rCASC supplementary data

Luca Alessandrì, Francesca Cordero, Marco Beccuti, Maddalena Arigoni, Martina Olivero, Greta Romano, Sergio Rabellino, Gennaro De Libero, Luigia Pace, and Raffaele A Calogero

25/10/2018

Contents

Contents

Section 1 rCASC: a single cell analysis workflow designed to provide data reproducibility

Since the end of the 90's omics high-throughput technologies have generated an enormous amount of data, reaching today an exponential growth phase. Analysis of omics big data is a revolutionary means of understanding the molecular basis of disease regulation and susceptibility, and this resource is accessible to the biological/medical community via bioinformatics frameworks. However, because of the fast evolution of computation tools and omics methods, the *reproducibility crisis* is becoming a very important issue [*Nature, 2018*] and there is a mandatory need to guarantee robust and reliable results to the research community [*Global Engage Blog*].

Our group is deeply involved in developing workflows that guarantee both **functional** (i.e. the information about data and the utilized tools are saved in terms of meta-data) and **computation** reproducibility (i.e. the real image of the computation environment used to generate the data is stored). For this reason we are managing a bioinformatics community called *reproducible-bioinformatics.org* [*Kulkarni et al.*] designed to provide to the biological community a reproducible bioinformatics ecosystem [*Beccuti et al.*].

rCASC, reproducible Cluster Analysis of Single Cells, is part of the *reproducible-bioinformatics.org* project and provides single cell analysis functionalities within the reproducible rules described by Sandve et al. [*PLoS Comp Biol. 2013*]. rCASC is designed to provide a workflow (Figure 1) for cell-subpopulation discovery.

The workflow allows the direct analysis of fastq files generated with *10X Genomics platform*, *InDrop technology* or a count matrix having as column-names cells identifier and as row names ENSEMBL gene annotation. In the following paragraphs the functionalities of rCASC workflow are described.

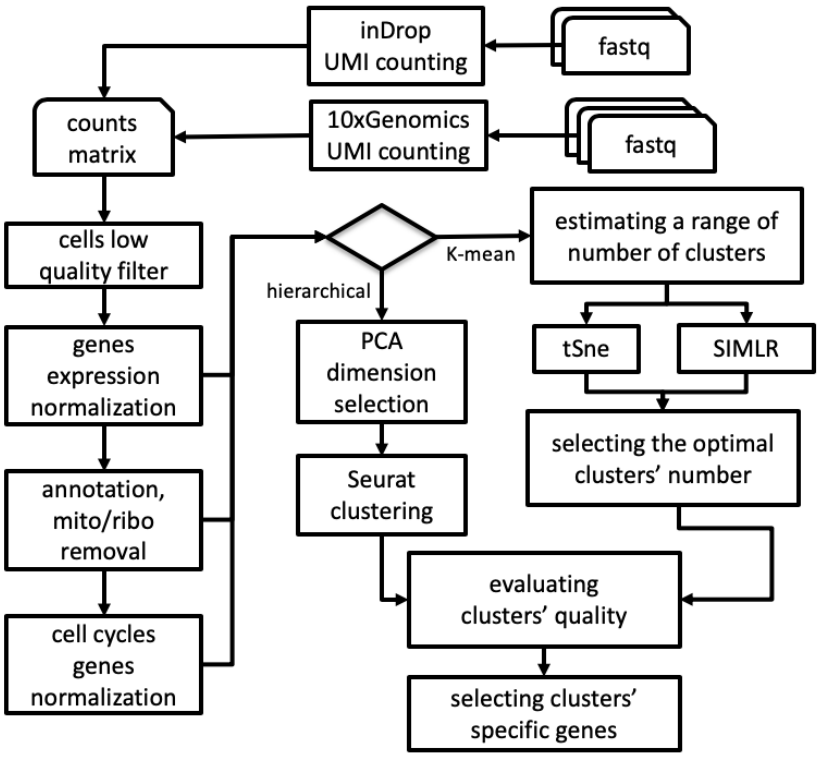


Figure 1: rCASC workflow

Section 1.1 Minimal hardware requirements to run rCASC

The RAM and CPU requirements are dependent on the data set under analysis, e.g. 500-600 cells can be effectively analysed using the hardware described by Beccuti [*Bioinformatics 2018*]:

- 32 Gb RAM
- 2.6 GHz Core i7 6700HQ with 8 threads.
- 500 GB SSD

The analysis time can be significantly improved increasing the number of cores. Implementation of the workflow for computers farm, using *swarm*, is under implementation.

Section 1.2 Installation

The minimal requirements for installation are:

- A workstation/server running 64 bits Linux.
- Docker daemon installed on the machine, for more info see this document:
  - *https://docs.docker.com/engine/installation/*.
- A scratch folder, e.g. /data/scratch, possibly on a fast I/O SSD disk, writable by everybody:

```
chmod 777 /data/scratch
```

The functions in rCASC package require that user belongs to a *group* with the rights to execute docker. See the following document for more info: *https://docs.docker.com/install/linux/linux-postinstall/*

The following commands allow the rCASC installation in *R* environment:

```
install.packages("devtools")
library(devtools)
install_github("kendomaniac/rCASC")

# downloading the required docker containers
library(rCASC)
downloadContainers()
```

Section 2 Counts generation

This session refers to the generation of a counts table starting from fastq files generated by inDrop and 10XGenomics platforms.

Section 2.1 inDrop seq

inDrop single-cell sequencing approach was originally published by Klein [*Cell 2015*]. Then, the authors published the detailed protocol in [*Zilionis et al. Nature Protocols 2017*], which has different primer design comparing to the original paper (Figure 2).

Figure 2: inDrop library structure

The analysis shown below is based on the protocol in Zilionis [*Nature Protocols 2017*], which is the version 2 (Figure 3) of the inDrop technology, actually distributed by *1CellBio*.

Section 2.1.1 inDrop data analysis

There are two main functions, **indropIndex** and **indropCounts**, allowing the generation of a counts table starting from fastq files.

Figure 3: inDrop v2

Section 2.1.1.1 indropIndex: Creates a reference genome for inDrop V2

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *index.folder*: the folder where the reference genome will be created
  - *ensembl.urlgenome*: the link to the ENSEMBL unmasked genome sequence of interest.
  - *ensembl.urlgtf*: the link to the ENSEMBL GTF of the genome of interest.

```
library(rCASC)
#running indropCounts index build
indropIndex(group="docker", index.folder=getwd(),
 ensembl.urlgenome="ftp://ftp.ensembl.org/pub/release-87/fasta/mus_musculus/dna/Mus_musculus.GRCm38.dna.toplevel.fa.gz",
 ensembl.urlgtf="ftp://ftp.ensembl.org/pub/release-87/gtf/mus_musculus/Mus_musculus.GRCm38.87.gtf.gz")
```

Section 2.1.1.2 indropCounts: Converts fastq in UMI counts using *inDrop workflow*

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, a character string indicating the path of the scratch folder
  - fastq.file*, a character string indicating the folder where input data are located and where output will be written
  - index.folder*, a character string indicating the folder where transcriptome index was created with indropIndex.
  - split.affixes*, the string separating SAMPLENAME from the Rz_001.fastq.gz
  - bowtie.index.prefix*, the prefix name of the bowtie index. If genome was generated with indropIndex function the bowtie index is genome (default).

```
system("wget 130.192.119.59/public/testMm_S0_L001_R1_001.fastq.gz")
system("wget 130.192.119.59/public/testMm_S0_L001_R2_001.fastq.gz")
library(rCASC)
indropCounts(group="docker", scratch.folder="/data/scratch", fastq.folder=getwd(),
        index.folder="/data/genomes/indropMm10", sample.name="testMm", split.affixes="S0_L001",
        bowtie.index.prefix="genome", M=10, U=2, D=400, low.complexity.mask="False")
```

Section 2.2 10XGenomics

The rCASC function, **cellrangerCount** allows the generation of a counts table starting from fastq files. This function implements *Cellranger*, the 10xGenomics tool allowing the conversion of the fastqs, generated with 10XGenomics platform, into a count matrix. Genome indexes are retrieved from 10Xgenomics repository:

```
setwd("/data/genomes/cellranger_hg38")
#getting the hg38 human genome cellranger index
system("wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-GRCh38-2.1.0.tar.gz")
setwd("/data/genomes/cellranger_hg19")
#getting the hg19 human genome cellranger index
system("wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-hg19-2.1.0.tar.gz")
setwd("/data/genomes/cellranger_mm10")
#getting the mm10 mouse genome cellranger index
system("wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-mm10-2.1.0.tar.gz")
setwd("/data/genomes/cellranger_hg19mm10")
#getting the human and mouse cellranger index
system("wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-hg19-and-mm10-2.1.0.tar.gz")
```

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *fastq*, the path to the folder, where 10XGenomics fastq.gz files are located.
  - *transcriptome*, the path to the Cellranger compatible transcriptome reference

```
home <- getwd()
library(rCASC)
downloadContainers()
setwd("/data/genomes/cellranger_hg19mm10")
#getting the human and mouse cellranger index
system("wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-hg19-and-mm10-2.1.0.tar.gz")
setwd(home)
# downloading 100 cells 1:1 Mixture of Fresh Frozen Human (HEK293T) and Mouse (NIH3T3) Cells
system("wget http://cf.10xgenomics.com/samples/cell-exp/2.1.0/hgmm_100/hgmm_100_fastqs.tar")
system("tar xvf hgmm_100_fastqs.tar")
# The cellranger analysis is run without the generation of the secondary analysis
cellrangerCount(group="docker",  transcriptome.folder="/data/genomes/cellranger_hg19mm10",
                fastq.folder="/data/test_cell_ranger/fastqs",  expect.cells=100,
                nosecondary=TRUE, scratch.folder="/data/scratch")
```

The analysis done above took 56.4 mins on a *SeqBox*, equipped with an Intel i7-6770HQ (8 threads), 32 Gb RAM and 500Gb SSD.

The output of the above analysis are two counts matrices **results_cellranger.cvs** and **results_cellranger.txt** and a folder called **results_cellranger**, which contains the full cellranger output, more information on cellranger output can be found at *10XGenomics web site*.

Section 3 Counts matrix editing

This paragraph describes a set of functions that can be used to remove low quality cells and non-informative genes from counts tables generated by any single-cell sequencing platform.

- **Counts manipulation**:
  - Removing non informative genes: **filterZeros**
  - Plotting detected genes versus total number of UMIs (Unique Molecular Identifier)/reads: **genesUmi**
  - Removing low quality cells: **lorenzFilter**
  - Data normalization: **scnorm**, minimal requirements 10K counts/cell, works best with whole transcript sequencing
  - Data normalization: **umiNorm**, global normalization methods TMM and RLE are suitable for UMI data
  - ENSEMBL annotation and mitochodrial/ribosomal genes removal: **scannobyGtf**
  - Converting a count table in log10: **counts2log**
  - Removing cell cycle bias: **recatPrediction**/**ccremove**

Section 3.1 Removing non informative genes

The function **filterZeros** retains all genes having a user defined fraction of zeros (between 0 and 1, where 1 indicate that only genes without any 0s are retained, and 0 indicates that genes with at least a single value different from zero are retained), and plots the frequency distribution of gene counts in the dataset.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, a character string indicating the path of the tab delimited file of cells un-normalized expression counts
  - *threshold*, a number from 0 to 1 indicating the fraction of max accepted zeros in each gene. 0 is set as default and it eliminates only genes having no expression in any cell.
  - *sep*, separator used in count file, e.g. '\t', ','

The output is a PDF providing zeros distributions before and after removal of genes with 0s counts. A tab delimited file with the prefix **filtered__** in which the filtered data are saved.

IMPORTANT: In case user would like to apply cell quality filter, e.g. **lorenzFilter**, it is convenient to remove only genes with 0 counts in all cells, i.e. threshold=0 (Figure 4).

```
system("wget http://130.192.119.59/public/testSCumi_mm10.csv.zip")
unzip("testSCumi_mm10.csv.zip")
tmp <- read.table("testSCumi_mm10.csv", sep=",", header=T, row.names=1)
dim(tmp)
#27998   806
write.table(tmp, "testSCumi_mm10.txt", sep="\t", col.names=NA)
```

```
filterZeros(data.folder=getwd(),counts.matrix="testSCumi_mm10.txt", threshold=0)
#Out of 27998 genes 11255 are left after removing genes with no counts
#output is filtered_testSCumi_mm10.txt
```

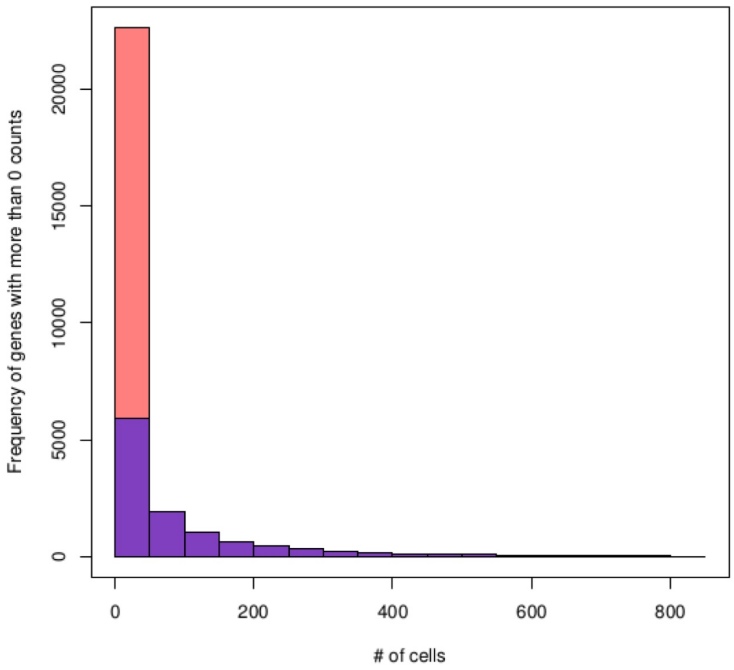


Figure 4: Zeros distribution in full table, orange, and filtered table, blue

Section 3.2 Plotting genes numbers versus total UMIs/reads in each cell

To estimate the overall amount of genes detectable in each cell, the function **genesUmi** generates a plot of the number of genes present in a cell with respect to the total number of UMI in the same cell. The number of UMIs required to call a gene present in a cell is a parameter defined by the user, the suggested value is 3 UMIs.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, a character string indicating the path of the file tab delimited of cells un-normalized expression counts.
  - *umiXgene*, an integer defining how many UMI are required to call a gene as present. default: 3
  - *sep*, separator used in count file, e.g. '\t', ','

The output is a pdf named **genes.umi.pdf**, where each dot represents a cell. X axis is the total number of UMIs/reads mapped on each cell in log10 format and Y axis is the number of detected genes.

```
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="filtered_testSCumi_mm10.txt", umiXgene=3)
```

In Figure 5 it is shown the distribution of genes in cells for 'filtered_testSCumi_mm10.txt' counts table.

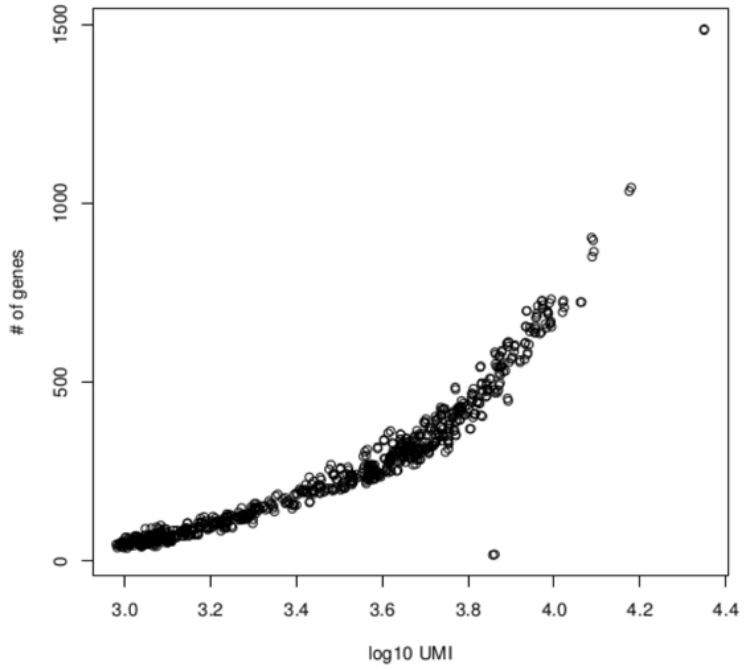


Figure 5: genesUmi output

Section 3.2.1 Further considerations about the number of reads/UMIs to be used in a single-cell sequencing experiment.

Ziegenhain et al. published a comparison between single cell sequencing protocols and they show that, in a simulated experiment, at least 250K reads/cell are required for the detection of at least 80% of differentially expressed genes between two groups (Figure 6). Ziegenhain observation clearly also apply to sub-populations clustering.

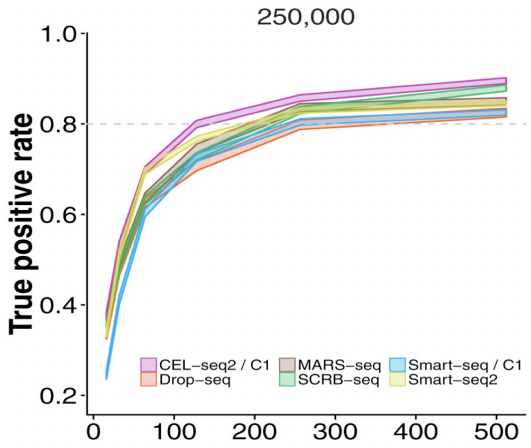


Figure 6: Modified from Figure 6 in Ziegenhain et al. (Mol. Cell 2017). Power of scRNA-Seq methods. For more information on the experiment please see Ziegenhain paper.

Sequencing depth, which affects differential expression analysis and sub-population partitioning, influences the structure of a single-cell dataset at two levels:

- number of genes called present in the experiment,
- robustness of gene expression, i.e. number of reads associated to a gene.

In particular, the number of genes called present in the experiment is the key element for the discrimination between sub-populations. In Figure 7 it shown the effect of sequencing depth on the number of detectable genes in a set of 10XGenomics sequencing experiments (25K sequenced reads/cell extracted from CD19 B-cells [*Zheng et al*], 83K sequenced reads/cell extracted from naive CD8+ T-cells [*GSM2833284*], and 250K sequenced reads/cell from an unpublished brain mouse experiment) and in an unpublished whole transcript human MAIT-cells single-cell sequencing done on Fluidigm C1 (25K, 100K, and 250K sequenced reads/cell were subsampled from the original fastqs). 3 UMI are used as minimal threshold to call present a gene in 10XGenomics experiments and 5 reads [*Tarazona et al. 2011*] as minimal threshold to call a gene present in single cell whole transcriptome experiments.

```
#10XGenomics experiments
system("wget http://130.192.119.59/public/Zheng_cd19_288cells.txt.zip")
unzip("Zheng_cd19_288cells.txt.zip")
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="Zheng_cd19_288cells.txt", umiXgene=3)
```

```
system("mv genes.umi.pdf genes.umi_25k.pdf")

system("wget http://130.192.119.59/public/GSM2833284_Naive_WT_Rep1_288cell.txt.zip")
unzip("GSM2833284_Naive_WT_Rep1_288cell.txt.zip")
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="GSM2833284_Naive_WT_Rep1_288cell.txt", umiXgene=3)
system("mv genes.umi.pdf genes.umi_86k.pdf")

system("wget http://130.192.119.59/public/brain_unpublished_288cells.txt.zip")
unzip("brain_unpublished_288cells.txt.zip")
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="brain_unpublished_288cells.txt", umiXgene=3)
system("mv genes.umi.pdf genes.umi_250k.pdf")

#whole transcript experiment
system("wget http://130.192.119.59/public/c1_experiment.zip")
unzip("c1_experiment.zip")
setwd("c1_experiment")
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="250K_counts.txt", umiXgene=5)
system("mv genes.umi.pdf genes.umi_250K.pdf")
genesUmi(data.folder=getwd(), counts.matrix="100K_counts.txt", umiXgene=5)
system("mv genes.umi.pdf genes.umi_100K.pdf")
genesUmi(data.folder=getwd(), counts.matrix="25K_counts.txt", umiXgene=5)
system("mv genes.umi.pdf genes.umi_25K.pdf")
```

Figure 7 clearly shows that the number of genes detectable by 10XGenomics sequencing (Figure 7A-C) is far less of those detectable using a whole transcript experiment (Figure 7D-F). In the case of 25K reads/cells in 3' end sequencing (Figure 7A) the number of called genes goes from a doxen of genes to 350. In a whole transcript sequencing experiment where 25K reads/cells were considered (Figure 7D), 350 genes is the lowest number of genes detectable in a cell. In 10XGenomics experiment with 250K reads/cell the range of detectable genes goes from few hunderd to 2000, which is relatively similar to the number of detectable genes in a whole transcripts experiment where the same number of reads/cells were considered. The larger scattering and the overall lower number of detectable genes in 3' end sequencing experiment, with respect to whole transcript experiments, is a peculiarity of the technology [*Ziegenhain et al.*].
It has also to be highlighted that cells with a very low number of genes called present will have a gene repertoire made mainly of housekeeping genes, ribosomal and mitochondrial genes, see Figure 10 in **Section 3.4**. Thus, the lack of cell-type specific genes make these cells useless for the identification of functional cell sub-populations.

