Reference manual: ScarTrace clone detection

version 1.0

Contents

1 Requirements **1**

2 Data files **2**

3 Scar extraction **2**
3.1 Map scar library with bwa 2
3.2 Raw scar extraction 2

4 Scar filtering **3**
4.1 Quality checks 3
4.2 Read filtering and normalization 4
4.3 Cleaning scars 4

5 Clone extraction **5**
5.1 Merge several scar tables 5
5.2 Hierarchical clustering and further scar filtering 6
5.3 Clustering based on the presence of scars 7

6 Clone Mergin and filtering **7**
6.1 Merging similar clones 7
6.2 Filtering small clones 8

7 Figures **8**

1 Requirements

All python scripts are written for `python 2.7`. The following packages are needed:

- `sys, os, numpy, pandas, Biophyton (), argparse, collections, gzip, lmfit, itertools, matplotlib.pyplot, multiprocessing, pickle, re, scipy.optimize, scipy.spatial, sklearn.cluster, Colors, Scar`

Packages `Colors` and `Scar` are self made and provided with the code. No installation is required.

Additionally, `bwa-0.7.10` is required. `Gnuplot` is recommended.

2 Data files

For each scar library (that usually contains scars for 384 cells) we need to `fastq` files, one for each read:

- `libname_R1.fastq`
- `libname_R2.fastq`

Sample data is provided in Supplementary Data 3. All scar data has been produced by sequencing pair-end at 75 bp read length on the Illumina NextSeq platform. Read 1 contains a 3-bp long UMI (that is not used) and the cell-specific barcode, and Read 2 contains scar information.

3 Scar extraction

3.1 Map scar library with bwa

```
> mapFastq.sh libname path_to_bwa
```

This script takes three input parameters:

- `libname`: name of the library to process. In case the script does not run on the folder with the data, the full path needs to be given.
- `path_to_bwa`: path to `bwa-0.7.10` software

The script produces a new `fastq` file (`libname_cbc.fastq`) where only scar reads (from Read 2) with proper cell-specific barcodes (extracted from the corresponding Read 1) are included. Next, `libname_cbc.fastq` is mapped to the reference GFP sequence using `bwa-0.7.10 mem`. Finally, it zips the resulting `libname.sam file`.

3.2 Raw scar extraction

```
> python bin/readSAMpileup.py --sam libname.sam.gz --out out1
```

The script selects reads in the `sam` file which have been mapped to the GFP with proper strand orientation and which contain the primer used in the nested PCR used for scar amplification (5'-GGCCCCGTGCTGCTGCCCGAC-3', end of read) with 3 or less mismatches. Next, it counts how many times each read is seen in each cell. The resulting table is stored in a tabular separated file (`out1.tsv`) and a pickle file (`out1.pickle`) containing the table.

```
> python bin/realignScars.py --picklein out1.pickle --out out2 --th 8
```

The script remaps unique reads from `out1` using the `biopython` function `pairwise2.align.globalms` with match, mismatch, open and extend parameters equal to 1, 0.25, -1 and -0.1, respectively. This corrects sequencing errors with reasonable accuracy and pools together reads that belong to the same scar for the same cell despite sequencing errors. From now on, scars are defined using CIGAR strings, and raw sequences are not used anymore. Output files are:

- `out2.txt`: text file containing the list of all reads with the corresponding alignment to the reference GFP sequence and CIGAR string.
- `out2.pickle`: pickle file containing the list of all reads with the corresponding alignment to the reference GFP sequence and CIGAR string.
- `out2.tsv`: tsv-file with the number of reads per cell for each scar, denoted using CIGAR strings.

4 Scar filtering

4.1 Quality checks

```
> python bin/findThresholds-QCplots.py out2.tsv
```

The script takes as an input the file `out2.tsv` and produces standard quality-check plots. Additionally, it provides possible thresholds to filter out cells where the ScarTrace protocol did not work, and to filter out noisy scars that arise due to sequencing errors and mapping artifacts. Output files are:

- `sumXcell.txt`: text file with the total number of reads detected per cell. Cells with no reads are not reported.
- `sumXcell.pdf`: pdf file with the barplot of the total number of reads detected per cell (example in Fig. 1a).
- `sumXcell.hst`: text file with the normalized histogram of the total number of reads detected per cell used to generate the `sumXcell-histo.pdf` plot (see below).
- `sumXcell.fit`: text file that summarizes the results from fitting the normalized histogram of the total number of reads per cell to a double Gaussian function, as shown in `sumXcell-histo.pdf` (see below).
- `sumXcell-histo.pdf`: pdf file with a plot showing the histogram of the total number of detected reads per cell and the corresponding fit to a double Gaussian function. The x-axis is shown in $\log_{10}$ scale. The plot contains a label that gives the value r_m for the minimum of the double Gaussian function, which can be used as a threshold to filter out cells with less reads than 10^{r_m} (Fig. 1b).
- `sumXscar.txt`: text file with number of reads for each scar, annotated with the CIGAR string.

