifies the correct threshold by identifying the minimum of the bimodal distribution of simulated doublets (right).
  - However, sometimes there is not a good bimodal distribution and sometimes you will have to set the threshold manually.
  - Here is an example of a good distribution (left) and a bad distribution (left)



    - In the case of the left sample, we would rerun with different parameters to try to get a better distribution and possibly manually set the threshold to ~0.2 depending on the results. In the event that we can't achieve a clear bimodal distribution, we don't use scrublet for doublet detecting.

2.21.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.21.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as Scrublet.

2.22 Solo

Solo is a transcription-based doublet detecting software that was one of the better transcription-based doublet detecting softwares that we tested (CITATION).

2.22.1 Data

This is the data that you will need to have prepare to run Solo:

Required

- Parameter json file (`$JSON`)
  - Solo has provided an `example file` that we have found to work well for most of our data.
- Counts (`$COUNTS`)
  - This can be a h5ad file, loom file, or 10x counts matrix directory (containing `barcodes.tsv`, `genes.tsv` and `matrix.mtx` **or** `barcodes.tsv.gz`, `features.tsv.gz` and `matrix.mtx.gz`)
- Output directory (`$SOLO_OUTDIR`)

Optional

- Expected number of doublets (`$N_DOUB`)

2.22.2 Run Solo

```
singularity exec Demuxafy.sif solo -o $SOLO_OUTDIR -e $N_DOUB -j $JSON -d $COUNTS
```

Solo also has additional parameters that can be seen with:

```
singularity exec Demuxafy.sif solo -h

usage: solo [-h] -j MODEL_JSON_FILE -d DATA_PATH
          [--set-reproducible-seed REPRODUCIBLE_SEED]
          [--doublet-depth DOUBLET_DEPTH] [-g] [-a] [-o OUT_DIR]
          [-r DOUBLET_RATIO] [-s SEED] [-e EXPECTED_NUMBER_OF_DOUBLETS] [-p]
          [-recalibrate_scores] [--version]

optional arguments:
  -h, --help            show this help message and exit
  -j MODEL_JSON_FILE    json file to pass VAE parameters (default: None)
  -d DATA_PATH          path to h5ad, loom, or 10x mtx dir cell by genes
                        counts (default: None)
  --set-reproducible-seed REPRODUCIBLE_SEED
                        Reproducible seed, give an int to set seed (default:
                        None)
  --doublet-depth DOUBLET_DEPTH
                        Depth multiplier for a doublet relative to the average
                        of its constituents (default: 2.0)
  -g                    Run on GPU (default: True)
```

(continues on next page)

(continued from previous page)

```
  -a                    output modified anndata object with solo scores Only
                        works for anndata (default: False)
  -o OUT_DIR
  -r DOUBLET_RATIO      Ratio of doublets to true cells (default: 2)
  -s SEED               Path to previous solo output directory. Seed VAE
                        models with previously trained solo model. Directory
                        structure is assumed to be the same as solo output
                        directory structure. should at least have a vae.pt a
                        pickled object of vae weights and a latent.npy an
                        np.ndarray of the latents of your cells. (default:
                        None)
  -e EXPECTED_NUMBER_OF_DOUBLETS
                        Experimentally expected number of doublets (default:
                        None)
  -p                    Plot outputs for solo (default: False)
  -recalibrate_scores   Recalibrate doublet scores (not recommended anymore)
                        (default: False)
  --version             Get version of solo-sc (default: False)
```

If Solo runs correctly, you should have the following files and directory structure in your `$SOLO_OUTDIR`:

```
.
├── classifier
│   ├── attr.pkl
│   ├── model_params.pt
│   └── var_names.csv
├── is_doublet.csv
├── is_doublet.npy
├── is_doublet_sim.npy
├── latent.npy
├── logit_scores.csv
├── logit_scores.npy
├── logit_scores_sim.npy
├── no_updates_softmax_scores.csv
├── no_updates_softmax_scores.npy
├── no_updates_softmax_scores_sim.npy
├── preds.csv
├── preds.npy
├── smoothed_preds.npy
├── softmax_scores.csv
├── softmax_scores.npy
└── vae
    ├── attr.pkl
    ├── model_params.pt
    └── var_names.csv
```