The above observations indicate that 3' end sequencing has a much lower genes called present with respect to whole transcript sequencing.

We have further investigated the following point:

- Is the number of total UMIs/cell affecting the separation between sub-populations?

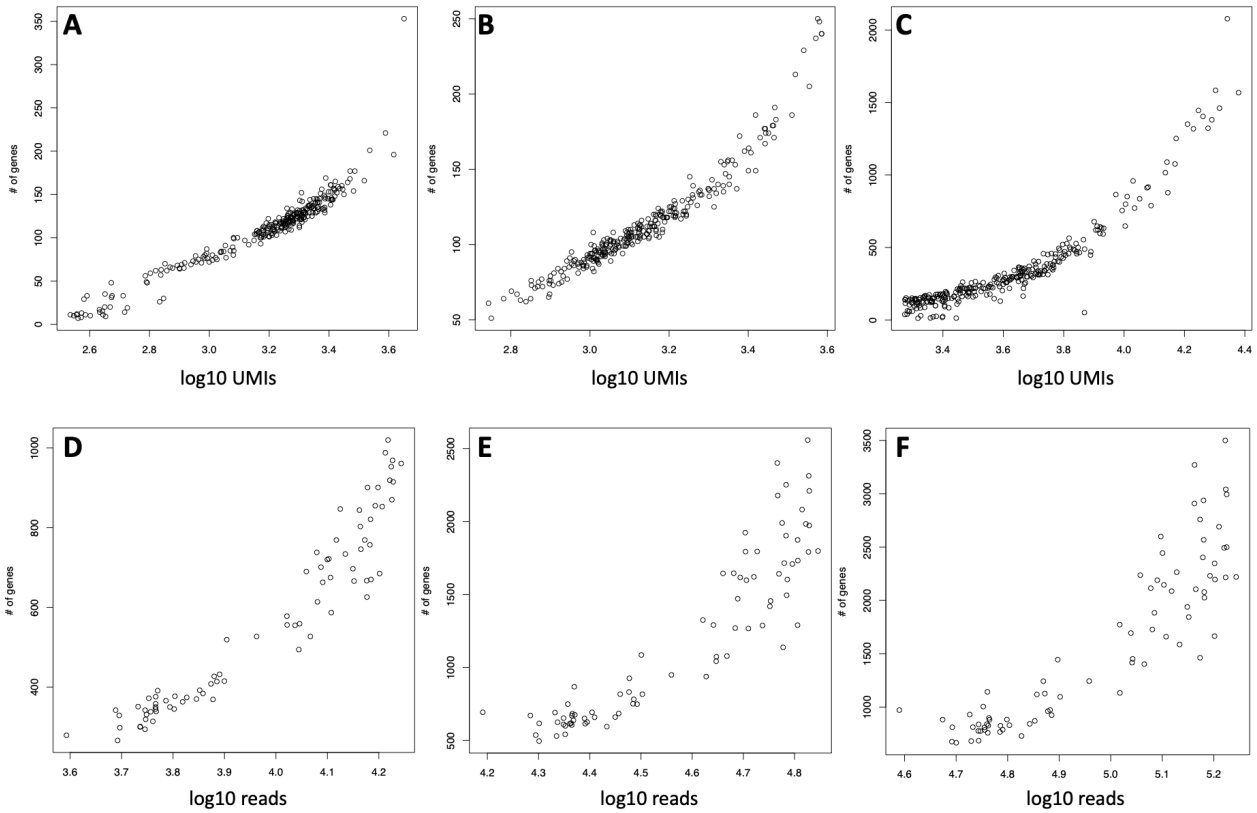


Figure 7: Number of detected genes with respect to mapped reads. A) 25K reads/cell 10XGenomics platform, 3' end sequencing. B) 83K reads/cell 10XGenomics platform, 3' end sequencing. C) 250K reads/cell 10XGenomics platform, 3' end sequencing. D) 25K reads/cell C1 platform, whole transcript sequencing, E) 100K reads/cell C1 platform, whole transcript sequencing, F) 250K reads/cell C1 platform, whole transcript sequencing.

To address the above point we used two types of cells belonging to the T-cells [*Zheng* 2016]. The two sets of cells were sequenced with a coverage of approximately 21K reads/cell:

- T-cytotoxic (10209 cells, Figure 8A) cells,
- T-regulatory (10263, Figure 8B) cells.

We generated three datasets:

- **d3.4**, which is made of 100 cells randomly selected within cells having, in each of the two datasets, a total UMIs/cell value greater than 2511 in each cell.
- **d3**, which is made of 100 cells randomly selected within cells having, in each of the two datasets, a total UMIs/cell value comprised between 2511 and 1000 in each cell.
- **d3m**, which is made of 100 cells randomly selected within cells having, in each of the two datasets, a total UMIs/cell smaller than 1000 in each cell.

The separation between T-cytoxic and T-regulatory achievable with PCA is shown in Figure 8C-E.

```
system("wget http://130.192.119.59/public/counts_effect.zip")
unzip(counts_effect.zip)
setwd("counts_effect")

topx(data.folder=getwd(),file.name="df3.4.txt",threshold=1000, logged=FALSE)
library(docker4seq)
pca(experiment.table="df3.4_1000.txt", type="counts",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())

topx(data.folder=getwd(),file.name="df3.txt",threshold=1000, logged=FALSE)
library(docker4seq)
pca(experiment.table="df3_1000.txt", type="counts",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())

topx(data.folder=getwd(),file.name="df3m.txt",threshold=1000, logged=FALSE)
library(docker4seq)
pca(experiment.table="df3m_1000.txt", type="counts",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())
```

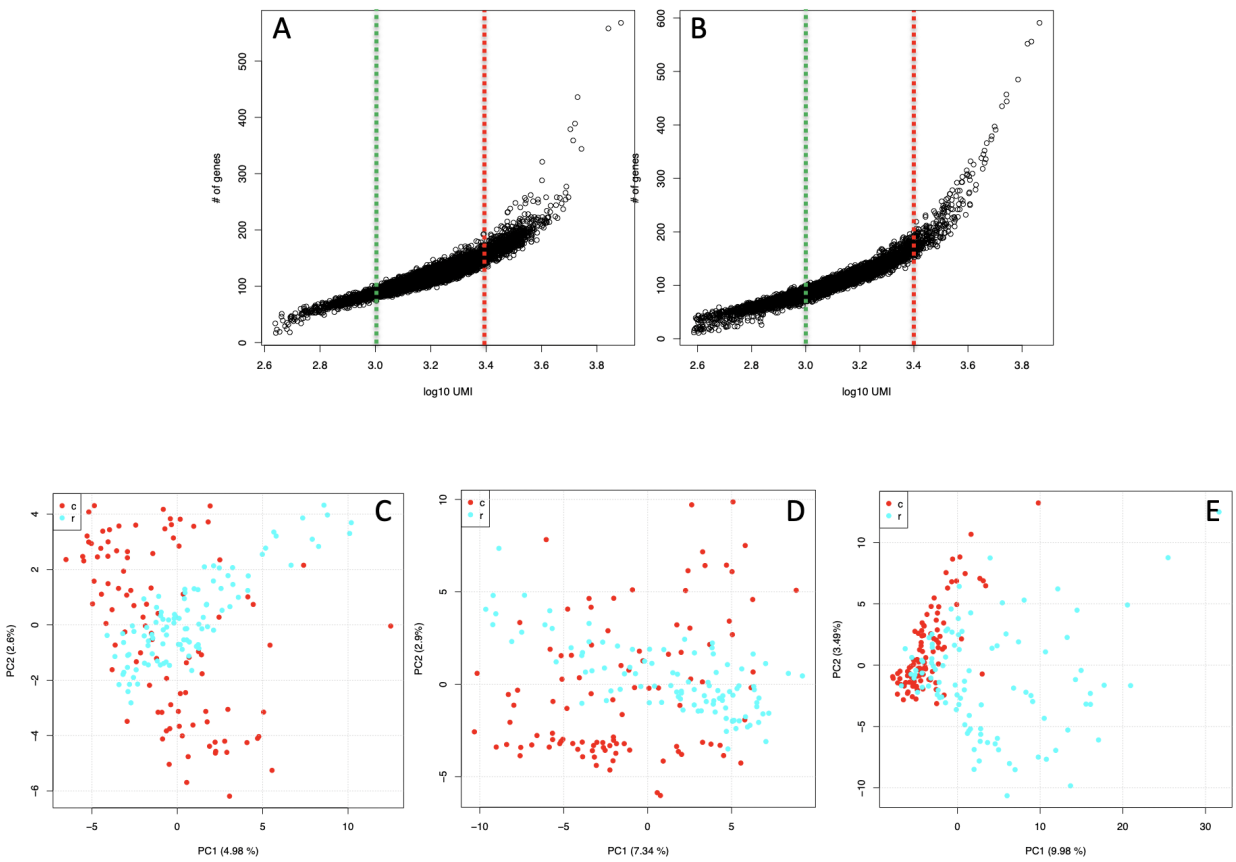


Figure 8: Detectable genes in a Zheng data subset. A) T-cytotoxic genes versus total cell reads plot. B) T-regulatory genes versus total cell reads plot. C) d3m set, made of cells with less than 1000 counts each, D) df3, made of cells with counts between 1000 and 2511. E) d3.4, made of cells with more than 2511 counts each.

It is notable that only the datasets including cells with more than 2511 UMIs/cell (Figure 8E) is the one where the separation between the two T-cell types improves. In the d3 and d3m (Figure 8C,D) the amount of variance explained by the first component is much lower of that observable in d3.4 and the datasets are intersperse.

Thus, since 3' end sequencing platforms (10Xgenomics, inDrop) has the advantage to produce high number of sequenced cells, users might decide to selected for clustering only the subset of cells with the highest number of genes called present.

Section 3.3 Identifying and removing cell low-quality outliers

Lorenz statistics was implemented in rCASC **lorenzFilter** function. This function derives from the work of Diaz and coworkers [*2016*] detecting low quality cells and this statistics correlates with live-dead staining [*Diaz et al. 2016*].

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):

- *scratch.folder*, the path of the scratch folder
- *file*, full path to the count file MUST be provided
- *p_value*, lorenz statistics threshold, suggested value 0.05 (i.e. 5% probability that the a cell of low quality is selected)
- *separator*, separator used in count file, e.g. '\t', ','

The output is a counts table without low quality cells and with the prefix **lorenz_filtered_**. Output will be in the same format and with the same separator of input.

```
#example data set provided as part of rCASC package
system("wget http://130.192.119.59/public/testSCumi_mm10.csv.zip")
unzip("testSCumi_mm10.csv.zip")
#IMPORTANT: full path to the file MUST be cell count file included!
library(rCASC)
# the p_value indicate the probability that a low quality cell is retained in the
# dataset filtered on the basis of Lorenz Statistics.
lorenzFilter(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(),"filtered_testSCumi_mm10.txt", sep="/"),
               p_value=0.05, separator='\t')

tmp0 <- read.table("filtered_testSCumi_mm10.txt", sep="\t", header=T, row.names=1)
#806 cells

tmp <- read.table("lorenz_filtered_testSCumi_mm10.txt", sep="\t", header=T, row.names=1)
#785 cells
```

In the example above 21 cells were removed because of their low quality (Figure 9).

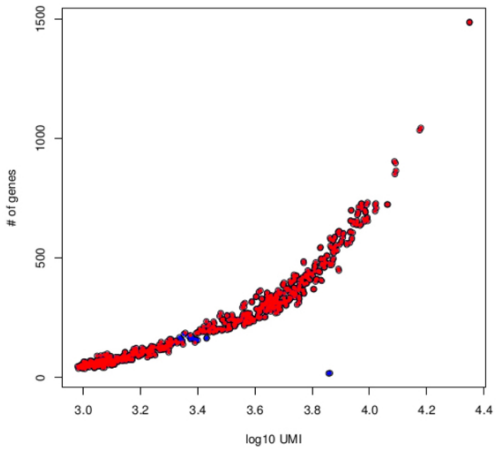


Figure 9: Effect of Lorenz filtering, cells shown in blue have been discarded because of their low quality.

Section 3.4 Annotation and mitocondrial/ribosomal protein genes removal

The function **scannobyGtf** allows the annotation of single-cell matrix, if ENSEMBL gene ids are provided. The function requires the ENSEMBL GTF of the organism under analysis and allows the selection of specific annotation biotypes, e.g. protein_coding.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, full path to the count file MUST be provided
  - *gtf.name*, ENSEMBL gtf file name. GTF is located in the same folder where counts file is.
  - *biotype*, biotype of interest. See *www.ensembl.org/info/genome/genebuild/biotypes.html* for more information
  - *mt*, a boolean to define if mitochondrial genes have to be removed, FALSE mean that mt genes are removed
  - *ribo.proteins*, a boolean to define if ribosomal proteins have to be removed, FALSE mean that ribosomal proteins (gene names starting with rpl or rps) are removed
  - *umiXgene*, a integer defining how many UMIs are required to call a gene as present. default: 3

```
#running annotation and removal of mito and ribo proteins genes
system("wget ftp://ftp.ensembl.org/pub/release-92/gtf/mus_musculus/Mus_musculus.GRCm38.92.gtf.gz")
system("gunzip Mus_musculus.GRCm38.92.gtf.gz")
scannobyGtf(group="docker", file=paste(getwd(),"lorenz_filtered_testSCumi_mm10.txt",sep="/"),
                    gtf.name="Mus_musculus.GRCm38.92.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE, umiXgene=3)
```

Ribosomal RNA and ribosomal proteins represent a significant part of cell cargo. Large cells and actively proliferating cells will have respectively more ribosomes and more active ribosome synthesis [*Montanaro et al. 2008*]. Thus, ribosomal proteins expression might represent one of the major confounding factor in cluster formation between active and dormient cells. Furthermore, the main function of mitochondria is to produce energy through aerobic respiration. The number of cell mitochondria depends on its metabolic demands [*Nasrallah and Horvath 2014*]. This might also affect clustering, favoring the separation between metabolic active and resting cells, with respect to functional differences between sub-populations. *scannobyGtf* allows also the removal of mitocondrial and ribosomal protein genes.

```
library(rCASC)
scannobyGtf(group="docker", file=paste(getwd(),"lorenz_filtered_testSCumi_mm10.txt",sep="/"),
                    gtf.name="Mus_musculus.GRCm38.92.gtf", biotype="protein_coding",
                    mt=FALSE, ribo.proteins=FALSE, umiXgene=3)
```

In figure 10 is shown the effect of the removal of both mitochondrial and ribosomal protein genes. It is notable that, in this example, in cells with less than 1000 UMIs nearly all detected genes were only mitochondrial and ribosomal protein genes. This filter is suitable to identify cells which do not contain any informative gene other than mitochondrial and ribosomal proteins. However, in case the difference between resting and actively proliferating cells represents an important element of cell sub-population discovery this filter should not be applied.

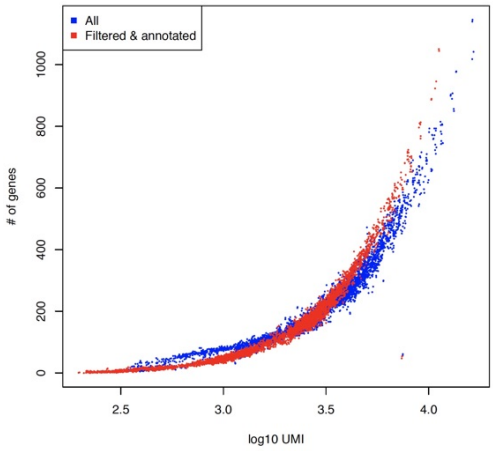


Figure 10: Removing mithochondrial and ribosomal proteins genes. In red is shown the dataset after removal of mitochondrial and ribosomal protein genes.

Section 3.5 Top expressed genes

For clustering purposes user might decide to use the top expressed genes. The function **topx** select the X top expressed genes given a user defined threshold. The function also produces a pdf file gene_expression_distribution.pdf showing the changes in the UMIs/gene expression distribution upon **topx** filtering.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *data.folder*, folder where input data are located and where output will be written
  - *file.name*, counts table file name.
  - *threshold*, number of top expressed genes to be selected
  - *logged*, boolean, if FALSE gene expression data are log10 transformed before being plotted.

```
library(rCASC)
genesUmi(data.folder=getwd(), counts.matrix="lorenz_filtered_testSCumi_mm10.txt", umiXgene=3)
topx(data.folder=getwd(),file.name="lorenz_filtered_testSCumi_mm10.txt",threshold=10000, logged=FALSE)
genesUmi(data.folder=getwd(), counts.matrix="lorenz_filtered_testSCumi_mm10_10000.txt", umiXgene=3)
```

IMPORTANT: The core clustering tool in rCASC is SIMLR, **Section 5**. SIMLR requires that the number of genes must be larger then the number of analysed cells.

Section 3.6 Data normalization

The best way to normalize single-cell RNA-seq data has not yet been resolved, expecially in the case of UMI data. We inserted in our workflow two possible options:

- *SCnorm*, which works best with whole transcript data.
- *scone*, which provides different global scaling methods that can be applyed to UMI single-cell data.

Section 3.6.1 SCnorm

SCnorm performs a quantile-regression based approach for robust normalization of single-cell RNA-seq data. SCnorm groups genes based on their count-depth relationship then applies a quantile regression to each group in order to estimate scaling factors which will remove the effect of sequencing depth from the counts.

IMPORTANT: SCnorm is not intended for datasets with more than ~80% zero counts, because of lack of algoritm convergence in these situations.