- sumXscar.hst: text file with the histogram of the number of reads per scar, used to generate the `sumXscar-histo.pdf` plot (see below).

- `sumXscar.fit`: text file with results from fitting the histogram of total number of reads per scar to a three domain piecewise-defined linear function:

$$f(x) = \begin{cases} a_1 + b_1 x, & x < f_1 \\ a_1 + (b_1 - b_2) f_1 + b_2 x, & f_1 \leq x < f_2 \\ a_1 + (b_1 - b_2) f_1 + (b_2 - b_3) f_2 + b_3 x, & x \geq f_2 \end{cases} \quad (1)$$

  The $x \geq f_2$ domain corresponds to highly-detected scars; the $f_1 \leq x < f_2$ domain contains scars whose sequence is a few nucleotides away from scars that are present in the previous domain, hence these scars are due to sequencing errors of sequences in the previous domain; the $x < f_1$ domain corresponds to hardly-detected scars that probably are generated by sequencing noise. The value 10^{f_2} provides a reasonable threshold to filter out noisy scars that are seen with less reads. Results from this fit are shown in `sumXscar.pdf` and `sumXscar-histo.pdf` (see below).

- `sumXscar.pdf`: plot showing the sorted number of reads for each scar (Fig. 1c) and the value of 10^{f_2} obtained from the aforementioned fit.

- `sumXscar-histo.pdf`: plot showing the normalized histogram of number of reads per scar and the fit to Eq. (1). The shape of this histogram is common in all scar libraries in all scar libraries (Fig. 1d). The plot also contains a label indicating the value of f_2.

4.2 Read filtering and normalization

```
> python bin/filter-normalize.py out2.tsv cell_threshold scar_threshold
out3
```

This script takes as an input the file out2.tsv that contains number of scars per cell, the read threshold for cells (`cell_threshold` $\sim 10^{r_m}$ is recommended), the read threshold for scars (`scar_threshold` $\sim 10^{f_2}$ recommended), and a root name for the output file. Output files are:

- `out3_filter.txt`: text file with the filtered table of raw reads for each scar per cell.

- `out3_norm-filter.txt`: text file with the filtered table with the percentage of each scar per cell, obtained by dividing the number of reads per scar by the total number of reads per cell, and multiplying the result by 100.

4.3 Cleaning scars

```
> python bin/scarPurityHistogram.py out3_norm-filter.txt out2.txt outdir
```

The script takes as input files `out3_norm-filter.txt` and `out2.txt`, and extracts the histogram of nucleotide content per position for each scar. Therefore, when in a given position there are mainly two nucleotides observed with a 50%-50% content, this indicates that the corresponding CIGAR string codes two different sequences (and hence, two different scars). All histograms (text files and pdf plots, Fig. 2) are stored in the `outdir` directory (which the script creates in case it does not exist).

```
> python bin/cleanScarErrors.py out3_norm-filter.txt outdir out4
```

The script performs a pairwise comparison between the sequence-content of the different CIGAR strings. When the number of mismatches between two CIGAR strings detected in the same cell is lower than a given threshold (set to 5 by default), the script assumes that these correspond to the same scar and merges them. Output files are:

- `out4.log`: log file that details which scars are considered to be sequencing errors of others and which have been pooled for each cell.
- `out4.txt`: text file with the table of renormalized scar percentage per cell after filtering out possible sequencing errors.

5 Clone extraction

5.1 Merge several scar tables

The following scripts can run on merged tables containing scar percentage per cell. In order to merge a given number `n` of tables, the following script can we used:

```
> python bin/mergeDF.py n file_1 label_1 file_2 label_2 ...  file_n label_n
method merged_out
```

The script needs as an input the total number of files to be merged, followed by the file names (`file_1, file_2, ..., file_n`), and the corresponding labels to be append at the cell-ID labels of each file. Additional input parameters are:

- `method`: merging method, which can be either `inner` (in case only scars present in all input files are used) or `outer` (in case all scars are used, even if some are not present in all input files).
- `merged_out`: root of the name for the output files.