Solo Summary

We have provided a script that will summarize the number of droplets classified as doublets and singlets by Solo and write it to the `$SOLO_OUTDIR`. This script also combines some of the Solo outputs into a single file that can be more easily used for downstream analyses. You can run this to get a fast and easy summary of your results with:

```
singularity exec Demuxafy.sif solo_summary.py -b $BARCODES -s $SOLO_OUTDIR
```

If successful, you should have two new files in your `$SOLO_OUTDIR`:

```
.
├── classifier
│   ├── attr.pkl
│   ├── model_params.pt
│   └── var_names.csv
├── is_doublet.csv
├── is_doublet.npy
├── is_doublet_sim.npy
├── latent.npy
├── logit_scores.csv
├── logit_scores.npy
├── logit_scores_sim.npy
├── no_updates_softmax_scores.csv
├── no_updates_softmax_scores.npy
├── no_updates_softmax_scores_sim.npy
├── preds.csv
├── preds.npy
├── smoothed_preds.npy
├── softmax_scores.csv
├── softmax_scores.npy
├── solo_results.tsv
├── solo_summary.tsv
└── vae
    ├── attr.pkl
    ├── model_params.pt
    └── var_names.csv
```

2.22.3 Solo Results and Interpretation

Solo puts most of the results in multiple separate files. However, the wrapper script and the example code has some steps to combine these results together into a single file, which will likely be the most informative output.

- `solo_summary.tsv`
  - A summary of the number of singlets and doublets predicted by Solo.

Classification	Droplet N
singlet	17461
doublet	3521

  - To check whether the number of doublets identified by Solo is consistent with the expected doublet rate expected based on the number of droplets that you captured, you can use our Expected Doublet Estimation Calculator.
- `solo_results.tsv`
  - The per-barcode singlet and doublet classification from Solo.

Barcode	solo_DropletType	solo_DropletScore
AAACCTGAGATAGCAT-1	singlet	-8.442187
AAACCTGAGCAGCGTA-1	singlet	-2.8096201
AAACCTGAGCGATGAC-1	singlet	-2.8949203
AAACCTGAGCGTAGTG-1	singlet	-5.928284
AAACCTGAGGAGTTTA-1	doublet	0.2749935
AAACCTGAGGCTCATT-1	singlet	-5.2726507
AAACCTGAGGGCACTA-1	singlet	-0.65760195
AAACCTGAGTAATCCC-1	singlet	-3.5948637
...	...	...

2.22.4 Merging Results with Other Software Results

We have provided a script that will help merge and summarize the results from multiple softwares together. See *Combine Results*.

2.22.5 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE) as well as solo.

2.23 Combining Results

After you have run each of the Demultiplexing and Doublet Detecting softwares you would like, it is helpful to convert them to similar nomenclature and combine the results into a single dataframe. In addition, we have found it helpful to generate summaries of each of the combinations of softwares identified. To help streamline this process, we have provided a script that will easily integrate all the softwares you have run into a single dataframe and can do the following:

1. Generate a dataframe that has all the software assignments per droplet in the pool

- A tab-separated dataframe with the droplet singlet-doublet classification and the individual assignment (for demultiplexing softwares) per droplet

1. Generate an upset plot that shows the droplet classificaitons by each software and the final classifications

2. Generate a droplet type summary file

- Provides the number of droplets classified for each combination of droplet classifications by each software

1. Generate demultiplexing individual assignment summary file

- Provides the number of droplets classified for each combination of individual assignment droplet classifications by each software

1. If individuals have not been assigned to each cluster for reference-free demultiplexing softwares, will create a common assignment across all demultiplexing softwares for easy comparison

2. Combined final droplet assignment from all softwares included

- Uses one of four intersectional methods to combine software assignments together into a single combined assignment per barcode

and to generate a summary file for all the software combinations and if you ran demultiplexing softwares, it will also generate a demultiplexing summary file for the individual and cluster assignments from the demultiplexing softwares.

2.23.1 Data

In order to use our script to combine the results from the various demultiplexing and doublet detecting softwares, you need the following:

Required

- Output directory (`$OUTDIR`)
- Path to results of each of the softwares you would like to merge into a single dataframe.
  - You need to provide the path to at least one software result, otherwise, it will not run.