Section 3.6.1.1 Check counts-depth relationship

Before normalizing using **scnorm**, it is advised to check the data count-depth relationship. If all genes have a similar relationship then a global normalization strategy such as median-by-ratio in the DESeq package or TMM in edgeR will also be adequate. However, when the count-depth relationship varies among genes global scaling strategies leads to poor normalization. In these cases the normalization provided by SCnorm is recommended.

checkCountDepth provides a wrapper, in rCASC, for the checkCountDepth of the *SCnorm package*, which estimates the count-depth relationship for all genes.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, full path to the file MUST be included. Only tab delimited files are supported
  - *conditions*, vector of condition labels, this should correspond to the columns of the un-normalized expression matrix. If not provided data is assumed to come from same condition/batch.
  - *ditherCounts*, boolean. Setting to TRUE might improve results with UMI data.
  - *outputName*, name of output files.
  - *nCores*, number of cores to use.

```
#this specific example is an UMI counts table made of 12 cells having at least 10K UMIs/cell.
system("wget http://130.192.119.59/public/example_UMI.txt.zip")
unzip("example_UMI.txt.zip")
conditions=rep(1,12)
checkCountDepth(group="docker", file=paste(getwd(), "example_UMI.txt", sep="/"),
     conditions=conditions, FilterCellProportion=0.1, FilterExpression=0,
     ditherCounts=TRUE, outputName="example_UMI", nCores=8)
```

The output is a PDF (Figure 11), providing a view of the counts distribution, and a file selected.genes.txt, which contains the genes selected to run the analysis.

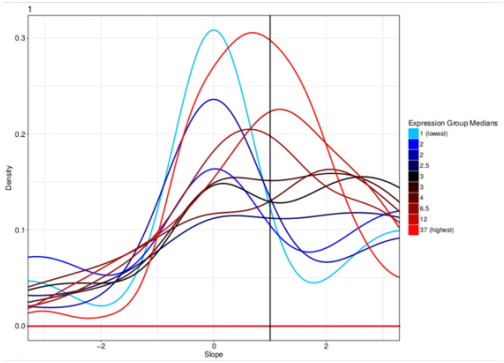


Figure 11: checkCountDepth output plot

Section 3.6.1.2 scnorm

the **scnorm** function executes SCnorm from *SCnorm package*, which normalizes across cells to remove the effect of sequencing depth on the counts and returns the normalized expression count.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, full path to the file MUST be included. Only tab delimited files are supported
  - *conditions*, vector of condition labels, this should correspond to the columns of the un-normalized expression matrix.
  - *outputName*, specify the name of output files.
  - *nCores*, number of cores to use.
  - *filtercellNum*, the number of non-zero expression estimate required to include the genes into the SCnorm fitting (default = 10). The initial grouping fits a quantile regression to each gene, making this value too low gives unstable fits.
  - *ditherCount*, boolean. Setting to TRUE might improve results with UMI data.
  - *PropToUse*, as default is set to 0.25, but to increase speed with large data set could be reduced, e.g. 0.1
  - *PrintProgressPlots*, boolean. If it is set to TRUE produces a plot as SCnorm determines the optimal number of groups

```
system("wget http://130.192.119.59/public/example_UMI.txt.zip")
unzip("example_UMI.txt.zip")
#this specific example is an UMI counts table made of 12 cells having at least 10K UMIs/cell.
conditions=rep(1,12)
scnorm(group="docker", file=paste(getwd(), "example_UMI.txt", sep="/"),
     conditions=conditions,outputName="example_UMI", nCores=8, filtercellNum=10,
     ditherCount=TRUE, PropToUse=0.1, PrintProgressPlots=TRUE, FilterExpression=1)
```

The output files are plots of the normalization effects (Figure 12), a tab delimited file containing the normalized data, with the prefix **normalized_**, and **discarded_genes.txt**, which contains the discarded genes.

scnorm is compliant with *SIMLR*, the rCASC core clustering tool.

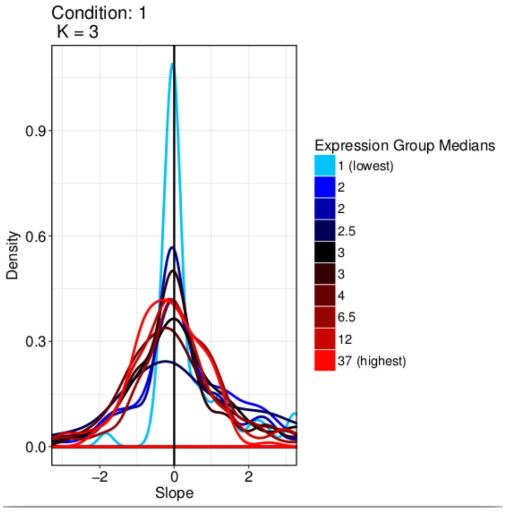


Figure 12: Effect of the SCnorm on the dataset in Figure 9

Section 3.6.2 scone

scone package embeds:

- Centered log-ratio (**CLR**) normalization
- Relative log-expression (**RLE**; DESeq) scaling normalization
  - the scaling factors are calculated for each lane as median of the ratio, for each gene, of its read count of its geometric mean across all lanes.
- Full-quantile normalization
  - quantile normalization is a technique for making two or more distributions identical in statistical properties. To quantile normalize two or more samples to each other, sort the samples, then set to the average (usually, arithmetic mean) of the samples. So the highest value in all cases becomes the mean of the highest values, the second highest value becomes the mean of the second highest values, and so on.
- Simple deconvolution normalization
- Sum scaling normalization
  - Gene counts are divided by the total number of mapped reads (or library size) associated with their lane and multiplied by the mean total count across all the samples of the dataset.
- Weighted trimmed mean of M-values (**TMM**, edgeR) scaling normalization (suitable for single-cell)
  - to compute the TMM factor, one lane is considered a reference sample and the others test samples, with TMM being the weighted mean of log ratios between test and reference, after excluding the most expressed genes and the genes with the largest log ratios.
- Upper-quartile (**UQ**) scaling normalization

- the total counts are replaced by the upper quartile of counts different from 0 in the computation of the normalization factors.

```
#Weighted trimmed mean of M-values (TMM) scaling normalization
system("wget http://130.192.119.59/public/example_UMI.txt.zip")
unzip("example_UMI.txt.zip")
umiNorm(group="docker", file=paste(getwd(), "example_UMI.txt", sep="/"),
       outputName="example_UMI", normMethod="TMM_FN")
```

IMPORTANT: In case sub-population discovery is the analysis task, it is important to check if a specific normalization is compliant with the clustering approach in use. For example in the case of *SIMLR*, the rCASC core clustering tool, the normalizations provided in **scone** are not compliant, because they remove some of the features required to run the SIMLR multikernel learning analysis. TMM is instead compliant with the rCASC implementation of **tSne**. In case *Seurat* clustering is used the data set does not required any normalization since a normalization procedure is included in the algorithm.

Section 3.7 Converting a count table in log10

The function **counts2log** can convert a count table in a log10 values saved in a comma separated or tab delimited file.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, full path to the file MUST be included.
  - *log.base*, the base of the log to be used for the transformation
  - *type*, the type of input file, txt, tab delimited. csv, comma separated

```
counts2log(file=paste(getwd(), "example_UMI.txt", sep="/"), log.base=10)
```

Section 3.8 Detecting and removing cell cycle bias

Single-cell RNA-Sequencing measurement of expression often suffers from large systematic bias. A major source of this bias is cell cycle, which introduces large within-cell-type heterogeneity that can obscure the differences in expression between cell types. *Barron and Li* developed in 2016 a R package called *ccRemover* which removes cell cycle effects and preserves other biological signals of interest.

However, before applying *ccRemover*, it is essential to address if the removal of cell cycle effect is required. *reCAT* is a modeling framework for unsynchronized single-cell transcriptome data that can reconstruct cell cycle time-series. Thus, reCAT cell cycle prediction step can be used to check if cell cycle effect can be detected in a dataset and therefore ccRemover normalization approach will be needed.

Section 3.8.1 Evaluating the presence of cell cycle effect in a dataset: *reCAT*

reCAT prediction step is implemented in rCASC in the function **recatPrediction**, which requires a data set annotated using **scannobyGtf**.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path to the input file
  - *separator*, separator used in count file, e.g. '\t', ','
  - *geneNameControl*, 0 if the matrix has gene symbol without ENSEMBL code. 1 if the gene names is formatted like this : ENSMUSG00000000001:Gnai3. If the gene names is only ENSEMBL name SCannoByGtf has to be executed before recatPrediction.
  - *seed*, important parameter for reproduce the same result with the same input, default 111

To show the differences existing between a dataset characterized by cell cycle bias and one that is not, we used two datasets:

- the dataset published by [*Buettner et al.*], containing naive-T-cells and T-helper2-cells mixed together and sorted on the basis of the cell cycle state.
- The quiescent naive T-cell dataset part of the publication of *Pace et al.*, expected to be in G0.

To execute the analysis on the same number of cells, 288 cells were randomly selected from quiescent naive T-cell dataset. In Figure 13A the presence of oscillatory behaviour is evident in the predicted cells time serie and the G1 and G2M trends are indicated respectively in blue and red dashed curves. On the other hand the oscillatory behaviour is totally absent (Figure 13B) in the naive T-cells, which are expected to be quiescent in G0.

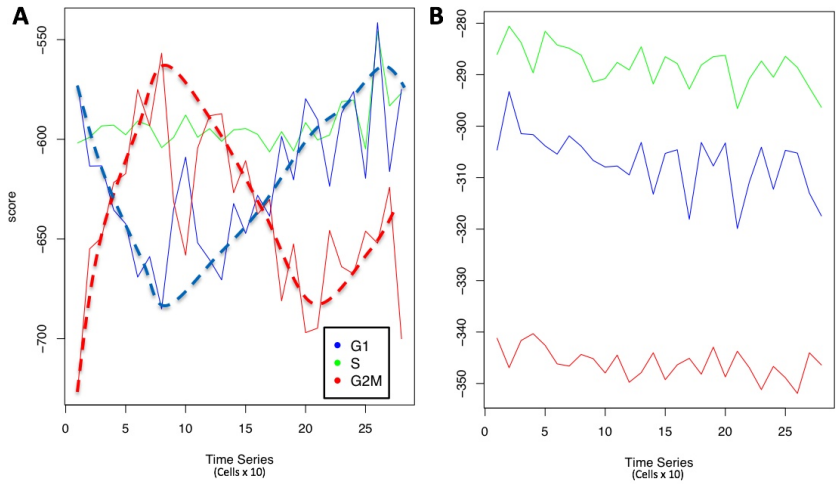


Figure 13: Cell cycle assignment to the cells. A) Buettner et al. (Nat. Biotechnol. 2015) raw dataset, cells are expected to be distributed in G1, S and G2M, B) Naive T-cells, expected to be mainly in G0 (Science 2018).

```
#preparing the data for the analysis
 system("wget http://130.192.119.59/public/buettner_G1G2MS_counts.txt.zip")
 unzip("buettner_G1G2MS_counts.txt.zip")

#annotatiing the data set to obtain the gene names in the format ensemblID:symbol
scannobyGtf(group="docker", file=paste(getwd(),"buettner_G1G2MS_counts.txt",sep="/"),
                    gtf.name="Mus_musculus.GRCm38.92.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE,umiXgene=3)

#selecting the top 10000 most expressed genes
topx(data.folder=getwd(),file.name="annotated_buettner_G1G2MS_counts.txt",threshold=10000, logged=FALSE)

#running cell cycle prediction
recatPrediction(group="docker",scratch.folder="/data/scratch",
                file=paste(getwd(), "annotated_buettner_G1G2MS_counts_10000.txt", sep="/"),
                separator="\t", geneNameControl=1, window=10, seed=111)


#same analysis as above on 10XGenomix data of quescent naive-T cells
system("wget http://130.192.119.59/public/GSM2833284_Naive_WT_Rep1_288cell.txt.zip")
unzip("GSM2833284_Naive_WT_Rep1_288cell.txt.zip")
scannobyGtf(group="docker", file=paste(getwd(),"GSM2833284_Naive_WT_Rep1_288cell.txt",sep="/"),
                    gtf.name="Mus_musculus.GRCm38.92.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE,umiXgene=3)
topx(data.folder=getwd(),file.name="annotated_GSM2833284_Naive_WT_Rep1_288cell.txt",threshold=10000, logged=FALSE)
recatPrediction(group="docker",scratch.folder="/data/scratch",
                file=paste(getwd(), "annotated_GSM2833284_Naive_WT_Rep1_288cell_10000.txt", sep="/"),
                separator="\t", geneNameControl=1, window=10, seed=111)
```

Section 3.8.2 Removing cell cycle effect in a dataset: *ccRemover*

ccRemover software is implemented in rCASC in the function **ccRemove**, which also requires a data set annotated using **scannobyGtf**.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path to the input file, including the counts table name
  - *separator*, separator used in count file, e.g. '\t', ','
  - *seed*, is important to reproduce the same results with the same input, default=111
  - *cutoff*, p-value to use: 3 is almost equal to 0.05
  - *species*, human or mouse
  - *rawCount*, 1 for unlogged and not-normalized, 0 otherwise

IMPORTANT: The output of ccRemover does not require log transformation before clustering analysis.

```
#removing cell cycle effect
ccRemove(group="docker" , scratch.folder="/data/scratch",
```

```
        file=paste(getwd(),"annotated_buettner_G1G2MS_counts_10000.txt", sep="/"), separator="\t",
        seed=111, cutoff=3, species="mouse", rawCount=1)
```

The analyses above were done using a SeqBox, equipped with an Intel i7-6770HQ (8 threads), 32 GB RAM and 500 GB SSD. They took 54 and 40 mins for recatPrediction respectively on Buettner and the naive T-cell datasets. 28 mins were needed by ccRemove on Buettner data set. ccRemover analysis produces a ready-for-clustering data normalized matrix. The matrix can be identify by the prefix **LS_cc_**. **ccRemove** output is compliant with *SIMLR*, the rCASC core clustering tool. **ccRemove** output does not require log transformation when applied to SIMLR.

```
#visualizing the dataset before and after cell cycle bias removal
#reformat the matrix header to be suitable with docker4seq PCA plotting function
tmp <- read.table("annotated_buettner_G1G2MS_counts_10000.txt", sep="\t", header=T, row.names=1)
tmp.n1 <- sapply(tmp.n, function(x)x[1])
tmp.n2 <- sapply(tmp.n, function(x)x[2])
names(tmp) <- paste(tmp.n2, tmp.n1, sep="_")
write.table(tmp, "annotated_buettner_G1G2MS_counts_10000bis.txt", sep="\t", col.names=NA)

library(devtools)
install_github("kendomaniac/docker4seq", ref="master")
library(docker4seq)

pca(experiment.table="annotated_buettner_G1G2MS_counts_10000bis.txt", type="counts",
      legend.position="topright", covariatesInNames=TRUE, samplesName=FALSE,
      principal.components=c(1,2), pdf = TRUE,
      output.folder=getwd())

#reformat the matrix header to be suitable with docker4seq PCA plotting function
tmp <- read.table("LS_cc_annotated_buettner_G1G2MS_counts_10000.txt", sep="\t", header=T, row.names=1)
tmp.n1 <- sapply(tmp.n, function(x)x[1])
tmp.n2 <- sapply(tmp.n, function(x)x[2])
names(tmp) <- paste(tmp.n2, tmp.n1, sep="_")
write.table(tmp, "LS_cc_annotated_buettner_G1G2MS_counts_10000bis.txt", sep="\t", col.names=NA)

pca(experiment.table="LS_cc_annotated_buettner_G1G2MS_counts_10000bis.txt", type="TPM",
      legend.position="topright", covariatesInNames=TRUE, samplesName=FALSE,
      principal.components=c(1,2), pdf = TRUE,
      output.folder=getwd())
```

In Figure 14 are shown the results obtained using the ccRemove implementation in rCASC, using the Buettner dataset. The removal of the cell cycle effect (Figure 14B) is clearly shown by a reduction of the variance explained by PC1 and PC2 in the PCA plot.

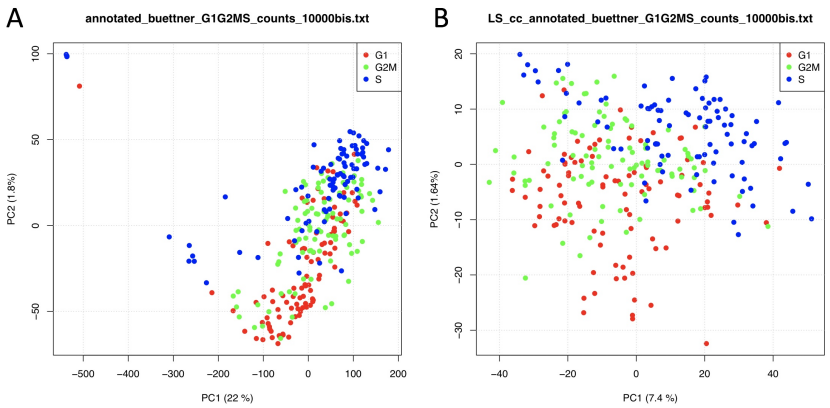


Figure 14: rCASC implementation of the ccRemove. A) PCA analysis of Buettner et al. (Nat. Biotechnol. 2015) raw dataset, B) PCA analysis of ccRemove cell-cycle normalized dataset.

Section 4 Estimating the number of cluster to be used for cell sub-population discovery.

The rCASC core clustering tool is *SIMLR*, which requires as input the number of clusters to be used to aggregate cell sub-populations. Unfortunately, there is no definitive answer to the definition of the most probable number of clusters, in which cells will aggregate. Some of the possible ways to identify the most probable number of clusters is summaries in: *"Determining the optimal number of clusters: 3 must known methods - Unsupervised Machine Learning"*.

Another important aspect is how the number of detectable clusters might change if the number of cells changes in the dataset, e.g. upon removal of a random subset of cells. Because single-cell experiment, at least today, are rarely characterized by biological replications and frequently they represent the initial step of an analysis aimed at the identification for new cell sub-populations, it is very important to assess the stability of cells aggregations detected by clustering methods.

Section 4.1 Estimating the number of cluster to be used for cell sub-population discovery by community detection method.