Output files are:

- `merged_out.txt`: merged table with the percentage of each scar per cell will contain all columns from file_1, file_2, ..., file_n with the appended corresponding label, and merged scars for all files as rows.
- `merged_out.log`: summary of files merged and corresponding labels.

5.2 Hierarchical clustering and further scar filtering

```
> python bin/hierarchicalClustering.py merged_out.tsv n m_out method_label
pdfplot_label
```

The script clusters cells based on scar percentage using hierarchical or agglomerative clustering. Input parameters are:

- `merged_out.tsv`: file with table of scar percentage per cell.
- `n`: Number of clusters. It is not important to get the exact number of clusters.
- `m_out`: root name for the output files.
- `method_label`: label to choose the clustering algorithm approach: "`hcl`" for hierarchical clustering and "`acl`" for agglomerative clustering.
- `pdfplot_label`: label to activate the generation of a plot with the stacked histogram of scar percentage per cell when set to "`y`". The generation of this plot slows down the code.

Output files are:

- `m_out_df.txt`: text file with the table of scar percentage per cell, which is now transposed and contains a new column with the identity of the assigned cluster for each cell.
- `m_out_clust.txt`: text file with the list of cluster assigned to each cell.
- `m_out_centroid.txt`: text file with table of mean scar expression for each cluster computed among all cells assigned to corresponding cluster.
- `m_out_rname.gpl`: script to generate scar barplot in `gnuplot`.
- `m_out_barplot.pdf`: barplot displaying scar percentage per cell, with cells sorted by clone identity (Fig. 3, top). This plot is only generated when pdfplot_label is set to "`y`". For the sake of speed, it is better to produce this plot with `gnuplot` using the script `m_out.gpl`.

```
> python bin/cleanCellScarfraction.py m_out_df.txt threshold output5 plotlabel
```

From the barplot it can be seen that all cells have residual scars (with a tiny percentage) that mainly consist of scars leaking from other cells. The script will allow removing these scars. The script takes as input parameters:

- `m_out_df.txt`: text file with input scar table, obtained by running the previous script.
- `threshold`: threshold to filter out scars in a cell that are present with a lower percentage. A common threshold is 3.5.

- `output5`: name for the output file.
- `plotlabel`: label to activate the generation of a plot with the new stacked histogram of scar percentage per cell when set to "`y`". The generation of this plot slows down the code.

As an output, the script produces a text file named `output5.txt` that contains the new filtered table. In case `plotlabel` equals "`y`", the script also produces a barplot `output5_barplot.pdf` displaying scar percentage per cell, with cells sorted by clone identity (Fig. 3, middle).

5.3 Clustering based on the presence of scars

```
> python bin/HDclustering.py output5 output6 pdfplot_label
```

The script clusters together cells sharing the identical scar profie, without taking into account which percentages. As an input, the script needs a text file with the table of the scar percentage per cell (`output5.txt` or `m_out_df.txt`); a root name for output files (`output6`); and a `pdfplot_label` label (see below). Output files are:

- `output6_HD.txt`: text file with the scar percentage table which contains a column with the newly assigned cluster identity for each cell.
- `output6_HDcentroids.txt`: text file with table of mean scar expression over all cells assigned to each cluster.
- `output6.gpl`: script to generate scar barplot in `gnuplot`.
- `output6_HDbarplot.pdf`: barplot displaying scar percentage per cell, with cells sorted by clone identity (Fig. 3, bottom). This plot is only generated when `pdfplot_label` is set to "`y`".

6 Clone Mergin and filtering

6.1 Merging similar clones

```
> python bin/automatPool_HDClustering.py output6_HD.txt output7 pdfplot_label
```

The script automatically merges clones from file `output6_HD.txt` (or any other scar table) that have a very similar scar content, which might be explained by either the drop out of some scars during amplification of clonal lineage relationship (one being an ancestor of the other). As an output, if gives a new scar table `output7.txt` with the newly determined clusters. Additionally, if `pdfplot_label` is set to `y`, it produces a `output7_barplot.pdf` file with the barplot of scars with cells sorted according to the assigned clone (Fig. 4, top).

6.2 Filtering small clones

```
> python bin/rmSamllCl_HDpool.py output7.txt output8 pdfplot_label
```

The script removes clones in `output7.txt` that contains less than 2 cells and stores the new filtered table in `output8.txt`. Additionally, if `pdfplot_label` is set to `y`, it produces a `output8_barplot.pdf` file with the barplot of scars with cells sorted according to the assigned clone (Fig. 4, bottom).