2.23.2 Merging Results with Combine_Results.R

The script has multiple options to provide the paths to each of the software results you would like to run. To see each of the options, simply run:

```
singularity exec Demuxafy.sif Combine_Results.R -h
```

Providing the possible parameter options:

```
usage: /directflow/SCCGGroupShare/projects/DrewNeavin/Demultiplex_Benchmark/
↪Demultiplexing_Doublet_Detecting_Docs/scripts/Combine_Results.R
      [-h] -o OUT [-d DEMUXLET] [-f FREEMUXLET] [-g FREEMUXLET_ASSIGNMENTS]
      [-a FREEMUXLET_CORRELATION_LIMIT] [-s SCSPLIT] [-w SCSPLIT_ASSIGNMENTS]
      [-j SCSPLIT_CORRELATION_LIMIT] [-u SOUPORCELL]
      [-x SOUPORCELL_ASSIGNMENTS] [-k SOUPORCELL_CORRELATION_LIMIT]
      [-v VIREO] [-e DOUBLETDECON] [-t DOUBLETDETECTION] [-i DOUBLETFINDER]
      [-n SCDBLFINDER] [-c SCDS] [-r SCRUBLET] [-l SOLO] [-b REF]
      [-p PCT_AGREEMENT] [-m METHOD]

optional arguments:
  -h, --help            show this help message and exit
  -o OUT, --out OUT     The file where results will be saved
  -d DEMUXLET, --demuxlet DEMUXLET
                        Path to demuxlet results. Only use this option if you
                        want to include the demuxlet results.
  -f FREEMUXLET, --freemuxlet FREEMUXLET
                        Path to freemuxlet results. Only use this option if
                        you want to include the freemuxlet results.
  -g FREEMUXLET_ASSIGNMENTS, --freemuxlet_assignments FREEMUXLET_ASSIGNMENTS
                        Path to freemuxlet cluster-to-individual assignments.
                        Only use this option if have used reference SNP
                        genotypes to assign individuals to clusters for the
                        freemuxlet results.
  -a FREEMUXLET_CORRELATION_LIMIT, --freemuxlet_correlation_limit FREEMUXLET_
↪CORRELATION_LIMIT
                        The minimum correlation between the cluster and the
                        individual SNP genotypes which should be considered as
                        a valid assignment. If you want no limit, use 0.
```

(continues on next page)

(continued from previous page)

```
                        Default is 0.7.
  -s SCSPLIT, --scSplit SCSPLIT
                        Path to scSplit results. Only use this option if you
                        want to include the scSplit results.
  -w SCSPLIT_ASSIGNMENTS, --scSplit_assignments SCSPLIT_ASSIGNMENTS
                        Path to scSplit cluster-to-individual assignments.
                        Only use this option if you have used reference SNP
                        genotypes to assign individuals to clusters for the
                        scSplit results.
  -j SCSPLIT_CORRELATION_LIMIT, --scSplit_correlation_limit SCSPLIT_CORRELATION_LIMIT
                        The minimum correlation between the cluster and the
                        individual SNP genotypes which should be considered as
                        a valid assignment. If you want no limit, use 0.
                        Default is 0.7.
  -u SOUPORCELL, --souporcell SOUPORCELL
                        Path to souporcell results. Only use this option if
                        you want to include the souporcell results.
  -x SOUPORCELL_ASSIGNMENTS, --souporcell_assignments SOUPORCELL_ASSIGNMENTS
                        Path to souporcell cluster-to-individual assignments.
                        Only use this option if you have used reference SNP
                        genotypes to assign individuals to clusters for the
                        souporcell results.
  -k SOUPORCELL_CORRELATION_LIMIT, --souporcell_correlation_limit SOUPORCELL_
↪CORRELATION_LIMIT
                        The minimum correlation between the cluster and the
                        individual SNP genotypes which should be considered as
                        a valid assignment. If you want no limit, use 0.
                        Default is 0.7.
  -v VIREO, --vireo VIREO
                        Path to vireo results. Only use this option if you
                        want to include the vireo results.
  -e DOUBLETDECON, --DoubletDecon DOUBLETDECON
                        Path to DoubletDecon results. Only use this option if
                        you want to include the DoubletDecon results.
  -t DOUBLETDETECTION, --DoubletDetection DOUBLETDETECTION
                        Path to DoubletDetection results. Only use this option
                        if you want to include the DoubletDetection results.
  -i DOUBLETFINDER, --DoubletFinder DOUBLETFINDER
                        Path to DoubletFinder results. Only use this option if
                        you want to include the DoubletFinder results.
  -n SCDBLFINDER, --scDblFinder SCDBLFINDER
                        Path to scDblFinder results. Only use this option if
                        you want to include the scDblFinder results.
  -c SCDS, --scds SCDS  Path to scds results. Only use this option if you want
                        to include the scds results.
  -r SCRUBLET, --scrublet SCRUBLET
                        Path to scrublet results. Only use this option if you
                        want to include the scrublet results.
  -l SOLO, --solo SOLO  Path to solo results. Only use this option if you want
                        to include the solo results.
  -b REF, --ref REF     Which demultiplexing software to use as a reference
                        for individuals when you do not have assignment key
                        for all demultiplexing method. Options are 'Demuxlet',
                        'Freemuxlet', 'scSplit', 'Souporcell' and 'Vireo'. If
                        blank when assignment keys are missing, default
                        softwares to use if present are Vireo, then Demuxlet,
                        then Freemuxlet, then Souporcell, then scSplit.
```