In rCASC, the identification of the optimal number of clusters is addressed, in presence of cells number perturbations, with *griph*. The clustering performed by griph is graph-based and uses the community detection method *louvain modularity*. Griph algorithm is closer to agglomerative clustering methods, since every node is initially assigned to its own community and communities are subsequently built by iterative merging. Griph is embedded **clusterNgriph** function, which evaluates the number of clusters in which a set of cells will aggregate upon a user defined leave-N%-out cells bootstraps. In the example below the number of clusters are detected for the file 'annotated_buettner_G1G2MS_counts_10000bis.txt', used in **Section 3.8**.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path to the input file, including the counts table name
  - *nPerm*, number of permutations to perform
  - *permAtTime*, number of permutations that can be computed in parallel
  - *percent*, percentage of randomly selected cells removed in each permutation
  - *separator*, separator used in count file, e.g. '\t', ','
  - *logTen*, integer, 1 if the count matrix is already in log10, 0 otherwise
  - *seed*, important value to reproduce the same results with same input, default is 111

```
library(rCASC)
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "annotated_buettner_G1G2MS_counts_10000bis.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)
```

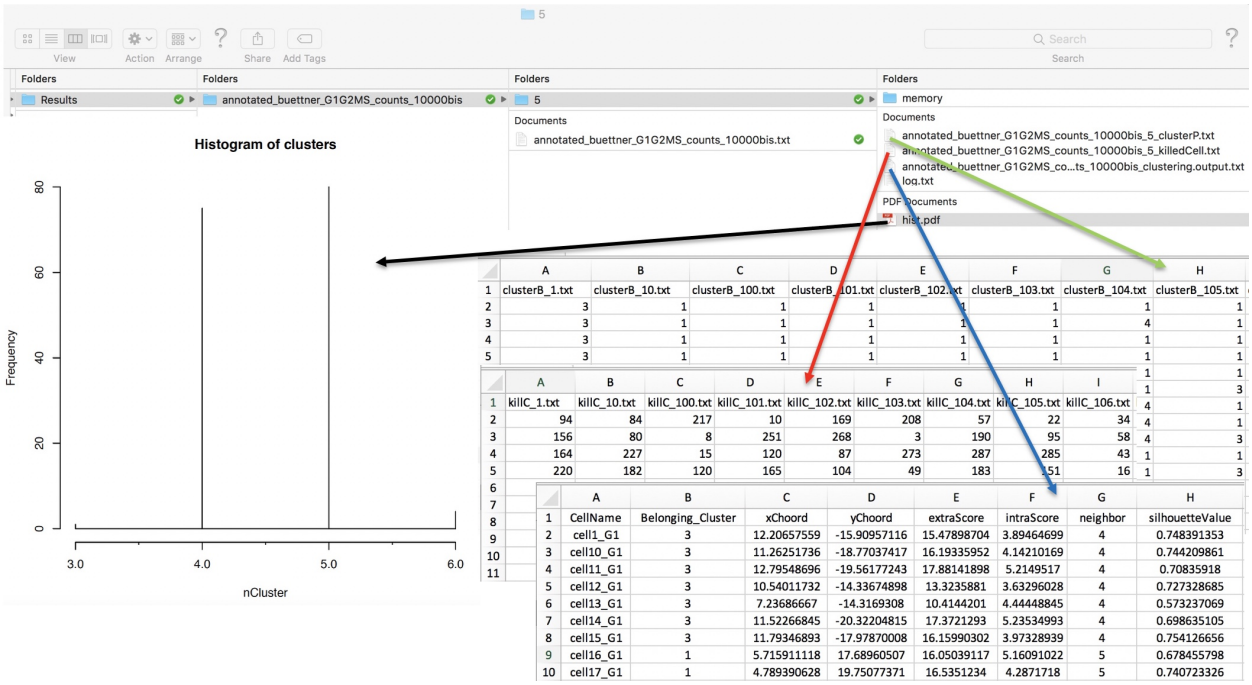
In Figure 15 it is shown the output generated by **clusterNgriph**. The output folder is called **Results** and it is located in the folder from which the analysis started. Within Results is present a folder named as the dataset used for the analysis. In this case 'annotated_buettner_G1G2MS_counts_10000bis'. In the latter folder is present a folder, in this specific example '5', named with the number of clusters that were more represented as result of the bootstrap analysis. The file indicated with the blue arrow contains all the information to generate the griph output plot, used as reference to allocate cells to a specific cluster at each bootstrap step. The file indicated with the green arrow contains the cluster position for each cell over all bootstrap steps. The file indicated with the red arrow contains the cells removed at each bootstrap steps. The file called 'hist.pdf', indicated with the black arrow, is the plot of the frequency of different number of clusters generated by griph as consequence of the bootstrap steps. In this specific case, over 160 permutations, 80 produced 5 clusters, 70 produced 4 clusters and 10 produced 6 clusters.

It has to be noted that in principle, since this dataset has a strong cell cycle effect, we would have expected ideally only three clusters: G1, S and G2M. This toy experiment clearly shows that perturbation of the dataset under analysis can affect the number of detectable clusters. Thus, to identify the clustering condition that guarantee the greatest cell stability in a cluster, in our opinion it is mandatory clustering cells taking in account perturbation effects. In Section 4.2 we further investigate this issue.

Section 4.2 K-mean clustering: Investigating how cell sub-populations aggregation is affected by dataset perturbations.

As indicated above we are interested to identify not only the optimal cluster number but also if the cluster number is affected by removal of a random subset of cells. To observe the effect of datasets perturbation in clustering we built 4 datasets combining different cell types available in *Zheng* 2016 paper (Bold cell types are those that were progressively substituted in setA):

- setA 100 cell randomly selected for each cell type:



	A	B	C	D	E	F	G	H
1	clusterB_1.txt	clusterB_10.txt	clusterB_100.txt	clusterB_101.txt	clusterB_102.txt	clusterB_103.txt	clusterB_104.txt	clusterB_105.txt
2	3	1	1	1	1	1	1	1
3	3	1	1	1	1	1	4	1
4	3	1	1	1	1	1	1	1
5	3	1	1	1	1	1	1	1

	A	B	C	D	E	F	G	H	I
1	killC_1.txt	killC_10.txt	killC_100.txt	killC_101.txt	killC_102.txt	killC_103.txt	killC_104.txt	killC_105.txt	killC_106.txt
2	94	84	217	10	169	208	57	22	34
3	156	80	8	251	268	3	190	95	58
4	164	227	15	120	87	273	287	285	43
5	220	182	120	165	104	49	183	151	16

	A	B	C	D	E	F	G	H
1	CellName	Belonging_Cluster	xChoord	yChoord	extraScore	intraScore	neighbor	silhouetteValue
2	cell1_G1	3	12.20657559	-15.90957116	15.47898704	3.89464699	4	0.748391353
3	cell10_G1	3	11.26251736	-18.77037417	16.19335952	4.14210169	4	0.744209861
4	cell11_G1	3	12.79548696	-19.56177243	17.88141898	5.2149517	4	0.70835918
5	cell12_G1	3	10.54011732	-14.33674898	13.3235881	3.63296028	4	0.727328685
6	cell13_G1	3	7.23686667	-14.3169308	10.4144201	4.44448845	4	0.573237069
7	cell14_G1	3	11.52266845	-20.32204815	17.3721293	5.23534993	4	0.698635105
8	cell15_G1	3	11.79346893	-17.97870008	16.15990302	3.97328939	4	0.754126656
9	cell16_G1	1	5.715911118	17.68960507	16.05039117	5.16091022	5	0.678455798
10	cell17_G1	1	4.789390628	19.75077371	16.5351234	4.2871718	5	0.740723326

Figure 15: Output of clusterNgriph

  - (B) B-cells (25K reads/cell), (M) Monocytes (100K reads/cell), (S) Stem cells (24.7K reads/cell), (NK) Natural Killer cells (29K reads/cell), (N) Naive T-cells (19K reads/cell)
- setB 100 cell randomly selected for each cell type:
  - (B) B-cells, (M) Monocytes, (H) **T-helper cells** (21K reads/cell), (NK) Natural Killer, (N) Naive T-cell
- setC 100 cell randomly selected for each cell type:
  - (C) **Cytotoxic T-cells** (28.6K reads/cell), (M) Monocytes, **T-helper cells**, (NK) Natural Killer, (N) Naive T-cell
- setD 100 cell randomly selected for each cell type:
  - (C) **Cytotoxic T-cells**, (NC) **Naive cytotoxic T-cells** (20K reads/cell), (H) **T-helper cells**, (NK) Natural Killer, (N) Naive T-cell

Moving from SetA to setD we added progressively cells coming from T-cell populations, making the cell-type partitioning more challenging because of the similarities between T-cell sub-populations.

We used PCA (Figure 16A-D) to visualize the dissimilarity between cells populations. PCA measures the variance between the elements of the dataset and the most important differences in variance are estimated by the PC1.

```
system("wget https://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
system("cd section4.1_examples")

#visualizing the complexity of the datasets using PCA
```

```
library(docker4seq)
topx(data.folder=getwd(),file.name="bmsnkn_5x100cells.txt",threshold=1000, logged=FALSE)
tmp <- read.table("bmsnkn_5x100cells_1000.txt", sep="\t", header=T, row.names=1)
tmp.n <- strsplit(names(tmp), "_")
tmp.n <- sapply(tmp.n, function(x)x[2])
tmp.n1 <- paste(tmp.n, seq(1:length(tmp.n)),"_",tmp.n, sep="")
names(tmp) <- tmp.n1
logtmp <- log10(tmp + 1)
write.table(logtmp, "log10_bmsnkn_5x100cells_1000.txt", sep="\t", col.names = NA)

pca(experiment.table="log10_bmsnkn_5x100cells_1000.txt", type="FPKM",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())

topx(data.folder=getwd(),file.name="bmHnkn_5x100cells.txt",threshold=1000, logged=FALSE)
tmp <- read.table("bmHnkn_5x100cells_1000.txt", sep="\t", header=T, row.names=1)
tmp.n <- strsplit(names(tmp), "_")
tmp.n <- sapply(tmp.n, function(x)x[2])
tmp.n1 <- paste(tmp.n, seq(1:length(tmp.n)),"_",tmp.n, sep="")
names(tmp) <- tmp.n1
logtmp <- log10(tmp + 1)
write.table(logtmp, "log10_bmHnkn_5x100cells_1000.txt", sep="\t", col.names = NA)

pca(experiment.table="log10_bmHnkn_5x100cells_1000.txt", type="FPKM",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())

topx(data.folder=getwd(),file.name="CmHnkn_5x100cells.txt",threshold=1000, logged=FALSE)
tmp <- read.table("CmHnkn_5x100cells_1000.txt", sep="\t", header=T, row.names=1)
tmp.n <- strsplit(names(tmp), "_")
tmp.n <- sapply(tmp.n, function(x)x[2])
tmp.n1 <- paste(tmp.n, seq(1:length(tmp.n)),"_",tmp.n, sep="")
names(tmp) <- tmp.n1
logtmp <- log10(tmp + 1)
write.table(logtmp, "log10_CmHnkn_5x100cells_1000.txt", sep="\t", col.names = NA)

pca(experiment.table="log10_CmHnkn_5x100cells_1000.txt", type="FPKM",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())

topx(data.folder=getwd(),file.name="CNCHnkn_5x100cells.txt",threshold=1000, logged=FALSE)
tmp <- read.table("CNCHnkn_5x100cells_1000.txt", sep="\t", header=T, row.names=1)
tmp.n <- strsplit(names(tmp), "_")
tmp.n <- sapply(tmp.n, function(x)x[2])
tmp.n1 <- paste(tmp.n, seq(1:length(tmp.n)),"_",tmp.n, sep="")
```

```
names(tmp) <- tmp.n1
logtmp <- log10(tmp + 1)
write.table(logtmp, "log10_CNCHnkn_5x100cells_1000.txt", sep="\t", col.names = NA)

pca(experiment.table="log10_CNCHnkn_5x100cells_1000.txt", type="FPKM",
    legend.position="topleft", covariatesInNames=TRUE, samplesName=FALSE,
    principal.components=c(1,2), pdf = TRUE,
    output.folder=getwd())
```

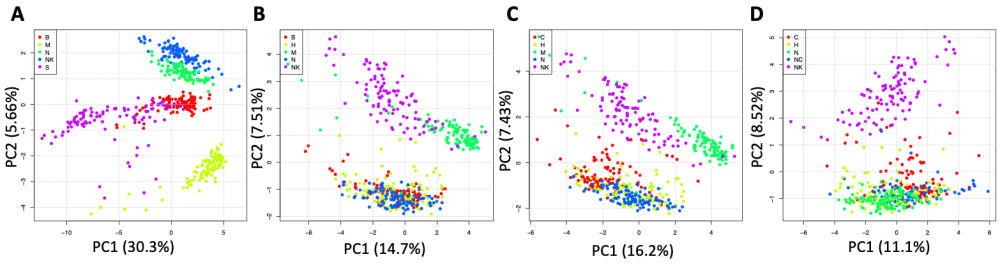


Figure 16: PCA is getting progressively unable to discriminate between the different cell subpopulations as the set of cells are getting functionally more similar to each other: A) PCA of setA, B) PCA of setB, C) PCA of setC, D) PCA of setD

PC1 shows that, as the differences between cell populations become smaller, moving from setA to setD, the aggregation in homogeneous groups of cells is compromised (Figure 16A-D).

To observe the effect of the reduced dissimilarity between populations on the stability of the number of clusters, we use the rCASC **clusterNgriph** function on the above mentioned 4 datasets using 160 permutations/each, and randomly removing in each permutation 10% of the cells. Each analysis took approximately 60 mins on a SeqBox hardware.

```
#downloading datasets
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")

#setA
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "bmsnkn_5x100cells.txt", sep="/"), nPerm=160, permAtTime=8,
              percent=10, separator="\t",logTen=0, seed=111)

#setB
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "bmHnkn_5x100cells.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)

#setC
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "CmHnkn_5x100cells.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)
```

```
#setD
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "CNCHnkn_5x100cells.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)
```

It is notable that as the differences between the cells populations is narrowing (i.e in setA cells types are quite different in overall functional activity, as in setD four out of five cell types are T-cells sub-populations) the fluctuations in the detected number of clusters increase. In setA, where PCA is able to discriminate between the five cell populations (Figure 16A), out of 160 bootstraps only 1 gave a number of clusters different from the effective number of cell sub-populations (Figure 17A). In all the other three subsets perturbations result in a higher number of events in which number of clusters different from 5 (Figure 17B-D).

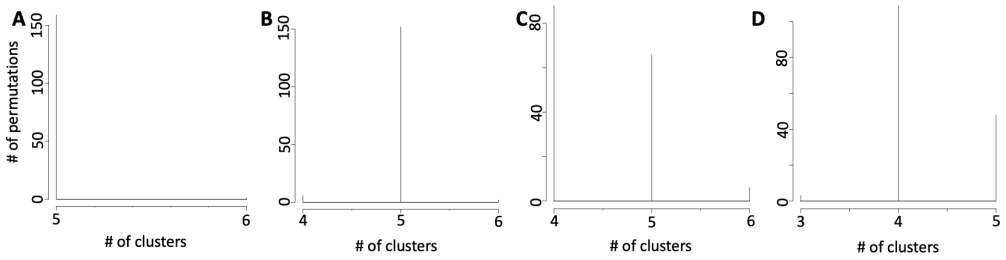


Figure 17: Clusters number is dependent by the cell type similarity: A) number of clusters detectable by griph in setA, B) number of clusters detectable by griph in setB, C) number of clusters detectable by griph in setC, D) number of clusters detectable by griph in setD

As highlighted in **Section 3.2.1**, the difficulties in detecting a stable number of clusters, upon dataset perturbation, is due to the limited number of detected genes. To further support this hypothesis we run a comparison between the most expressed genes in the cell types used in the above example.

```
#downloading datasets
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")

#loading the datasets used to generate PCA
b <- read.table("cd19_b_100cell.txt", sep="\t", header=T, row.names=1)
mono <- read.table("cd14_mono_100cell.txt", sep="\t", header=T, row.names=1)
stem <- read.table("cd34_stem_100cell.txt", sep="\t", header=T, row.names=1)
nk <- read.table("cd56_nk_100cell.txt", sep="\t", header=T, row.names=1)
naiveT <- read.table("naiveT_100cell.txt", sep="\t", header=T, row.names=1)
cyto <- read.table("cytoT_100cell.txt", sep="\t", header=T, row.names=1)
naiveCyto <- read.table("naiveCytoT_100cell.txt", sep="\t", header=T, row.names=1)
helper <- read.table("cd4_h_100cell.txt", sep="\t", header=T, row.names=1)

#calculating the gene-level expression and ranking the genes from the most expressed to the least expressed
b.s <- sort(apply(b,1,sum), decreasing=T)
mono.s <- sort(apply(mono,1,sum), decreasing=T)
stem.s <- sort(apply(stem,1,sum), decreasing=T)
```

```
nk.s <- sort(apply(nk,1,sum), decreasing=T)
naiveT.s <- sort(apply(naiveT,1,sum), decreasing=T)
cyto.s <- sort(apply(cyto,1,sum), decreasing=T)
naiveCyto.s <- sort(apply(naiveCyto,1,sum), decreasing=T)
helper.s <- sort(apply(helper,1,sum), decreasing=T)

#function that measure the identity between lists of increasing lengths
overlap <- function(x,y){
  overlap.v <- NULL
  for(i in 1:length(x)){
     overlap.v[i] <- length(intersect(x[1:i], y[1:i]))
  }
  return(overlap.v)
}

#calculating the level of identity between lists of increasing lengths all comparisons are run with respect to naive T-cell.
naiveCyto.naiveT <- overlap(names(naiveT.s), names(naiveCyto.s))
b.naiveT <- overlap(names(naiveT.s), names(b.s))
mono.naiveT <- overlap(names(naiveT.s), names(mono.s))
stem.naiveT <- overlap(names(naiveT.s), names(stem.s))
nk.naiveT <- overlap(names(naiveT.s), names(nk.s))
naiveCyto.naiveT <- overlap(names(naiveT.s), names(naiveCyto.s))
helper.naiveT <- overlap(names(naiveT.s), names(helper.s))
cyto.naiveT <- overlap(names(naiveT.s), names(cyto.s))

#plotting the above data
plot(seq(1, 500), seq(1, 500), type="l", col="black", lty=2)
points(seq(1, 500), naiveCyto.naiveT[1:500], type="l", col="black")
points(seq(1, 500), cyto.naiveT[1:500], type="l", col="blue")
points(seq(1, 500), helper.naiveT[1:500], type="l", col="green")
points(seq(1, 500), mono.naiveT[1:500], type="l", col="red")
points(seq(1, 500), b.naiveT[1:500], type="l", col="brown")
points(seq(1, 500), stem.naiveT[1:500], type="l", col="orange")
points(seq(1, 500), nk.naiveT[1:500], type="l", col="violet")
legend("topleft", legend=c("Naive T-cytotoxic", "T-cytotoxic", "T-helper", "Monocytes", "B-cells", "Stem cells", "NK"),
       pch=15, col=c("black", "blue", "green", "red", "brown", "orange", "violet"))
```

Figure 18 shows the number of identical genes found in common between naive T-cells and the other sub-populations in setA and setD, using lists of increasing size and ordered by expression level. The plot shows that naive T-cytotoxic, T-cytoxic and T-helper, from setD, share with naive T-cells, within the top 500 most expressed genes, more genes with respect to the other cell types present in setA. Thus, the lack of cell-type specific genes between 4 out 5 cell types in SetD dataset negatively affect the stability dataset partitioning, upon bootstraps.

The results described in this section indicate that **clusterNgriph** is a valuable instrument to define a range of numbers of clusters to be further investigated with supervised clustering approaches.

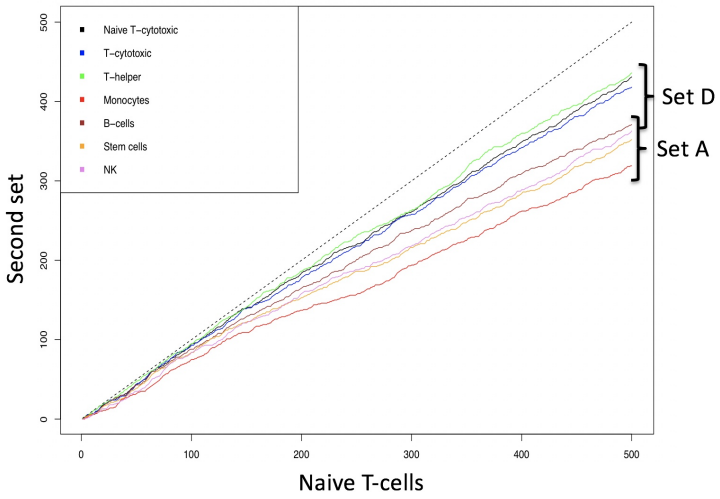


Figure 18: Identity between naive T-cells and the other cell types in set A and D in gene lists of increasing length. Identity between two data sets is shown by the dashed line.

Section 5 K-mean clustering: detecting cell sub-populations by mean of kernel based similarity learning (*SIMLR*).

The number of clustering and dimension reduction methods for single cell progressively increased over the last few years. Last year *Wang and coworkers* published SIMLR, a framework which learns a similarity measure from single-cell RNA-seq data in order to perform dimensions reduction. We decided to select this method as core clustering tool in rCASC, because outperformed eight methods published before 2017 [*Wang and coworkers*].

Specifically, we use SIMLR as clustering method recording the effects of data perturbation, i.e. removal of random subset of cells, on the clustering structure. Although, we think SIMLR provides important advantages with respect to other clustering methods, rCASC framework can embed also other data reduction tools. At the present time, tSne is also implemented within the rCASC data permutation framework.