7 Figures

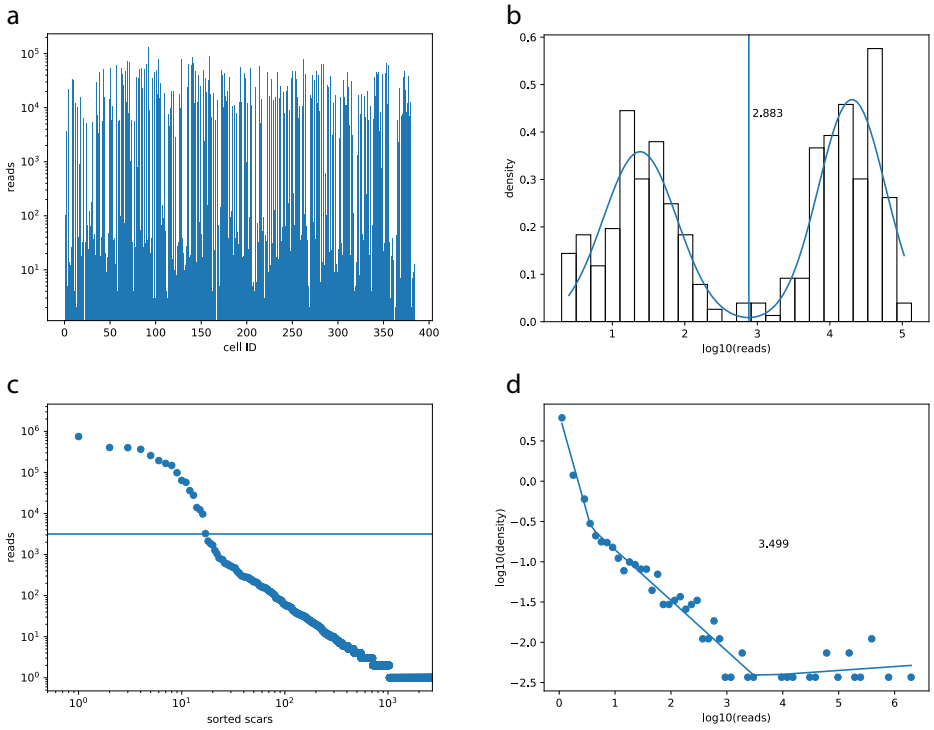

Figure 1: **Quality plots. a.** Number of reads per cell in the scar library provided as sample data in Supplementary Data 3. **b.** Normalized histogram of the number of reads per cell in $\log_{10}$ scale (black) and fit to a double Gaussian distribution (blue). The vertical line and the label (2.883) indicate the position of the minimum of the double Gaussian distribution function. **c.** Number of reads per scar in the sample data provided in Supplementary Data 3. The horizontal line indicates the value of 10^{f_2} obtained by fitting Eq. (1) to the histogram of reads per scar (panel **d**). **d.** Normalized histogram of the number of reads detected for each scar (blue circles) and fit to Eq. (1) (blue line). The label indicates the value obtained for f_2.

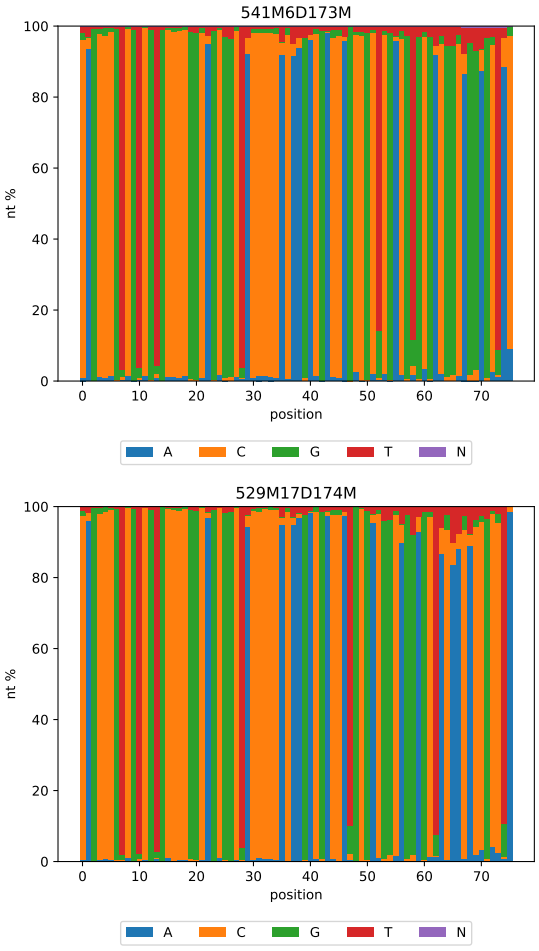

Figure 2: **Scar sequence content.** Percentage of nucleotide content in each position of a read, for two different scars labeled with CIGAR strings 541M6D173M (top) and 529M17D174M (bottom).