(continues on next page)

(continued from previous page)

```
-p PCT_AGREEMENT, --pct_agreement PCT_AGREEMENT
                      The proportion of a cluster that match the 'ref'
                      assignment to assign that cluster the individual
                      assignment from the reference. Can be between 0.5 and
                      1. Default is 0.9.
-m METHOD, --method METHOD
                      Combination method. Options are 'MajoritySinglet',
                      'AtLeastHalfSinglet', 'AnySinglet' or 'AnyDoublet'. We
                      have found that 'MajoritySinglet' provides the most
                      accurate results in most situations and therefore
                      recommend this method. See https://demultiplexing-
                      doublet-detecting-
                      docs.readthedocs.io/en/latest/CombineResults.html for
                      detailed explanation of each intersectional method.
                      Leave blank if you just want all the softwares to be
                      merged into a single dataframe.
```

Combination Methods - Additional Information

There are four options for making combined droplet type (singlet or doublet) and individual assignment from the softwares used:

- MajoritySinglet
  - If more than half of the softwares identify a droplet as a singlet, it is classified as a singlet.
  - If more than half the demultiplexing softwares identify the same indivdual, that assignment is used for the droplet.
  - We have found
- AtLeastHalfSinglet
  - If at least half of the softwares identify a droplet as a singlet, it is classified as a singlet.
  - If at least half the demultiplexing softwares identify the same indivdual, that assignment is used for the droplet.
- AnySinglet
  - If this droplet is identified as a singlet by any software, the droplet is classified as a singlet.
  - In other words, a doublet is only called if all softwares identified that droplet as a doublet.
- AnyDoublet
  - A droplet is classified as a singlet only if all softwares identify it as a singlet.
  - In other words, a doublet is called if any software identifies that droplet as a doublet.

An example command that combines *Demuxlet* results, *Souporcell* results, *Solo* results and *Scds* results would look like this: There are a two different options for using this script:

Combine Results + Joint Droplet Calls

Combine Results

The first option is to select a method to make joint calls on the individual assignment and singlet-doublet droplet types using the softwares included.

```
singularity exec Demuxafy.sif Combine_Results.R \
  -o $OUTDIR/combined_results.tsv \
  --demuxlet $DEMUXLET_OUTDIR \
  --souporcell $SOUPORCELL_OUTDIR \
  --solo $SOLO_OUTDIR \
  --scds $SCDS_OUTDIR \
  --method "MajoritySinglet"
```

The other option is to just combine the results together without instersectional joint calls on the assignment and droplet type for each droplet.

```
singularity exec Demuxafy.sif Combine_Results.R \
  -o $OUTDIR/combined_results.tsv \
  --demuxlet $DEMUXLET_OUTDIR \
  --souporcell $SOUPORCELL_OUTDIR \
  --solo $SOLO_OUTDIR \
  --scds $SCDS_OUTDIR
```

Note

The path to the directories will work if the file names are the expected file names based on the example tutorials. However, if you used a different file naming convention or changed the names, you can also provide the full path to the exact file for each software.

2.23.3 Results and Interpretation

After running the `Combine_Results.R` script, you should have two, three or four files depending on if you used demultiplexing softwares and if you used joint droplet calling. Here, we show the results for the above example that also provides combined calls with the "MajoritySinglet" calls.