One of the peculiarity of rCASC is the user tunable bootstrap procedure. rCASC represents bootstrap results via a cell stability score (Figure 19). In brief, a set of cells to be organized in clusters (Figure 19A) is analysed with SIMLR, applying a users defined k number of clusters (Figure 19B). A user defined % of cells is removed from the original data set and these cells are clustered again (Figure 19C). The clusters obtained in each bootstrap step are compared with the clusters generated on the full dataset using Jaccard index (Figure 19D-E). If the Jaccard index is greater of a user defined threshold, e.g. 0.8, the cluster is called confirmed in the bootstrap step (Figure 19F). Then to each cell, belonging to the confirmed cluster, *cell stability score* value is increased of 1 unit (Figure 19G). At the end of the bootstrap procedure, cells are labeled with different symbols describing their *cell stability score* in a specific cluster (Figure 19H).

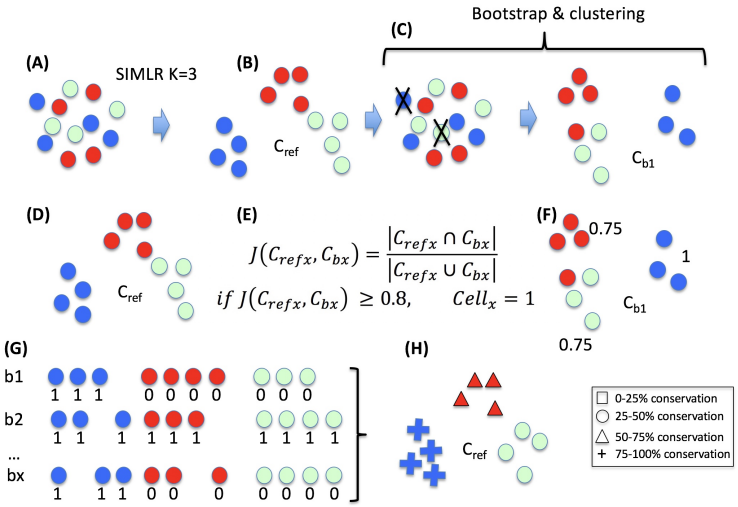


Figure 19: Cell stability score

Section 5.1 Cell Stability Score: mathematical description.

Stability score Algorithm Let be C the count matrix with $N \times M$ dimension where N is the gene number and M is cell number.

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots \\ \vdots & \ddots & \\ c_{N1} & & c_{NM} \end{bmatrix}$$

Then we define C_q^p as a matrix derived by matrix C in which q columns (randomly selected in permutation p) are removed. The function $RemoveCell(p)$ takes as input a specific permutation p and returns a list L^p containing all the removed cells in p(observe that the size of L^p is q). We denote $\mathbf{cl}^p$ the vector with length $M - q$ storing the output of the clustering algorithm in the permutation p. Hence, $\mathbf{cl}^p[i]$ identified the cluster in which the ith cell is inserted in permutation p. Moreover we use the notation **cl** for indicating the the output of the clustering algorithm obtained by all cells.

The relation symmetric matrix R^p with dimension $M \times M$ is defined as follows:

$$R^p = \begin{bmatrix} r_{1,1}^p & r_{1,2}^p & \dots \\ \vdots & \ddots & \\ r_{M,1}^p & & r_{M,M}^p \end{bmatrix}$$

where $r_{i,j}^p$ is:

$$r_{i,j}^p = \begin{cases} 1 \; if \; \mathbf{cl}^p[i] = \mathbf{cl}^p[j] \\ 0 \; otherwise \end{cases}$$

Similarly we defined R relation symmetric matrix obtained considering all cells. Then we introduce a function $RMV()$ that takes as input R and the list of removed cells in a permutation (i.e. l^p) and returns an new matrix R' in which the columns and the rows associated with cells in l^p are removed.

Function $length(j, p, k)$ counts the occurrences of k in the row j of the matrix $R + R^p$. It is formally defined as follows:

$$length(j, p, k) = \sum_i = 1^{M-|L^p|} 1_{R[i,j]} + R^p[i, j] = k$$

where 1_A is an indicator function returning 1 when condition A is satisfied.

Finally we define a permutation score $pscore_{j,p}$ as:

$$pscore_{j,p} = \frac{length(j, p, 2)}{length(j, p, 2) + length(j, p, 1)}$$

where p is a permutation and j a cell.

We define $tscore_{m,s}$ as follow

$$tscore_{m,s} = \frac{1}{P} \sum_{p \in P} 1_{score_{m,p} >= s}$$

where P is the permutation number and s the threshold score.

1 Example

Be $\mathbf{cl} = \begin{Bmatrix} 1 & 2 & 2 & 1 & 2 & 1 \end{Bmatrix}$

Be $\mathbf{L} = \begin{Bmatrix} 6 & 2 & 2 & 4 \end{Bmatrix}$

Be $\mathbf{cl}^1 = \begin{Bmatrix} 1 & 2 & 1 & 1 & 2 \end{Bmatrix}$

Be $\mathbf{cl}^2 = \begin{Bmatrix} 1 & 2 & 1 & 2 & 2 \end{Bmatrix}$

Be $\mathbf{cl}^3 = \begin{Bmatrix} 1 & 2 & 1 & 1 & 2 \end{Bmatrix}$

Be $\mathbf{cl}^4 = \begin{Bmatrix} 1 & 2 & 2 & 1 & 2 \end{Bmatrix}$

$\forall\ p \in \{1, 2, 3, 4\},\ R_p$ is calculated

for istance hereafter I reported R and R^1

$$R = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$R^1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$\forall\ p\ R' + R^p$ *is calculated*

for istance hereafter I reported R' + R^1

$$R' + R^1 = \begin{bmatrix} 2 & 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 0 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 2 & 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 0 & 2 \end{bmatrix}$$

$\forall\ p,\ pscore\ is\ evaluated\ with\ S = 0.6$ for istance hereafter I reported $pscore_1$

$$pscore_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

This means that in permutation 1 the third cell is unstable "jumping" from cluster number 1 to cluster number 2 in P. $\text{tscore}_{m,s}$ is evaluated then for each cell $tscore_s = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.25 \\ 0.25 \\ 0 \end{bmatrix}$

In this Example number of permutation is 4, for statistical relevance, an higher number of permutation is required.

SIMLR is embedded in **simlrBootstrap** function within the rCASC bootstrap framework.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path of the file, with file name and extension included
  - *nPerm*, number of permutations to be executed
  - *permAtTime*, number of permutations computed in parallel
  - *percent*, percentage of randomly selected cells removed in each permutation
  - *range1*, beginning of the range of clusters to be investigated
  - *range2*, end of the range of clusters to be investigated
  - *separator*, separator used in count file, e.g. '\t', ','
  - *logTen*, 1 if the count matrix is already in log10, 0 otherwise
  - *seed*, important value to reproduce the same results with same input, default is 111
  - *sp*, minimun number of percentage of cells that has to be in common in a cluster, between two permutations, default 0.8
  - *clusterPermErr*, probability error in depicting the number of clusters in each permutation, default = 0.05

```
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")
library(rCASC)
#annotating data setA
#annotating data setA
system("wget ftp://ftp.ensembl.org/pub/release-94/gtf/homo_sapiens/Homo_sapiens.GRCh38.94.gtf.gz")
system("gzip -d Homo_sapiens.GRCh38.94.gtf.gz")
system("mv Homo_sapiens.GRCh38.94.gtf genome.gtf")

scannobyGtf(group="docker", file=paste(getwd(),"bmsnkn_5x100cells.txt",sep="/"),
                    gtf.name="genome.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE,umiXgene=3)
#running SIMLR analysis using the range of clusters suggested by clusterNgriph in session 4.2
simlrBootstrap(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
               nPerm=160, permAtTime=8, percent=10, range1=4, range2=6,
               separator="\t",logTen=0, seed=111)

#annotating data setB
scannobyGtf(group="docker", file=paste(getwd(),"bmHnkn_5x100cells.txt",sep="/"),
                    gtf.name="genome.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE,umiXgene=3)
#running SIMLR analysis using the range of clusters suggested by clusterNgriph in session 4.2
simlrBootstrap(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(), "annotated_bmHnkn_5x100cells.txt", sep="/"),
               nPerm=160, permAtTime=8, percent=10, range1=4, range2=6,
               separator="\t",logTen=0, seed=111)
```

The output of **simlrBootstrap** function is described in Figure 20. The output is localized in the *Results* folder (Figure 20 Folder 1), which is created in the folder where the analysis started. This folder contains the count matrices used in the analyses. In this example the count matrices are those belonging to setA and B (see Section 4.1). In the *Results* folder are also present folders labeled with the name of the count matrix used in the analysis (Figure 20 Folder 2, annotated_bmsnkn_5x100cells, annotated_bmHnkn_5x100cells). In each of these folders there are a set of folders indicated with a number that refers to the analysed range of number of clusters (Figure 20 Folder 3). In this specific example the range of clusters goes from 4 to 6. In each *NameOfCountMatrix* folder there is a file called **_Stability_Violin_Plot.pdf** (Figure 20A) which represents the distribution of the cells stability scores over the bootstraps in the range of number of clusters investigated. In this example, it is clear that the analysis done with 5 clusters is the one providing the highest stability of cells within each cluster. The other plot that is also available in the folder (Figure 20 Folders 4) is **NameOfCountMatrix_vioplot.pdf**, Figure 20B, which contains the distribution of the *Silhuette* values for each cluster over the bootstraps. *Silhuette* value is a measure of computation cluster stability, and evaluates how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Clearly the information provided by Silhuette plot is much less informative for the definition of the optimal clustering number with respect to the information provided by the cells stability score (Figure 20 blue arrow). In each clustering folder there is a pdf named *NameOfCountMatrix_Stability_Plot.pdf*, which contains two plots (Figure 20C-D) generated with the clustering program. The plot in Figure 20C provides a 2D view of the clustering results. In this plot each cell is labeled with a symbol indicating its cell stability score. In this specific example of the plot generated with 5 clusters shows that all cells remain in a cluster between 75 to 100% of the bootstraps (+ symbol). The plot, Figure 20D, shows the genes detectable in each cell in function of the total number of reads/cell. In this plot cells are colored with the same color of their belonging cluster. This plot is useful to observe if the clustering is biased by the number of genes called in each cluster. In this specific example, only the blue cluster is charaterized by a number of detected genes, which is larger of those detectable in the other clusters (Figure 20D). This is expected, since blue cluster is made of Monocytes, which have been sequenced to 100K reads/cells, as all other cells in setA were sequenced between 19 to 29K reads/cell, see **Section 4.2**.

In Figure 21 are shown the effects, on cell stability, in SIMLR analysis done with 4 and 6 clusters for SetA (*annotated_bmsnkn_5x100cells.txt*) counts matrix. Using 4 as number of clusters, Figure 21A, some of the cells in each cluster show a reduced stability (50-75%, triangle) and arrows highlights cells characterized by a cell stability score between 25 to 50%. The circles in Figure 21B show the clusters where the full cluster has a cell stability between 50 to 75%. This observation indicates that the stability score is a useful measure to identify the optimal number of partitions to be used, i.e. the highest cell stability score was observed when five clusters were selected, corresponding to the number of cell types combined in SetA, see **Section 4.1**.

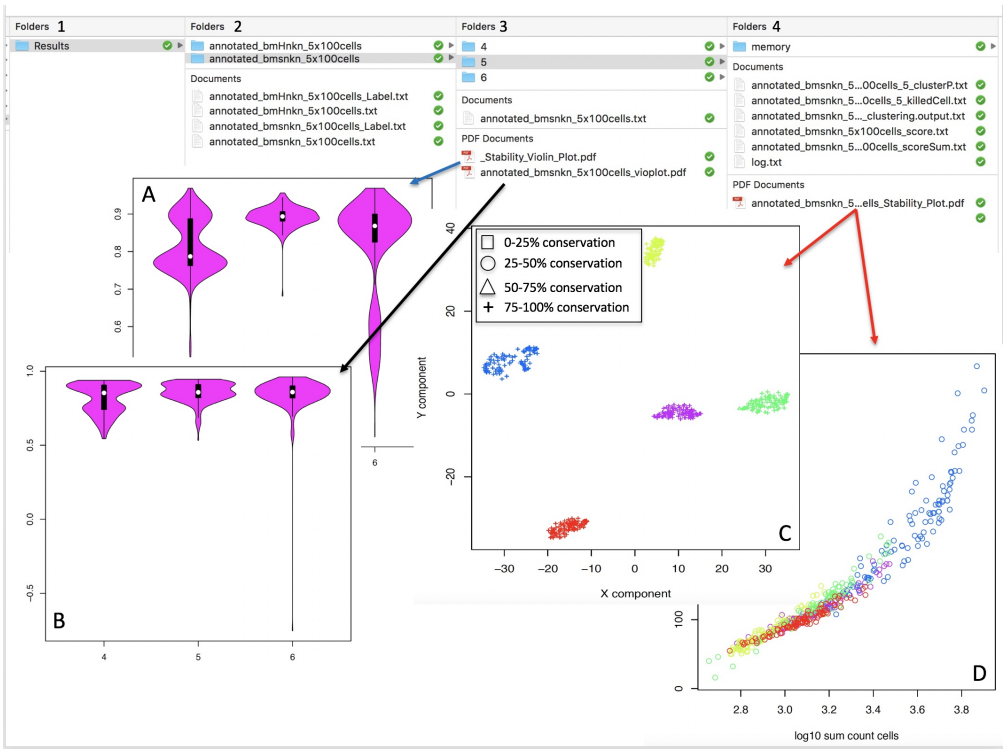


Figure 20: simlrBootstrap output

Section 5.2 Visualizing the cell clusters relocation during bootstraps.

The function **permutationMovie** allows the generation of a video showing the relocation of cells at each bootstrap. In this example we use the results shown in Figure 21B, where circles show the clusters where all cells have a cell stability between 50 to 75%.

- *Parameters*
  - *scratch.folder*, path of the scratch folder
  - *file*, path of the file, with file name and extension included
  - *separator*, separator used in count file, e.g. '\t', ','
  - *framePP*, number of frames for each permutation
  - *permutationNumber*, number of random permutations, must be less or the same value of the total permutations dome in simlrBootstrap

```
system("wget http://130.192.119.59/public/test_permutationvideo.zip")
unzip("test_permutationvideo.zip")
setwd("test_permutationvideo")
library("rCASC")
bootstrapsVideo(group="docker", scratch.folder="/data/scratch",
                file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
                nCluster=6, separator="\t", framePP=200, permutationNumber=80)
```

The video was generated in 8 minutes on SeqBox hardware. The video is saved in the cluster folder used for

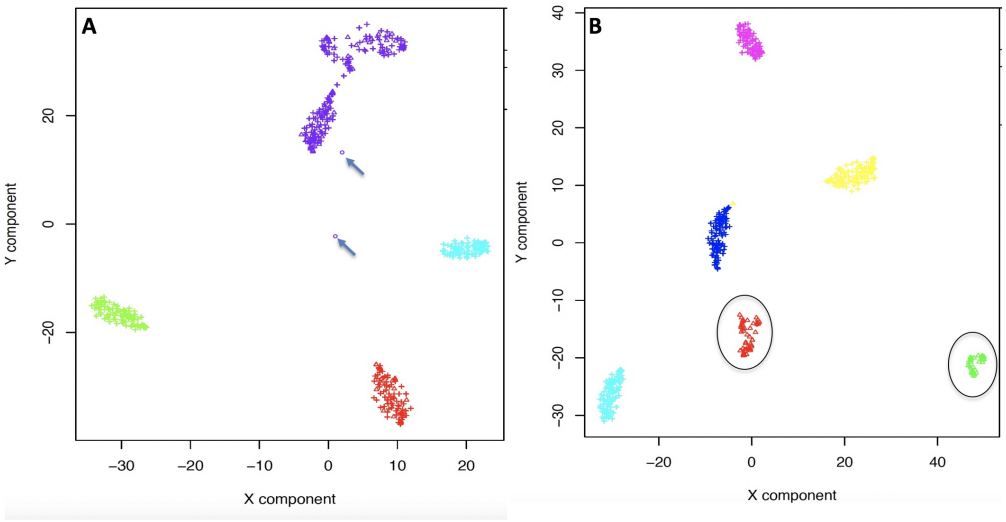


Figure 21: Clustering output including cell stability score

the analysis and it has the name **outputname.mp4**. The video generated with the above script is accessible *here*.

In Figure 22A is shown a screenshot of the output of **bootstrapsVideo**. The output of this function provides extra information with respect to the standard output of the **simlrBootstrap** and **tsneBootstrap**, since the video provides an overview of the area (colored circles) in which the cells of a specific cluster are localized as consequence of the bootstraps. This visualization provides a better description of the maximum size of the clusters and simplify the identification of clusters that are more near to each other, because of clusters structure, Figure 22A. In this specific example, is notable that, even if the yellow cluster has a high cell stability score, Figure 22B, its size is much greater of those observable for all the other clusters, because few cells with lower cell stability score (50-75%, triangle) are located very near to the blue cluster, arrow in Figure 22B.

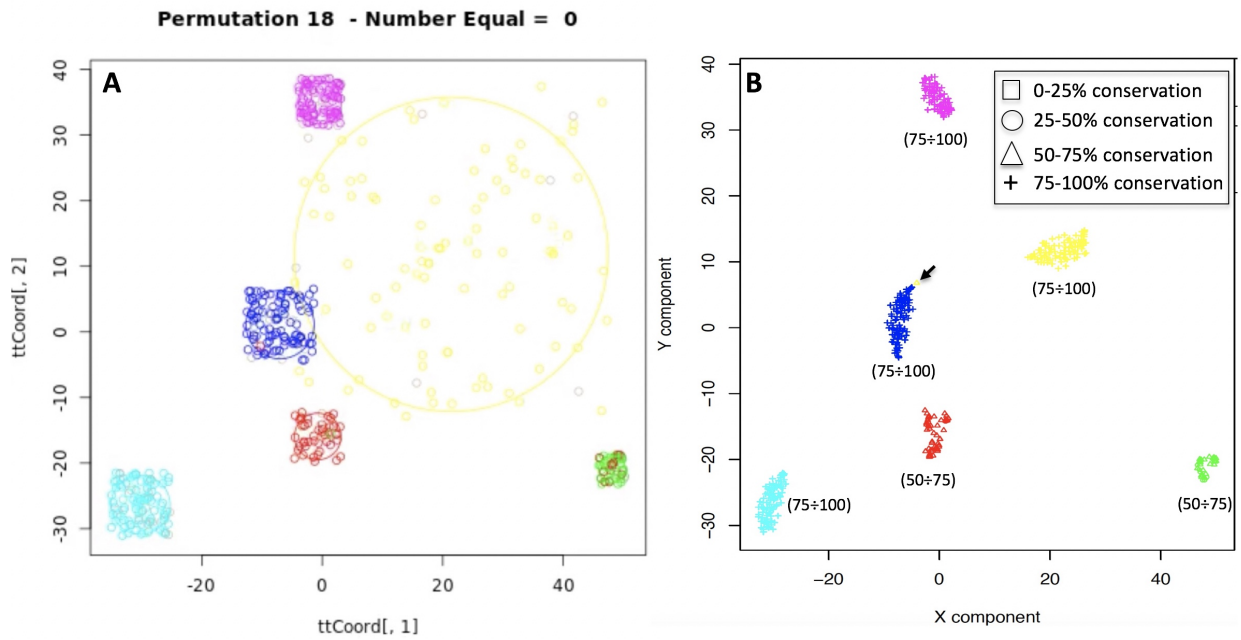


Figure 22: Comparing the output of bootstrapsVideo with respect to the one produced by simlrBootstrap. A) Output of bootstrapsVideo. B) simlrBootstrap output.

Section 6 Hierarchical clustering: *Seurat* graph-based clustering.