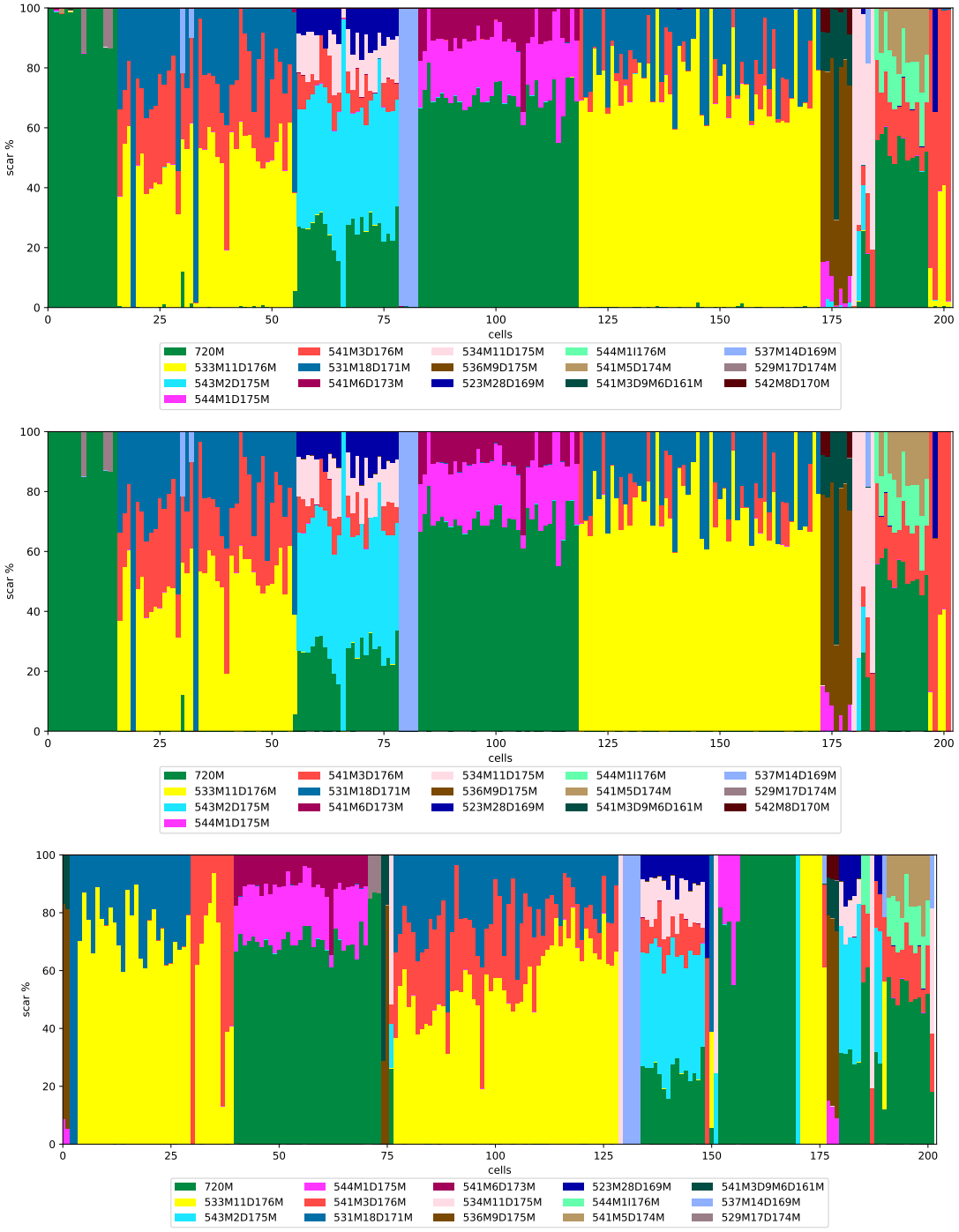

Figure 3: **Clustering cells based on scars**. **top.** Scar content per cell with cells sorted according to cluster identity obtained by hierarchical clustering them with a cluster number equal to 10 on the raw scar table produced after running `hierarchicalClustering.py` on the sample data provided in the Supplementary Data 3. **middle.** Scar content per cell with cells sorted according to cluster identity obtained by hierarchical clustering them (top panel) after removing scars in each cell with a percentage below 3.5% and renormalizing scar conent. **bottom.** Scar content per cell with cells sorted according to cluster identity obtained with `HDclustering.py`.