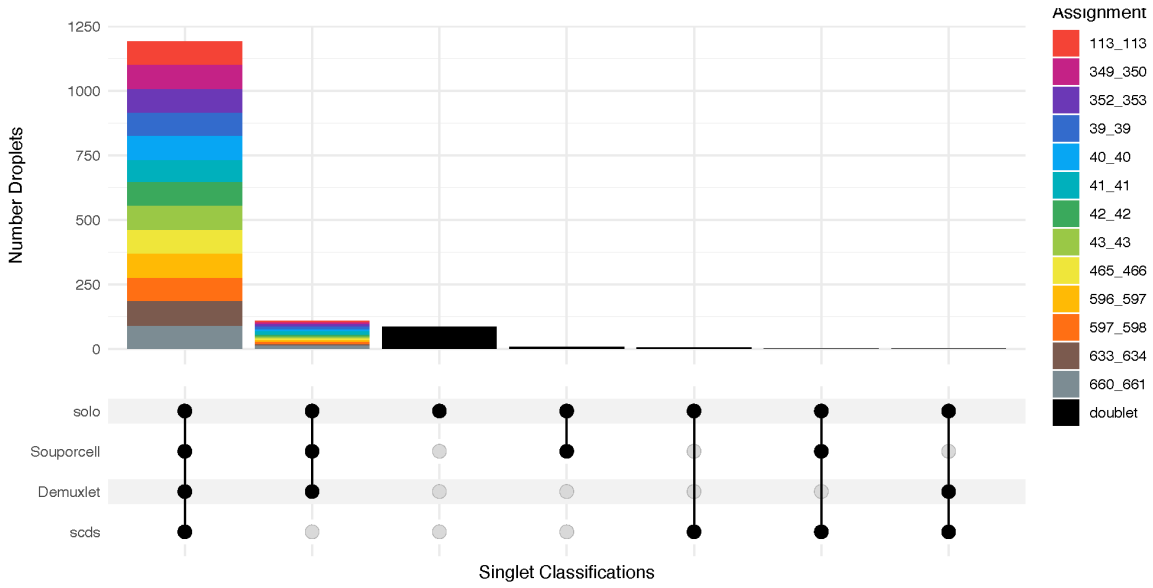
```
.
├── combined_results_demultiplexing_summary.tsv
├── combined_resultsSinglets_upset.pdf
├── combined_results_summary.tsv
├── combined_results.tsv
└── combined_results_w_combined_assignments.tsv
```

Note

- You will only have the `combined_results_demultiplexing_summary.tsv` file if you included demultiplexing softwares.
- And you will only have the `combined_results_w_combined_assignments.tsv` file if you ran it with `--method`

Here's a deeper look at the contents of each of these results:

- `combined_resultsSinglets_upset.pdf`
  - This is an upset figure of the droplets which are colored by their finall individual or doublet classification.
  - A filled circle indicates the that those droplets are classified as singlets by that method while empty circles indicate a doublet classification by that software



- `combined_results.tsv`
  - Has the selected results combined; only including key columns.

Bar-code	De-muxlet_DropletType	De-mux_Individual_Assigned	Soupor-cell_Individual_Assigned	Soupor-cell_Status	Soupor-cell_DropletType	scds_scores	scds_DropletType	solo_DropletType	DropletScore
AAACCTG...1	singlet	41_41	41_41	6	singlet	0.11634358493288	singlet	singlet	-8.442187
AAACCTG...1	singlet	465_466	465_466	11	singlet	0.53985637845398	singlet	singlet	-2.8096201
AAACCTG...1	singlet	113_113	113_113	5	singlet	0.02371843801345	singlet	singlet	-2.8949203
AAACCTG...1	singlet	349_350	349_350	3	singlet	0.16369586536657	singlet	singlet	-5.928284
AAACCTG...1	singlet	632_633	632_633	7	singlet	0.11591462421927	singlet	doublet	0.2749935
AAACCTG...1	singlet	39_39	39_39	12	singlet	0.04799417557007	singlet	singlet	-5.2726507
AAACCTG...1	singlet	465_466	465_466	11	singlet	0.37442650641161	singlet	singlet	-0.65760195
AAACCTG...1	singlet	660_661	660_661	4	singlet	0.24784297210456	singlet	singlet	-3.5948637
AAACCTG...1	doublet	unassigned	unassigned	unassigned	unassigned	0.34299838528193	singlet	singlet	-0.50507957
…	…	…	…	…	…	…	…	…	…