Recently *Freytag et al.* and *Do et al.* described, in their independent comparison papers, that *Seurat* delivered the overall best performance in cells clustering. Thus, we decided to include in our workflow also this clustering tool. Seurat clusters cells based on their PCA scores, with each PC essentially representing a 'metagene' that combines information across a correlated gene set. The first step of the analysis is the identification of the how many PCs have to be to included.

seuratPCAEval function allows the exploration of PCs to determine relevant sources of heterogeneity and to define the range of PCs to be used.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path of the file, with file name and extension included
  - *separator*, separator used in count file, e.g. '\t', ','
  - *logTen*, 1 if the count matrix is already in log10, 0 otherwise
  - *seed*, important value to reproduce the same results with same input, default is 111

```
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")
library(rCASC)
#defining the PC threshold for clustering.
seuratPCAEval(group="docker",scratch.folder="/data/scratch/",
              file=paste(getwd(), "bmsnkn_5x100cells.txt", sep="/"),
```

```
            separator="\t", logTen = 0, seed = 111)
```

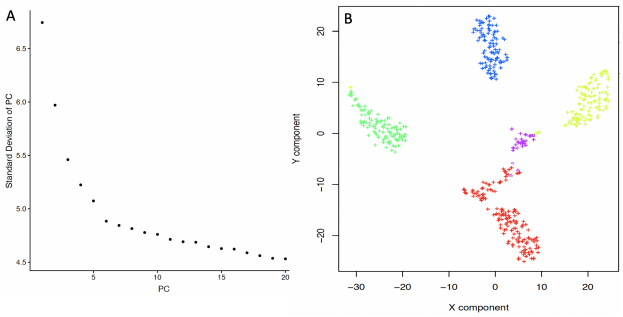


Figure 23: Seurat clustering, A) Seurat dataset PC component. B) Clustering results

The function generate a **Result** folder in which is present a folder with the same name of the dataset under analysis, in this specific case "bmsnkn_5x100cells". In the latter folder is present a PCA plot (PCE_bowPlot.pdf, Figure 23A), which allow the identification of the PCs to use, defining a cutoff where there is a clear elbow in the graph. In this example, it looks like the elbow would fall around PC 6.

seuratBootstrap function executes the seurat clustering (data reduction method PCA) and estimates cell stability score.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, a character string indicating the path of the scratch folder
  - *file*, a character string indicating the path of the file, with file name and extension included
  - *nPerm*, number of permutations to be executed
  - *permAtTime*, number of permutations computed in parallel
  - *percent*, percentage of randomly selected cells removed in each permutation
  - *separator*, separator used in count file, e.g. '', ','
  - *logTen*, 1 if the count matrix is already in log10, 0 otherwise
  - *pcaDimensions*, 0 for automatic selection of PC elbow.
  - *seed*, important value to reproduce the same results with same input

```
seuratBootstrap(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(), "bmsnkn_5x100cells.txt", sep="/"),
                nPerm=160, permAtTime=8, percent=10, separator="\t",logTen=0, pcaDimensions=6, seed=111)
```

The function generate a **Result** folder in which is present a folder with the same name of the dataset under analysis, in this specific case "bmsnkn_5x100cells.txt". In the latter folder is present a folder with the number of cluster detected by Seurat, in this specific case **5**. In **5** folder there is the file with extension **__Stability__Plot.pdf**, which is the clustering picture with cells labeled with their stability score, Figure 23B). The file with the extension **__clusterP.txt**,containing for each permutation the location of each cell in the clusters. The file with the extension **__killedCell.txt**, where are saved the cells removed at each permutation. The file with the extension **__score.txt**, containing the scores assigned at each permutation to each cell. The file with extension **__scoreSum.txt** containing the Cell stability score for each cell.

The file with extension **__clustering.output.txt**, summarizing all information required to generate the **__Stability__Plot.pdf** plot.

Section 6.1 Comparing SIMLR, tSne and Seurat clustering

In Wang [*2017*] article it is demonstrated that SIMLR outperforms other data reduction methods (Figure **??**). *Freytag et al.* and *Do et al.* described that *Seurat* delivers the overall best performance in cells clustering. Here, we run a comparison between SIMLR, tSne and Seurat clustering (data reduction by PCA) within the rCASC bootstrap framework to observe how cells stability score is affected by the three clustering methods.

tSne is embedded in **tsneBootstrap** function within the rCASC bootstrap framework.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path of the file, with file name and extension included
  - *nPerm*, number of permutations to be executed
  - *permAtTime*, number of permutations computed in parallel
  - *percent*, percentage of randomly selected cells removed in each permutation
  - *range1*, beginning of the range of clusters to be investigated
  - *range2*, end of the range of clusters to be investigated
  - *separator*, separator used in count file, e.g. '\t', ','
  - *logTen*, 1 if the count matrix is already in log10, 0 otherwise
  - *seed*, important value to reproduce the same results with same input, default is 111
  - *sp*, minimun number of percentage of cells that has to be in common in a cluster, between two permutations, default 0.8
  - *clusterPermErr*, probability error in depicting the number of clusters in each permutation, default = 0.05
  - *perplexity*, number of close neighbors for each point. This parameter is specific for tSne. Default value is 10.

```
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")

system("wget ftp://ftp.ensembl.org/pub/release-94/gtf/homo_sapiens/Homo_sapiens.GRCh38.94.gtf.gz")
system("gzip -d Homo_sapiens.GRCh38.94.gtf.gz")
system("mv Homo_sapiens.GRCh38.94.gtf genome.gtf")

library(rCASC)
#annotating data setA

scannobyGtf(group="docker", file=paste(getwd(),"bmsnkn_5x100cells.txt",sep="/"),
                    gtf.name="genome.gtf", biotype="protein_coding",
                    mt=TRUE, ribo.proteins=TRUE,umiXgene=3)
```

```
#running tSne analysis using the range of clusters suggested by clusterNgriph in session 4.2
tsneBootstrap(group="docker",scratch.folder="/data/scratch/",
              file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
              nPerm=160, permAtTime=8, percent=10, range1=4, range2=6, separator="\t",
              logTen=0, seed=111, sp=0.8, perplexity=10)
# required time 159 mins

#running SIMLR analysis using the range of clusters suggested by clusterNgriph in session 4.2
simlrBootstrap(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
               nPerm=160, permAtTime=8, percent=10, range1=4, range2=6,
               separator="\t",logTen=0, seed=111)
# required time 168 mins

#running Seurat clustering: defining the PC threshold for clustering.
seuratPCAEval(group="docker",scratch.folder="/data/scratch/",
              file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
              separator="\t", logTen = 0, seed = 111)
#running Seurat clustering with bootstrap: threshold was detected at PC 6, see Section 6.
seuratBootstrap(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(), "annotated_bmsnkn_5x100cells.txt", sep="/"),
                nPerm=160, permAtTime=8, percent=10, separator="\t",logTen=0, pcaDimensions=6, seed=111)
# required time 194 mins
```

We run the analysis on the SetA described in Section 4.1. Both SIMLR and tSne are able to detect that five clusters provides the more stable organization in terms of cells stability score (Figure 24A-B). However, SIMLR clustering produces highly stable clusters (Figure 24C), as instead tSne clustering produces much less stable clusters (Figure 24D). From the point of view of the computation time the two clustering methods are relatively similar. The above mentioned analysis took 159 mins for tSne and 168 mins for SIMLR. Seurat clustering detected 5 clusters, Figure 23B, as SIMLR and tSne. The cell stability score for SIMLR (Figure 24C) and Seurat (Figure 23B) are superimposable and computation time is very similar to the other two methods.

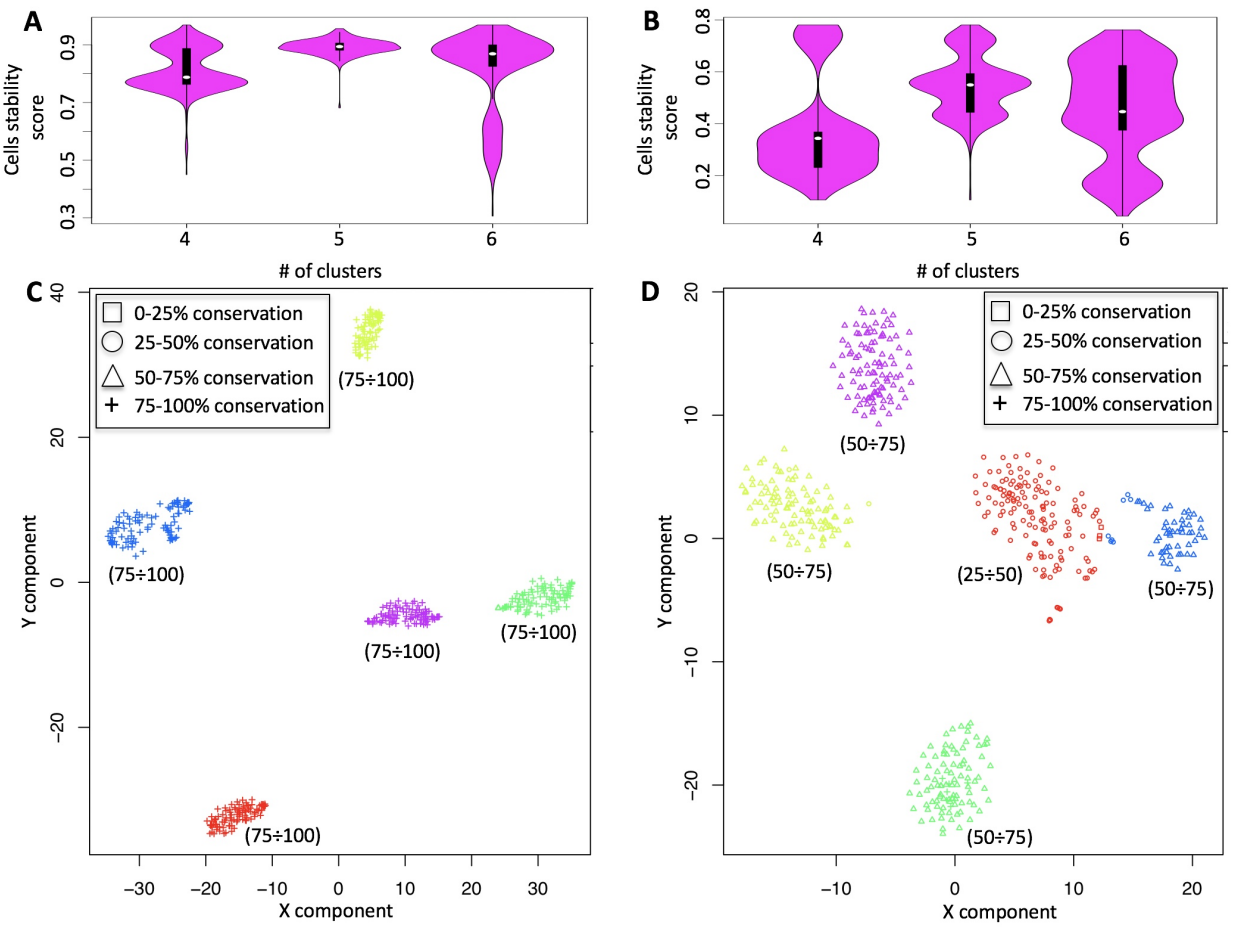


Figure 24: Comparing SIMLR and tSne within rCASC bootstrap framework. A,C) SetA clustered with SIMLR. B,D) SetA clustered with tSne.

Section 7 Detecting the genes playing the major role in clusters formation

Nowaday there are a lot of methods for the identification of differentally expressed genes between two populations of single-cell experiments [*Soneson et al.*]. However, the number of tools applicable to single-cells and allowing something similar to an ANOVA are very few. An ANOVA-like approach is described as applicable to single-cells in *edgeR Bioconductor package*. The edgeR ANOVA-like requires a comparison with respect to a reference set of cells. SIMLR and Seurat also provide ranking score for the genes, which are affecting mostly the clusters organization. SIMLR and Seurat gene ranking do not require a reference cluster for the analysis, thus providing wider applications with respect to ANOVA-like. The three methods are part of the rCASC framework.

Section 7.1 A statistical approach to select genes affecting clusters formation: EdgeR ANOVA-like

The function **anovaLike** executes edgeR ANOVA-like with the settings required for single-cells analysis, for more information please refer to the edgeR vignette.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *file*, a character string indicating the counts table file with the path of the file.
  - *sep*, separator used in count file, e.g. '\t', ','
  - *cluster.file*, a character string indicating the _clustering.output.txt file of interest, generated by bootstrapSimlar or bootStrapTsne. IMPORTANT this file must be located in the same folder where counts.table is placed
  - *ref.cluster*, a number indicating the cluster to be used a reference for anova-like comparison with the other clusters.
  - *logFC.threshold*, minimal logFC present in at least one of the comparisons with respect to reference covariate
  - *FDR.threshold*, minimal FDR present in at least one of the comparisons with respect to reference covariate
  - *logCPM.threshold*, minimal average abundance
  - *plot*, boolean, TRUE a plot of differentially expressed genes is generated

The input file is a counts table, which is modified by **anovaLike** adding the cluster number to cell name with an underscore. It is mandatory that no other underscores are located in the table. Then, counts table is reorganized to locate at the beginning of the table the set of cells belonging to the cluster used as reference group. The differentially expressed genes statistics for all genes are saved in the file with prefix **DE_**, followed by the counts table name. The filtered differentially expressed genes statistics, including only genes detected as differentially expressed in at least one of the clusters is saved with prefix **filtered_DE_** followed by the counts table name. **logFC_filtered_DE_** refers to the file containing only logFC extracted from **filtered_DE_** file. The count table is also saved reordered on the basis of cluster positions and has the extension **_reordered.txt**

```
system("wget http://130.192.119.59/public/annotated_setPace_10000_noC5_clustering.output.txt")
system("wget http://130.192.119.59/public/annotated_setPace_10000_noC5.txt.zip")
unzip("annotated_setPace_10000_noC5.txt.zip")

anovaLike(group="docker", data.folder=getwd(), counts.table="annotated_setPace_10000_noC5.txt",
       file.type="txt", cluster.file="annotated_setPace_10000_noC5_clustering.output.txt", ref.cluster=3,
       logFC.threshold=1, FDR.threshold=0.05, logCPM.threshold=4, plot=TRUE)

#the output of interest is logFC_filtered_DE_annotated_setPace_10000_noC5_reordered.txt
```

The function **clustersFeatures** selects the genes, which are more up-regulated in a specific cluster using **anovaLike** outputs.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *fileLogFC*, a character string indicating the absolute path to logFC_filtered_DE_ file generated by anovaLike function
  - *fileCounts*, a character string indicating the absolute path to reordered counts table file generated by anovaLike function
  - *delta*, the minimal distance between the max value of FC for a gene in a cluster of interest and the nearest other max FC in any of the other clusters. This value define the minimal distance with respect to the same gene in an other cluster to identify it as a cluster specific gene.
  - *sep*, separator used in count file, e.g. '\t', ','

This function focus only on upregulated genes because the ANOVA-like procedure compares all clusters with respect to a reference cluster. Thus, up-regulated genes are genes specific of the cluster under analysis as those down-regulated are characteristics of the reference cluster. The delta parameter describes the minimal distance existing, for a specific gene average logFC in a specific cluster, with respect to all the other clusters under analysis. The output of the function are four tab delimited files:

- the file with prefix **onlyUP_**, followed by the counts table name, i.e. the count table only containing the selected genes,
- the file with prefix**onlyUP_**, followed by **logFC_filtered_DE_**, the table containing logFC only for the selected genes,
- the file **onlyUP_clusters_specific_genes.txt**, which contains the list of specific genes associated with the corresponding cluster.
- the file **onlyUP_clusters_specific_genesSYMBOLs.txt**, which contains the list of specific genes SYMBOLS from onlyUP_clusters_specific_genes.txt. This file is used by function **hfc** to generate features specific heatmap.

```
system("wget http://130.192.119.59/public/clusters.features.zip")
unzip("clusters.features.zip")
setwd("clusters.features")

clustersFeatures(group="docker", data.folder=getwd(),
         logFC.table="logFC_filtered_DE_annotated_setPace_10000_noC5_reordered.txt",
         counts.table="annotated_setPace_10000_noC5_reordered.txt", delta=0.5)
```

Section 7.1.1 Visualizing cluster specific genes by heatmap.

The gene-features extracted by **clustersFeatures** cannot be used again in **simlrBootstrap**, if the overall number of genes is smaller of the cell number, for more information see SIMLR publication. This limitation does not apply to tsneBootstrap.

The function **hfc** allows to create a heatmap for the results generated from **simrBootstrap/tsneBootstrap** and the output of **clustersFeatures**.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, the path of the scratch folder
  - *file*, the path of the file, with file name and extension included
  - *nCluster*, n of the clusters on which to run the analysis. The function expect the presence of Result folder generated by **simrBootstrap/tsneBootstrap**
  - *separator*, separator used in count file, e.g. '\t', ','
  - *lfn*, name of the list of genes WITHOUT extension, if **clustersFeatures** function was used on anovaLike output the name of the file is "onlyUP_clusters_specific_geneSYMBOLs"
  - *geneNameControl*, 0 if the matrix has gene name without ENSEMBL geneID. 1 if the gene names is formatted like this : ENSMUSG00000000001:Gnai3.
  - *status*, 0 if is raw count, 1 otherwise
  - *b1*, the lower range of signal in the heatmap,for negative value write "/-5". To ask the function to do automatic value assignment set 0
  - *b2*, the upper range of signal in the heatmap,for negative value write "/-2". To ask the function to do automatic value assignment set 0

```
system("wget http://130.192.119.59/public/hfc.zip")
unzip("clusters.features.zip")
setwd("clusters.features")

hfc(group="docker",scratch.folder="/data/scratch",
       file=paste(getwd(),"annotated_setPace_10000_noC5.txt", sep="/"),
       nCluster=5, separator="\t", lfn="onlyUP_clusters_specific_geneSYMBOLs",
       geneNameControl=1, status=0, b1="/-1", b2="1")
```

The output is made of four pdf (Figure 25):

- *lfn-parameter*_hfc_log10.pdf counts in log10 format, e.g. onlyUP_clusters_specific_geneSYMBOLs_hfc_log10.pdf
- *lfn-parameter*_hfc_zscore.pdf Z score
- *lfn-parameter*_hfc_CSS.pdf cell cluster stability score
- *lfn-parameter*_hfc_all_genes.pdf

Section 7.2 A machine learning approach: SIMLR genes prioritization

SIMLR gene prioritization approach relies on the output similarity and does not depend on the downstream dimension reduction or clustering. The advantage of this step is that one can identify highly fluctuating genes that are consistent with the learned similarity in the multiple-kernel space.