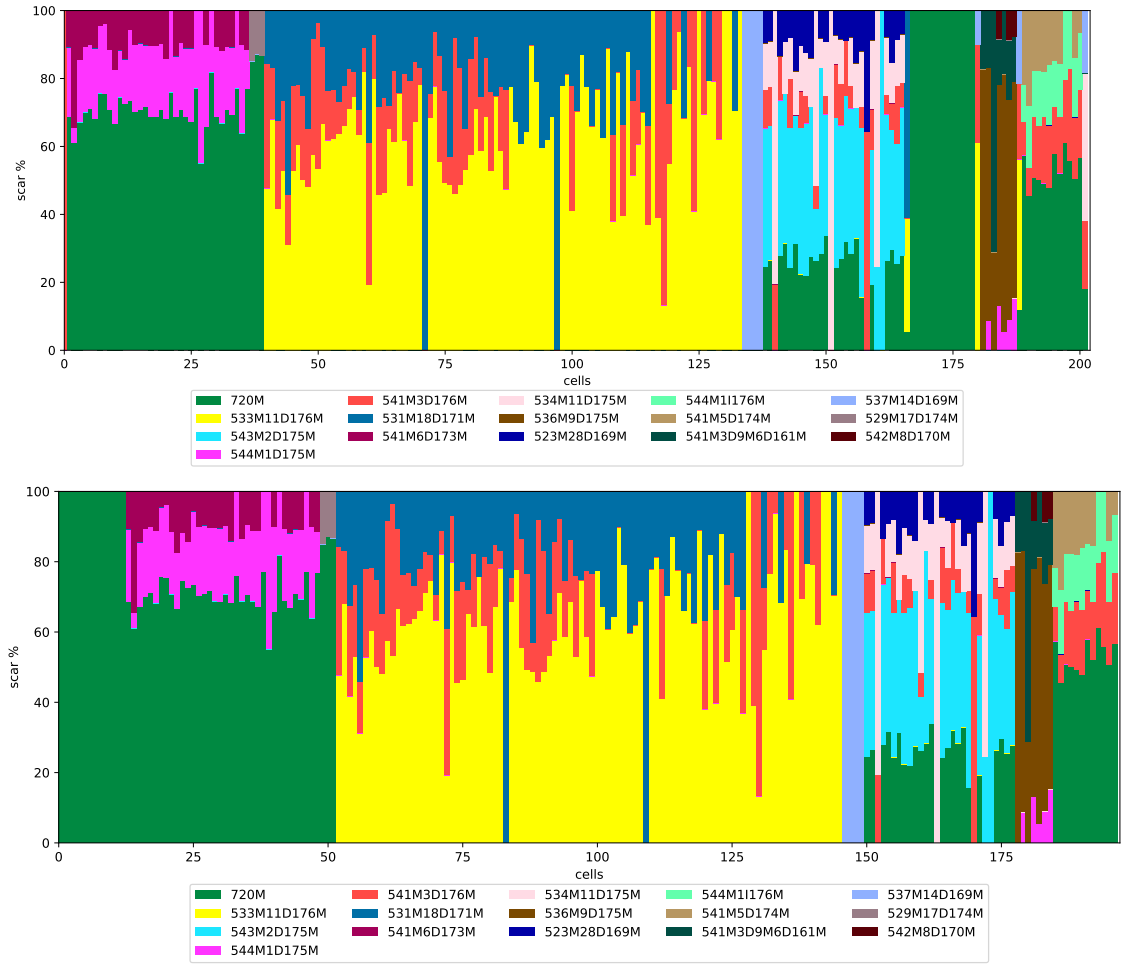

Figure 4: **Merge and filter clones**. Scar content per cell with cells sorted according to cluster identity obtained by pooling together clones with a highly similar scar pattern, using `rmSamllCl_HDpool.py` (**top**), and subsequently removing clones with less than 2 cells (**bottom**).