- `combined_results_summary.tsv`
  - The number of each of the combinations of the software cell type classifications

Demuxlet_DropletType	Souporcell_DropletType	scds_DropletType	solo_DropletType	N
singlet	singlet	singlet	singlet	16193
doublet	doublet	doublet	doublet	1714
singlet	singlet	singlet	doublet	947
doublet	doublet	singlet	singlet	468
singlet	singlet	doublet	singlet	392
singlet	singlet	doublet	doublet	345
doublet	doublet	singlet	doublet	335
doublet	singlet	singlet	singlet	171
doublet	doublet	doublet	singlet	169
doublet	singlet	doublet	doublet	114
doublet	singlet	singlet	doublet	44
doublet	singlet	doublet	singlet	18
singlet	doublet	singlet	singlet	17
singlet	unassigned	singlet	singlet	13
doublet	unassigned	singlet	singlet	11
singlet	doublet	doublet	doublet	9
singlet	doublet	singlet	doublet	6
singlet	doublet	doublet	singlet	5
doublet	unassigned	singlet	doublet	4
doublet	unassigned	doublet	doublet	3
doublet	unassigned	doublet	singlet	2
unassigned	unassigned	singlet	singlet	2

- `combined_results_demultiplexing_summary.tsv`
  - Summary of the number of each of the combination of classifications by demultiplexing software:

Demuxlet_Individual_Assignment	Souporcell_Individual_Assignment	N
doublet	doublet	2706
352_353	352_353	1603
43_43	43_43	1547
597_598	597_598	1510
349_350	349_350	1450
42_42	42_42	1417
660_661	660_661	1358
113_113	113_113	1333
39_39	39_39	1289
…	…	…

- `combined_results_w_combined_assignments.tsv`
  - Dataframe combining all the software results together + combined assignment based on selected method:

Barcode	Demuxlet_DropletType	Demuxlet_Individual_Assignment	Souporcell_Cluster	Souporcell_Individual_Assignment	Souporcell_DropletType	scds_score	scds_DropletType	DoubletDetection_DropletType	DoubletDetection_DoubletScore	MajoritySinglet_DropletType	MajoritySinglet_Individual_Assignment
AAACCTGAGATAGCAT-1	singlet	41_41	6	41_41	singlet	0.116344358493288	singlet	singlet	-8.442187	singlet	41_41
AAACCTGAGCAGCGTA-1	singlet	465_466	11	465_466	singlet	0.539856378453988	singlet	singlet	-2.8096201	singlet	465_466
AAACCTGAGCGATGAC-1	singlet	113_113	5	113_113	singlet	0.0237184380134577	singlet	singlet	-2.8949203	singlet	113_113
AAACCTGAGCGTAGTG-1	singlet	349_350	3	349_350	singlet	0.163695865366576	singlet	singlet	-5.928284	singlet	349_350
AAACCTGAGGAGTTGA-1	singlet	632_633	7	632_633	singlet	0.115914624219027	singlet	doublet	0.2749935	singlet	632_633
AAACCTGAGGCTCATT-1	singlet	39_39	12	39_39	singlet	0.0479944175570073	singlet	singlet	-5.2726507	singlet	39_39
AAACCTGAGGGCACTA-1	singlet	465_466	11	465_466	singlet	0.374426050641161	singlet	singlet	-0.65760195	singlet	465_466
AAACCTGAGTAATCCC-1	singlet	660_661	4	660_661	singlet	0.247842972104563	singlet	singlet	-3.5948637	singlet	660_661
AAACCTGAGTAGCGGA-1	doublet	doublet	unassigned	doublet	doublet	0.342998285281922	singlet	singlet	-0.50507957	doublet	doublet
…	…	…	…	…	…	…	…	…	…	…	…

2.23.4 Citation

If you used the Demuxafy platform for analysis, please reference our paper (REFERENCE).

CHAPTER

THREE

SUPPORT

If you're having trouble with any part of the Demultiplexing and Doublet Detecting Pipeline, feel free to submit an issue.