The function **genesPrioritization** executes SIMLR gene prioritization within the rCASC bootstrap framework.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):

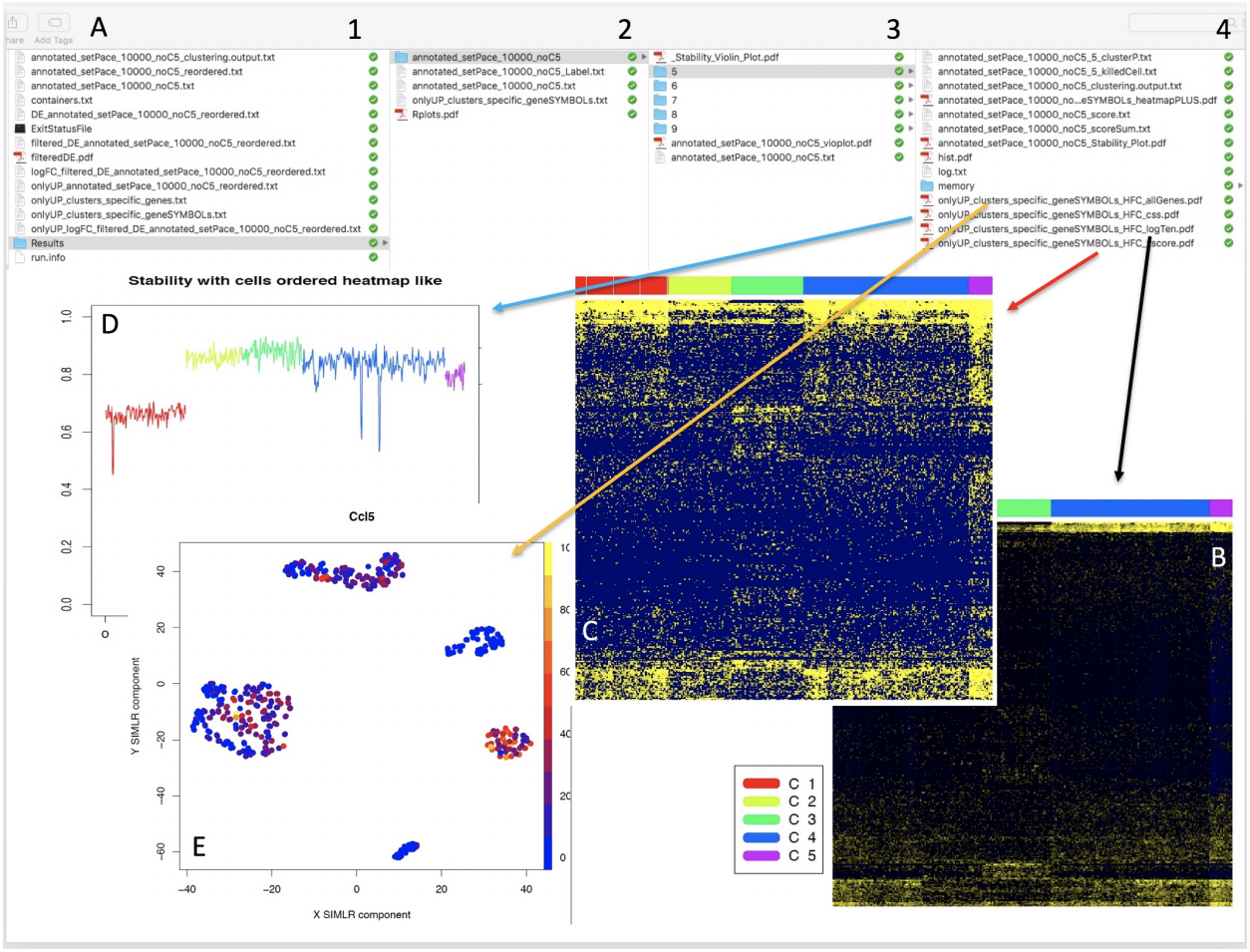


Figure 25: Output of hfc. A) Path of location of hfc function results. The output is located in cluster folder used for the analysis, B) Log10 counts heatmap. On the top of the figure clusters are represented as bars of different colors, C) Z-score heatmap, D) Cell Stability Score for each cell in each cluster, for more information see Section 5, E) SIMLR clustering output for each gene within those used to generate the heatmap. Cells are colored on the basis of the amount of CPMs for the gene under analysis.

- *scratch.folder*, the path of the scratch folder
- *file*, the path of the file, with file name and extension included
- *nPerm*, number of permutations to be executed
- *permAtTime*, number of permutations that can be computes in parallel
- *percent*, percentage of randomly selected cells removed in each permutation
- *nCluster*, the number of clusters, where to run prioritization
- *separator*, separator used in count file, e.g. '\t', ','
- *logTen*, 1 if the count matrix is already in log10, 0 otherwise
- *seed*, important value to reproduce the same results with same input
- *sp*, minimun number of percentage of cells that has to be in common between two permutation to be the same cluster, default 0.8.
- *clusterPermErr*, error that can be done by each permutation in cluster number depicting, de-

fault=0.05

```
system("wget http://130.192.119.59/public/annotated_setPace_10000_noC5.txt.zip")
unzip("annotated_setPace_10000_noC5.txt.zip")
genesPrioritization(group="docker",scratch.folder="/data/scratch/",
                    file=paste(getwd(),"annotated_setPace_10000_noC5.txt", sep="/"),
                    nPerm=160, permAtTime=8, percent=10, nCluster=5, separator="\t",
                    logTen=0, seed=111, sp=0.8, clusterPermErr=0.05)
```

The output of the function are a pdf named **geneRank.pdf** (Figure 26), containing the overall distribution of -log10 expression mean with respect to the Delta confidence of a gene being important for clustering, and a tab delimited file with the extension **__pvalList.txt** which contains, for each bootstrap, the gene importance score for clustering.

The function **genesSelection** allows to run a selection of the most interesting genes for clustering on the basis of two parameters:

- maxDeltaConfidence:
  - max value for Delta confidence for genes feature selection. Default is 0.01
- minLogMean:
  - min value for Log mean for genes feature selection. Default is 0.05

The above values can be set in the **genesSelection** function on the basis of the output of geneRank.pdf (Figure 26).

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, path of the scratch folder
  - *file*, the path of the counts file, with file name and extension included
  - *nCluster*, the number of clusters, where prioritization was run. In the working folder should be present the Result folder generated by **genesPrioritization** function.
  - *separator*, separator used in count file, e.g. ‘ ’, ‘,’
  - *seed*, important value to reproduce the same results with same input, default 111
  - *sp*, minimun number of percentage of cells that has to be in common in a cluster, between two permutations, default 0.8
  - *clusterPermErr*, probability error in depicting the number of clusters in each permutation, default = 0.05
  - *maxDeltaConfidence*, max value for Delta confidence for genes feature selection
  - *minLogMean*, min value for Log mean for genes feature selection

```
genesSelection(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(), "annotated_setPace_10000_noC5.txt", sep="/"),

#mv annotated_setPace_10000_noC5_clusters_specific_geneSYMBOLs.txt in the main folder

hfc(group="docker",scratch.folder="/data/scratch",
      file=paste(getwd(),"annotated_setPace_10000_noC5.txt", sep="/"),
      nCluster=5, separator="\t",
```

```
    lfn="annotated_setPace_10000_noC5_clusters_specific_geneSYMBOLs",
    geneNameControl=1, status=0, b1="/-1", b2="1")
```

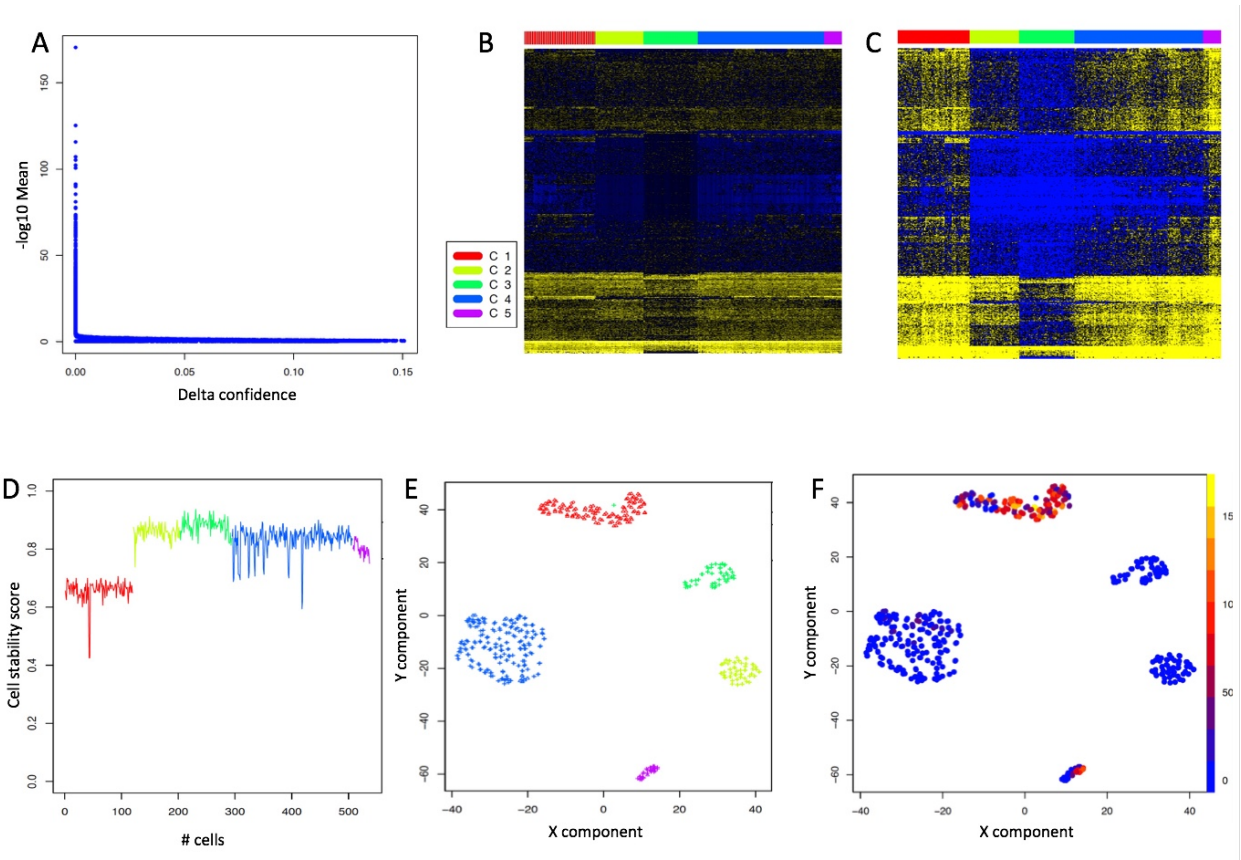


Figure 26: genePrioritization and genesSelection outputs. A) Prioritization plot, from this plot the maxDeltaConfidence (x axis) and the minLogMean (y axis), required by genesSelection can be extrapolated, i.e. in this case respectively 0.01 and 20. B) log10 counts heatmap from the output of genesSelection: 577 genes. C) Z-score heatmap. D) Cell stability score. E) simlrBootstrap output. F) Single gene CPM expression in SIMLR clusters. The expression is available for each of the genes from genesSelection output

The output of **genesSelection** is a file with extension **_clusters_specific_geneSYMBOLs.txt**, which can be used as input for** hfc** function, see **Section 6.1.1**. The list of prioritized genes cannot not be associated to a specific cluster as in the case of the anovaLike, see **Section 6.1**. Thus, more information will be provided on the cluster specificity from the heatmap visualization (Figure 26), see **hfc** function.

Section 7.3 Seurat genes prioritization

Seurat gene prioritization is performed by **seuratPrior** function.

- *Parameters* (only those without default; for the full list of parameters please refer to the function help):
  - *scratch.folder*, a character string indicating the path of the scratch folder
  - *file*, a character string indicating the path of the file, with file name and extension included

- *separator*, separator used in count file, e.g. ‘ ’, ‘,’
- *logTen*, 1 if the count matrix is already in log10, 0 otherwise
- *seed*, important value to reproduce the same results with same input
- *PCADim*, dimensions of PCA for seurat clustering
- *geneNumber*, numbers of specific genes for each clusters
- *nCluster*, number of cluster analysis.

```
system("wget http://130.192.119.59/public/section4.1_examples.zip")
unzip("section4.1_examples.zip")
setwd("section4.1_examples")

library(rCASC)

#running Seurat PC distribution
seuratPCAEval(group="docker",scratch.folder="/data/scratch/",
              file=paste(getwd(), "bmsnkn_5x100cells.txt", sep="/"),
              separator="\t", logTen = 0, seed = 111)

#running Seurat clustering with bootstrap
seuratBootstrap(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(), "bmsnkn_5x100cells.txt", sep="/"),
                nPerm=160, permAtTime=8, percent=10, separator="\t",logTen=0, pcaDimensions=6, seed=111)

#running Seurat clustering with bootstrap: elbow was detected at PC 6, see Section 6.
seuratPrior(group="docker", scratch.folder="/data/scratch/",
            file=paste(getwd(), "bmsnkn_5x100cells.txt", sep="/"), separator="\t",
            logTen=0, seed=111, PCADim=6, geneNumber=20, nCluster=5)
```

The output is a file with the prefix **Markers_**

	p_val	avg_logFC	pct.1	pct.2	p_val_adj	cluster	gene
1	3.22E-52	1.5362096	0.921	0.344	1.05E-47	1	ENSG00000111716
2	3.74E-46	0.844168	0.993	0.834	1.22E-41	1	ENSG00000122406
3	1.36E-45	2.2045356	0.57	0.034	4.46E-41	1	ENSG00000198851
4	2.52E-43	2.0700739	0.503	0.011	8.24E-39	1	ENSG00000167286
5	9.59E-26	1.3982863	0.589	0.149	3.14E-21	1	ENSG00000142546
6	1.17E-25	0.9933811	0.391	0.034	3.83E-21	1	ENSG00000166681
7	1.77E-25	0.845277	0.901	0.544	5.81E-21	1	ENSG00000181163
8	1.17E-23	1.0833788	0.821	0.367	3.83E-19	1	ENSG00000133872
9	2.47E-23	1.4599244	0.377	0.046	8.07E-19	1	ENSG00000139193
10	3.27E-19	0.8650031	0.338	0.04	1.07E-14	1	ENSG00000078596
11	1.83E-17	0.798671	0.709	0.301	5.99E-13	1	ENSG00000136732
12	5.46E-17	0.8894942	0.285	0.032	1.79E-12	1	ENSG00000237943
13	2.65E-16	0.7977282	0.649	0.255	8.66E-12	1	ENSG00000100380
14	1.61E-14	0.9703806	0.318	0.063	5.27E-10	1	ENSG00000100100
15	3.35E-14	0.9499176	0.311	0.063	1.10E-09	1	ENSG00000185198
16	1.42E-13	1.0419252	0.305	0.063	4.64E-09	1	ENSG00000257698
17	5.14E-12	0.8403336	0.483	0.189	1.68E-07	1	ENSG00000179144
18	7.95E-12	0.8991283	0.358	0.112	2.60E-07	1	ENSG00000182866
19	1.44E-11	0.8572148	0.444	0.158	4.72E-07	1	ENSG00000188404
20	6.99E-09	0.9712438	0.695	0.593	2.29E-04	1	ENSG00000171223
21	6.93E-88	3.0576497	0.928	0.041	2.27E-83	2	ENSG00000105369
22	1.15E-80	2.7387083	0.91	0.062	3.76E-76	2	ENSG00000007312
23	1.89E-60	2.2059988	0.613	0.003	6.20E-56	2	ENSG00000156738

Figure 27: Seurat gene prioritization

Section 8 An example of rCASC analysis: *GEO:GSE106264*.

Pace et al. explored the role of histone methyltransferase Suv39h1 in murine CD8+ T cells activated after Listeria monocytogenes infection. They showed that Suv39h1 controls the expression of a set of stem cell-related memory genes and its silencing increases long-term memory reprogramming capacity. The single-cell sequencing data from the *Pace's Science paper*:

- naive CD8+ T lymphocytes, here named **N**
- CD8+ T cells activated after Listeria monocytogenes infection, here named **NA**
- naive Suv39h1-defective CD8+ T lymphocytes, here named **Nd**
- Suv39h1-defective CD8+ T cells activated after Listeria monocytogenes infection, here named **NdA**

Section 8.1 Selecting subsets of cells with the highest number of called genes.

On the basis of our observation in **Section 3.2.1**, we selected, for each of the above mentioned groups, the cells with higher number of total reads/cell. Specifically we analysed a total of 600 cells, combining respectively 50 cells from **N** and **Nd** and 250 cells for **NA** and **NdA**. The rational of the different number of cells for the naive and the activated is due to the expectation that activated cells will be organize in mutiple clusters and naive are expected to be homogeneous. Cell labels in the counts table were modified adding the groups *N*, *Nd*, *NA* and *NdA*.

Using this dataset we tried to address the following questions:

1. Does Suv39h1 silence gene expression programs in naive CD8+ T cells?
2. Does Suv39h1 silence gene expression programs in activated CD8+ T cells?
3. Does Suv39h1 silencing affect the differentiation of new subsets of activated CD8+ T lymphocytes?

With the script below we have performed the following steps:

- annotating the counts table with **scannobyGt** function,
- removing the ribosomal and mitochondrial genes with **scannobyGt** function,
- selecting the most expressed 10K genes for the analysis with **topx** function,
- detecting the range of interesting number of cluster to be used for data partitioning (6-9) with **clusterNgriph**
- identifying 6 as the optimal number of clusters and evaluated the cell stability in each cluster with **simlrBootstrap**
- generating the video of the bootstrap analysis with **bootstrapsVideo**

```
system("wget http://130.192.119.59/public/setPace.txt.zip")
unzip("setPace.txt.zip")

#annotating genes and removing ribosomal and mitochondrial proteins
scannobyGtf(group="docker", file=paste(getwd(),"setPace.txt",sep="/"),
                    gtf.name="genome_mm10.gtf", biotype="protein_coding",
                    mt=FALSE, ribo.proteins=FALSE,umiXgene=3)

#selecting 10K top expressed genes
topx(data.folder=getwd(),file.name="annotated_setPace.txt",threshold=10000, logged=FALSE)

#defining the range of number of clusters to be investigated
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "annotated_setPace_10000.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)
#38 mins on SeqBox hardware

#running the SIMLR clustering in the range 6-9 detected by clusterNgriph
simlrBootstrap(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(), "annotated_setPace_10000.txt", sep="/"),
               nPerm=160, permAtTime=8, percent=10, range1=6, range2=9,
               separator="\t",logTen=0, seed=111)

# visualizing cells instability
bootstrapsVideo(group="docker", scratch.folder="/data/scratch",
                file=paste(getwd(), "annotated_setPace_10000.txt", sep="/"),
                nCluster=6, separator="\t", framePP=200, permutationNumber=80)
```

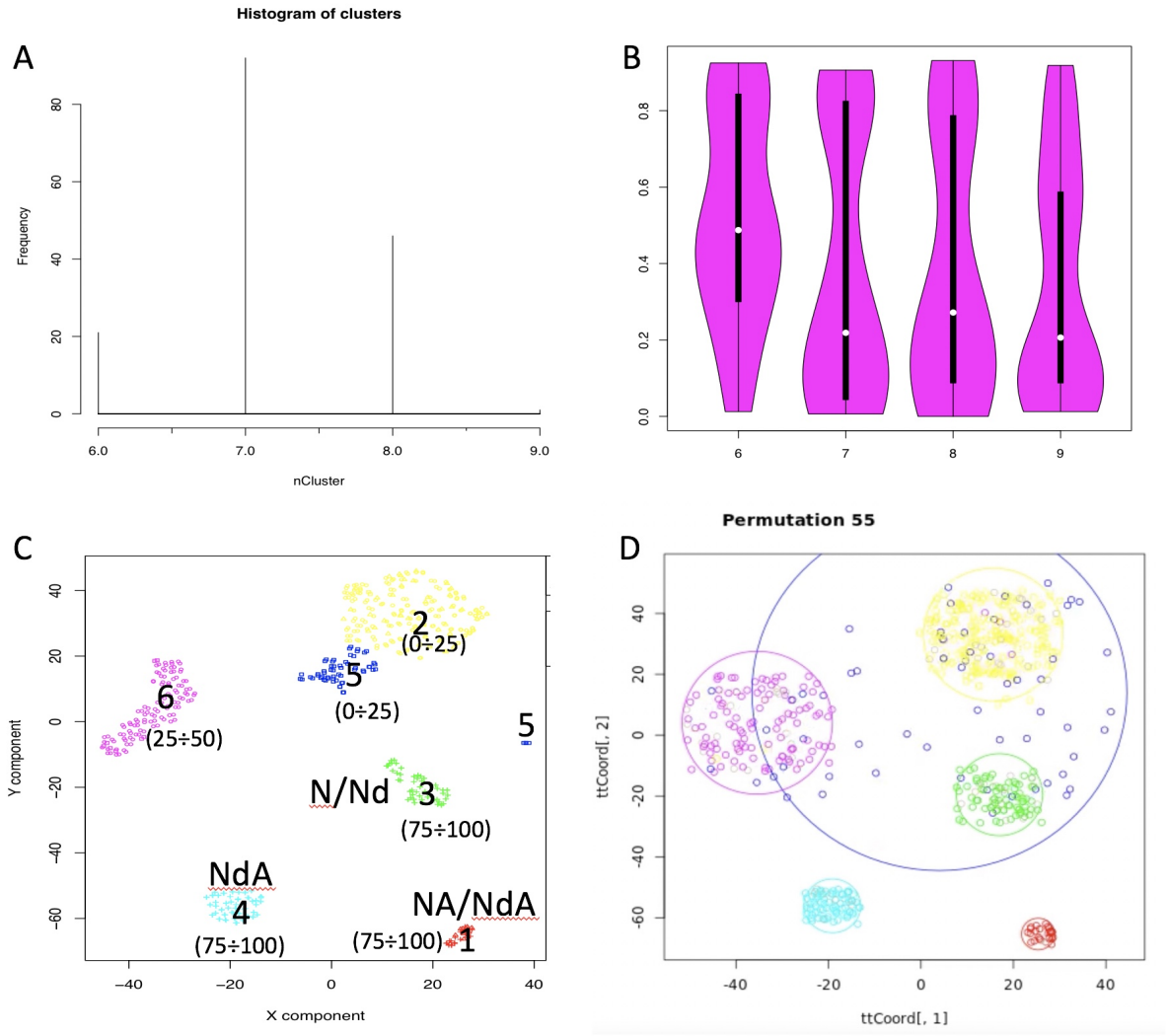


Figure 28: Analysis of a subset of cells from GSE106264 dataset. A) Clusters detected by clusterNgriph. B) Cell stability score detected by simlrBootstrap. C) Clusterization with 6 clusters with simlrBootstrap, in parentesis is given the overall stability score of each cluster. The components of each cluster, in terms of cells experiment groups, are also indicated for the most stable clusters. D) Permutation 55 extracted from the video generate with bootstrapsVideo.

In Figure 28 are summarized the results of the analysis executed on the Pace's dataset. Data perturbations, Figure 28A, allows data organization between 6 to 9 clusters, where 7 clusters is the most represented group. Cell stability score, from the SIMLR analysis executed on the above range of clusters, is shown in Figure 28B. Six clusters show a slighly higher stability with respect to the others. The overall stability of 6 clusters is however sub-optimal, since it is spread between 0 and 0.9 cell stability score. In Figure 28C it is shown the clusters structure generated with SIMLR on 6 clusters. Clusters 1, 3 and 4 show a quite good stability, Figure 28C. Cluster 3 is made of 44 **N** (88%) and 48 **Nd** (96%), suggesting that naive CD8+ T lymphocytes are not affected by the silencing of Suv39h1 gene. Cluster 1 contains 16 **NA** (6.4%) and 14 **NdA** (5.6%). Cluster 2 is made of 44.8% of **NA** and 39.6% of **NdA** cells. Clusters 1 and 2 group together,

interdependently from Suv39h1 gene silencing. Cluster 6 is made of 35% **NA** and 13.6% of **NdA**. In cluster 6 the amount of **NA** and **NdA** is unbalance, suggesting that the Suv39h1 silencing does not guarantee the efficient differentiation of the cell subpopulation in cluster 6. Cluster 4 only contains **NdA** (33%), indicating that at least a subpopulation of activated Suv39h1-silenced cells has a specific transcription profile that differentiate them from the wild type activated cells. Cluster 5 is made of 6 **N** cells, 2 **Nd** cells, 34 **NA** cells, 21 **NdA** cells. Despite the presence of a limited amount of naive cells, which might be explained as partially activated, cluster 5 is made mainly of activated cells, i.e. 13.6% NA and 8.4% NdA of total cells. The cluster structure (Figure 28D) and cell cluster stability scores (Figure 28C) might suggest that cluster 5 is made of a precursor subset.

NB % always refers to the total number of cells used in this example (50 N, 50 Nd, 250 NA, 250 NdA); *annotated_setPace_10000_clustering.output.txt*, contains all information required to reproduce clusters shown in Figure 28C. In Figure 28D it is shown permutation 55 of the output of **bootstrapsVideo**, which can be seen *here*. This permutation is representative of the majority of the observed permutations. From **bootstrapsVideo** analysis is clear that cluster 5 is strongly affecting the cell score stability of clusters 6 and 2 ad for less extent that of cluster 3.

This analysis can provide some preliminary answers to the questions raised above:

1. Does Suv39h1 silence gene expression programs in naive CD8+ T cells?
    a. No, naive cells cluster together independently from Suv39h1 silencing.
2. Does Suv39h1 silence gene expression programs in activated CD8+ T cells?
    a. Yes, cluster 4 is only made of an activated Suv39h1 KO sub-population. Furthermore, although quite unstable, cluster 6 sub-population is less represented in activated Suv39h1 silenced cells, suggesting a reduction in efficiency in the differentiation of this sub-population upon Suv39h1 silencing.
3. Does Suv39h1 silencing affect the differentiation of new subsets of activated CD8+ T lymphocytes?
    a. Yes, Cluster 4 represents a unique Suv39h1 subset of activated CD8+ T cells.

Section 8.2 Improving clustering results

On the basis of Figure 28D and the output of *bootstrapsVideo* clearly cells in cluster 5 represent a strong interference for an efficient clustering. Thus, we investigated if, removing cluster 5 cells (Figure 28C), cell stability score for cluster 6 and 2 could be improved.

```
system("wget http://130.192.119.59/public/annotated_setPace_10000.txt.zip")
unzip("annotated_setPace_10000.txt.zip")
system("wget http://130.192.119.59/public/annotated_setPace_10000_clustering.output.txt")

#removing cluster 5 cells from annotated_setPace_10000.txt
```

```
pace <- read.table("annotated_setPace_10000.txt", sep="\t", header=T, row.names=1)
clusters.info <- read.table("annotated_setPace_10000_clustering.output.txt", sep="\t", header=T, row.names=1)

discard <- rownames(clusters.info)[which(clusters.info$Belonging_Cluster==5)]

pace <- pace[,which(!names(pace)%in%discard)]
write.table(pace, "annotated_setPace_10000_noC5.txt", sep="\t", col.names=NA)

#defining the range of number of clusters to be investigated
clusterNgriph(group="docker",scratch.folder="/data/scratch/",file=paste(getwd(),
              "annotated_setPace_10000_noC5.txt", sep="/"), nPerm=160,
              permAtTime=8, percent=10, separator="\t",logTen=0, seed=111)


#running the SIMLR clustering in the range 5-9 detected by clusterNgriph
simlrBootstrap(group="docker",scratch.folder="/data/scratch/",
               file=paste(getwd(), "annotated_setPace_10000.txt", sep="/"),
               nPerm=160, permAtTime=8, percent=10, range1=5, range2=9,
               separator="\t",logTen=0, seed=111)
#execution time 276 mins in SeqBox
```

The results of the analysis done removing cells associated to cluster 5 are shown in Figure 29. The range of clusters suggested by **clusterNgriph** are between 5 and 9, Figure 29A. The best cells stability score is observable for 5 clusters, Figure 29B. **simlrBootstrap** results show a large improvment of the overall stablity of the clusters Figure 29C with respect to the previous analysis, Figure 28C. It is also notable that the samples organization with in clusters agrees with preliminary answers in Section 5.1.1 .

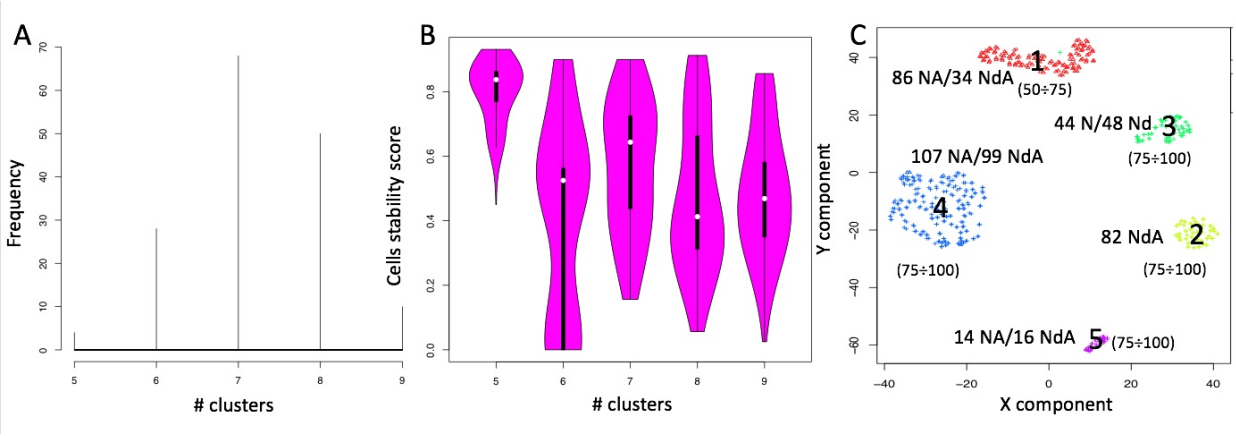


Figure 29: Analysis of Pace dataset. A) Clusters detected by clusterNgriph. B) Cell stability score detected by simlrBootstrap. C) Clusterization with 5 clusters with simlrBootstrap, in parentesis is given the overall stability score of each cluster. The components of each cluster, in terms of cells experiment groups, are also indicated for the most stable clusters.

Section 8.3 Detecting the genes playing the major role in clusters generation: EdgeR ANOVA-like analysis

We detected a total of 2742 differentially expressed genes using the naive cluster 3 as reference in an ANOVA-like analysis, see **Section 6.1**.

```
system("wget http://130.192.119.59/public/annotated_setPace_10000_noC5_clustering.output.txt")
system("wget http://130.192.119.59/public/annotated_setPace_10000_noC5.txt.zip")
unzip("annotated_setPace_10000_noC5.txt.zip")

anovaLike(group="docker", data.folder=getwd(), counts.table="annotated_setPace_10000_noC5.txt",
       file.type="txt", cluster.file="annotated_setPace_10000_noC5_clustering.output.txt", ref.cluster=3,
       logFC.threshold=1, FDR.threshold=0.05, logCPM.threshold=4, plot=TRUE)

#the output of interest is logFC_filtered_DE_annotated_setPace_10000_noC5_reordered.txt
```

The following analysis focus only on up-regulated genes, i.e. those gene specifically more expressed in a cluster with respect to the naive cluster used as reference in the ANOVA-like.

```
system("wget http://130.192.119.59/public/clusters.features.zip")
unzip("clusters.features.zip")
setwd("clusters.features")

clustersFeatures(group="docker", data.folder=getwd(),
        logFC.table="logFC_filtered_DE_annotated_setPace_10000_noC5_reordered.txt",
        counts.table="annotated_setPace_10000_noC5_reordered.txt", delta=0.5)
```

We analysed the results coming out of the **clustersFeatures** selection focusing on the two clusters that are different in cell ratio between wt and Suv39h1-defective T-cells: cluster 1, showing a 2.4:1 ratio between wt and Suv39h1-defective T-cells, and 2, only detected in Suv39h1-defective T-cells, Figure 29C. 31 genes were detected as specific for cluster 1. *EnrichR* analysis indicated the overepresentation of STAT3 in ENCODE and ChEA Consensus TFs from ChIP-X datasets and TNFRSF13B, CCL5, CCL4 and CXCR6 in Cytokine-cytokine receptor interaction dataset. On the basis of the above observations and on a PUBMED search (*Ccl5*, *STAT3*, *Ccl9* and *Cxcr6*), we suggest that cluster 1 is made of early activated precursors having some traits of memory cells. This cluster is also characterized by an unbalance ratio between activated wt T-cells and Suv39h1-defective T-cells.

The cluster 2 has some traits of memory cells, on the basis of EnrichR analysis of 100 genes. Enrichment analysis shows enrichment for STAT3 transcription specific paths and GO Biological process enrichment for **response to cytokine**. Furthermore, a PUBMED search showed the presence in this cluster of memory specific genes such as *TCF7*, *FOXO1*, *Bach2*, *Ccr7*, *Id3*, *Il7r*, *Myc*, *Pdk1*, *Socs3*, *Tnfsf8*,

The analysis of cluster 2 was extended using Ingenuity Pathways Analysis (IPA). This sub-population of cells, only detectable within Suv39h1-defective T-cells, is enriched for immuno-related functions: Development of hematopoietic progenitor cells, Adhesion of lymphocytes, Quantity of T-lymphocytes, systemic autoimmune syndrome. A subset of cluster 2 genes can be also aggregated in a network characterized by various transcription

factors (Figure 30). Interestingly, in this network are present Id3, Bach2 IL7R and Myc. Combining the information that Bach2 is associated to the generation (*Roychoudhuri et al. 2016*) of long-lived memory cells and Id3 to their maintainance (*Yang et al. 2011*) together with the knowledge that IL7R/Myc expression is found to be associated to a longer persistence of mouse memory T-cell (*Pace et al. 2018*) it coud be suggested that cells belonging to this cluster are precursors of long-lived memory cells.

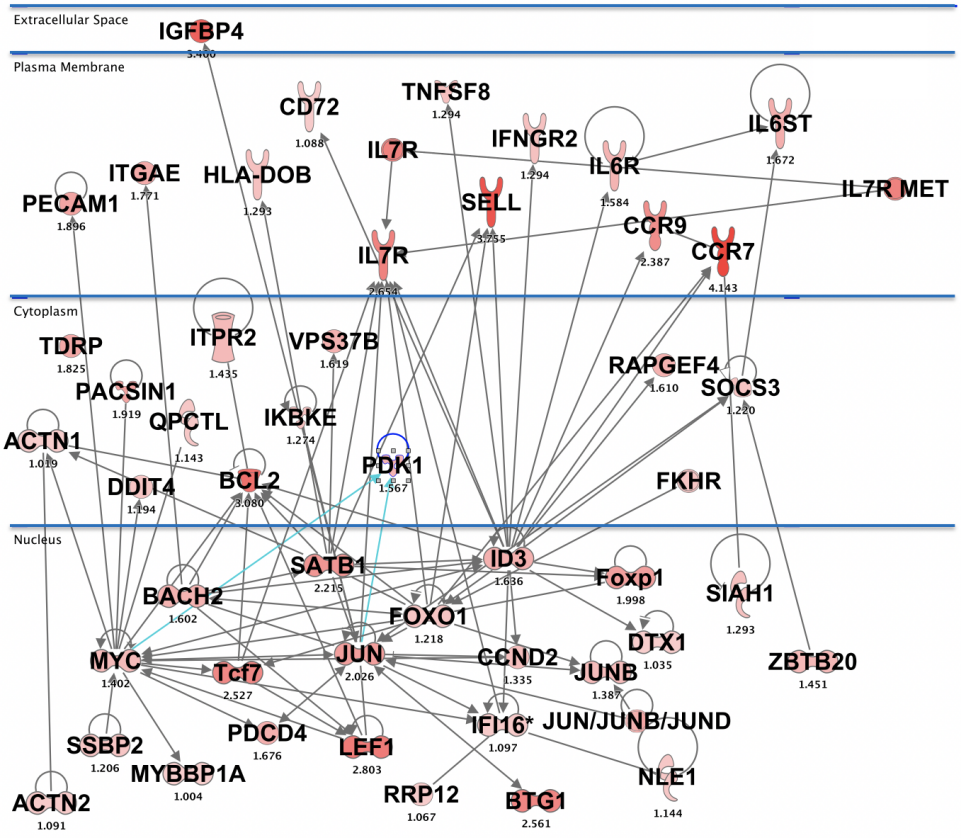


Figure 30: IPA C2 connected genes.

The paper *Pace* showed that Suv39h1-defective CD8+ T cells show sustained survival and increased long-term memory reprogramming capacity. Our reanalysis with rCASC extends the information provided from the single cell analysis in Pace paper, suggesting the presence of an enriched Suv39h1-defective memory subset, cluster 2 in Figure 29C. Noteworthy, this example suggests that even few hunders cells are sufficient to discriminate between sub-populations.

Section 9 rCASC versus other workflows

In the last few years *a lot of tools for single-cell data manipulation have been published*. However, the number of single cell analysis workflows are relatively few. Our non-exaustive search identified the following:

- *Lun and coworkers* recently described a single-cell workflow in Bioconductor, which is embeded in *simpleSingleCell* Bioconductor workflow package.
- *Zhu and coworkers* published, *Granatum*, a web-based scRNA-Seq analysis suite.
- *Diaz and coworkers* published a *SCell* a graphical workflow for single-cell analysis.
- *Butler and coworkers* published the R toolkit Seurat.

For the comparison we evaluated the following parameters:

- supported single-cell platforms
- available tools provided by the workflow
- type of reproducibility granted by the workflow
- usage flexibility

		rCASC (stand alone)	simpleSingleCell (stand alone)	Granatum (web)	Scell (stand alone)	Seurat (stand alone)
Platforms		10Xgenomics, inDrop, counts table	counts table	counts table	counts table	counts table
	Fastq conversion in counts table	Y	-	-	-	-
	Quality Control / Outlier Filtering	Y	-	Y	Y	Y
	Annotation	ENSEMBL ID -> Gene Symbol	-	-	-	-
	Genes filter	Y	-	Y	Y	Y
	Data normalization	Y	Y	Y	Y	Y
	Cell cycle bias removal	Y	-	-	Y	Y
	Data dimensionality reduction	Y	Y	Y	Y	Y
Tools	*Supported clustering methods*	tSne, SIMLR, Seurat PCA	Walktrap	Non-negative matrix factorization, K-mean (Euclidean), K-mean (tSne)	PCA, k-means, Gaussian mixture, Minkowski weighted k-means, DBSCAN	PCA, tSne, ica, dmap
	Cluster quality score	Silhuette, Cell Stability Score	-	-	-	-
	Features selection and visualization	Y	Y	Y	-	-
	Supported methods	ANOVA-like (edgeR), SIMLR Seurat genes prioritization	filtering on expression	NODES, SCDE, EdgeR, Limma	-	wilcox, bimod, roc, t-test, tobit, negbinom, MAST, DESeq2
	Biological inference	-	-	Y	-	-
Reproducibility	*Functional reproducibility*	Y	Y	-	-	Y
	Computational reproducibility	Y	-	Y	Y	-
Flexibility	*line command execution*	Y	Y	-	-	Y
	graphical interface	Y	-	Y	Y	-

Figure 31: Workflows comparisons.

In Figure 31 are summarized the similarities and differences between rCASC and the other four workflows. rCASC is the only one providing support at fastq level, as all the others require a processed counts table. rCASC, SCell and Seurat data preprocessing offer more options than the others. The five workflows implement very different data reduction and clustering methods. Recently, *Freytag and coworkers* published a comparison of clustering methods, in which Seurat and SIMLR were included. The two methods performed in similar

way on the golden standard dataset used by Freytag. Freytag observation is also confermed in our test in **Section 6.1**.

Only rCASC performs clustering in presence of data perturbation and it is also the only one that provides measurement for cluster quality using silhuette score, measuring the consistency within clusters of data, and Cells Stability Score, measuring the persistence of each cell in a cluster upon data perturbation. Gene feature selection approaches are quite different between workflows and biological inference is only provided by Granatum. rCASC is the only one embedding three different feature selection approaches: ANOVA-like (EdgeR package), SIMLR and Seurat genes prioritization procedures. Considering reproducibility, only rCASC provides both computational and functional reproducibility. Both Granatum and Scell perform data manipulation based on graphical tools, e.g. cell outlyers removal in Granatum and genes filtering in SCell preprocessing, which are not trackable, thus not providing functional reproducibility. On the other side simpleSingleCell and Seurat package cannot guarantee computational reproducibility unless they are embedded in a virtual machine or another type of virtual container, e.g. docker.

In conclusion, from the above comparisons, rCASC results to be a workflow with valuable new features that could help researchers in defining cells sub-populations and detecting sub-population specific markers, under the umbrella of data reproducibility.