CATALYST: Tools for (pre)processing & analysis of cytometry data

Helena L Crowell [*1] ***and Mark D Robinson*** [2]

[1]Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland
[2]Institute of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

[*]crowellh@student.ethz.ch

12 February 2017

Abstract

By addressing the limit of measurable fluorescent parameters due to instrumentation and spectral overlap, mass cytometry (CyTOF) combines heavy metal spectrometry to allow examination of up to 100 parameters at the single cell level. While spectral overlap is significantly less pronounced in CyTOF than flow cytometry, spillover due to detection sensitivity, isotopic impurities, and oxide formation can impede data interpretability. We designed *CATALYST* (Cytometry dATa anALYSis Tools) to provide tools for (pre)processing and analysis of cytometry data, including compensation and in particular, an improved implementation of the single-cell deconvolution algorithm (Zunder et al. 2015, Nature Protocols 10, 316-333).

Package

CATALYST 0.1.3

Contents

1 Introduction

2 Data examples

To demonstrate the debarcoding and compensation work-flow with *CATALYST*, we provide two `flowFrame`s, `sample_ff` and `ss_exp`, obtained from 36ab-panel triple- and single-staining experiments, respectively. The former follows a 6-choose-3 barcoding scheme where mass channels 102, 104, 105, 106, 108, and 110 were used for labeling such that each of the 20 individual barcodes are positive for exactly 3 out of the 6 possible barcode channels. Accompanying this, the barcoding scheme provided in `sample_key` contains a binary code of length 6 for each sample, e.g. 111000, as its unique identifier. For the single-staining experiment, beads were stained with antibodies captured by mass channels 139, 141 through 156, and 158 through 176, respectively, and pooled together. Note that, to decrease running time, we sampled 10'000 of all recorded events in both data sets at the cost of not necessarily arriving at biologically meaningful results.

```
library(CATALYST)
data(sample_ff, sample_key, ss_exp)
```

3 Single-cell deconvolution

3.1 The `dbFrame` class

Data returned and used throughout the debarcoding process are stored in a debarcoding frame.

- Event information, stored in a matrix, is passed from the input `flowFrame` specified in `assignedPrelim` to the `exprs` slot, and is accessible via `exprs` as before.
- The `bc_key` slot is a binary matrix with numeric masses as column names and sample names as row names. If supplied with a numeric vector of masses, `assignPrelim` will generate a concurrent representation.
- `bc_ids` is a numeric vector of the ID assignments that have been made. If a given event's population separation falls below its separation cutoff, or above the population's Mahalanobis distance cutoff, it will be give ID 0 for *"unassigned"*. Assignments can be manipulated through standard replacement via `bc_ids<-`.
- The `deltas` slot contains for each event the separations between positive and nergative populations, that is, between the lowest positive and highest negative intesity.
- `normed_bcs` are the barcode intensities normalized by population. Here, each event is scaled to the 95% quantile of the population it's been assigned to. `sep_cutoffs` are applied to these normalized intensities.
- Slots `sep_cutoffs` and `mhl_cutoff` contain the devoncolution parameters.

- `counts` and `yields` are matrices of dimension number of samples x 101. Each row in the `counts` matrix contains the number of events within a sample for which positive and negative populations are separated by a distance between in [0,0.01), ..., [0.99,1], respectively. The percentage of events within a sample that will be obtained after applying a separation cutoff of 0, 0.01, ..., 1, respectively, is given in `yields`.

An overview of the object's dimensionality, current event assignments, deconvolution parameters, and yields achieved upon debarcoding is given by `show`.

```
## dbFrame objectect with
##  10000 events, 64 observables and 20 barcodes:
##
## Current assignments:
##       940 event(s) unassigned
## ID    B3   B4   B2   B5   B1   D3   D1   D4   D2   C4   D5   C3
## Count 1351 1257 1242 1232 1093  394  333  328  313  285  268  247
##
## ID    C1   C2   C5   A5   A2   A1   A4   A3
## Count  244  238  176   27   11    8    7    6
##
## Separation cutoffs:
## ID    B3   B4   B2   B5   B1   D3   D1   D4   D2   C4   D5   C3
## Yield 0.08 0.06 0.08 0.07 0.08 0.22 0.11 0.08 0.27 0.18 0.07 0.11
##
## ID    C1   C2   C5   A5   A2   A1   A4   A3
## Yield 0.09 0.17 0.08 0.16 0.11 0.10 0.09 0.10
##
## Yields upon debarcoding:
##       70.2% overall yield
## ID    B3  B4     B2     B5     B1     D3     D1     D4     D2
## Yield 98% 98.49% 97.83% 98.54% 97.62% 88.83% 92.49% 94.21% 82.43%
##
## ID    C4     D5    C3     C1     C2     C5     A5 A2     A1 A4 A3
## Yield 88.07% 95.9% 86.23% 89.34% 86.97% 90.91% 0% 18.18% 0% 0% 0%
```

3.2 Deconvolution work-flow with *CATALYST*

CATALYST provides three functions for debarcoding and two visualizations that guide selection of thresholds and give a sense of barcode assignment quality.

In summary, events are assigned to a sample when i) their positive and negative barcode populations are separated by a distance larger than a threshold value and ii) the combination of their positive barcode channels appears in the barcoding scheme.

3.2.1 `assignPrelim`: Assignment of preliminary IDs

The debarcoding step commences by assigning each event a preliminary barcode ID. `assignPrelim` thereby takes either a binary barcoding scheme or a vector of numeric masses as input, and accordingly assigns each event the appropirate row index or mass as ID. Depending on the `bc_key` supplied, there are two possible ways of proceeding:

1. **Doublet-filtering**:
   Given a binary barcoding scheme with a coherent number k of positive channels for all IDs, the k highest channels are considered positive and $n-k$ channels negative. Separation of positive and negative events equates to the difference between the kth highest and $(n-k)$th lowest intensity value. If a numeric vector of masses is supplied, the barcoding scheme will be an idensitiy matrix; the most intense channel is considered positive and its respective mass assigned as ID.

2. **Non-constant number of 1's**:
   Given an inconsistent number of 1's in the binary codes, the highest separation between consecutive barcodes is looked at. In both, the doublet-filtering and the latter case, each event is assigned a binary code that, if matched with a code in the barcoding scheme supplied, dictates which row index will be assigned as ID. Cells whose positive barcodes are still very low or whose binary pattern of positive and negative barcodes doesn't occur in the barcoding scheme will be given ID 0 for *"unassigned"*.

FCS files are read into R with `read.FCS` of the *flowCore* package, and are represented as an object of class `flowFrame`. Provided with a `flowFrame` and a compatible barcoding scheme (barcode channel masses have to occur in the measurement data), `assignPrelim` will return a `dbFrame` containing `exprs` passed from the input `flowFrame`, a numeric vector of event assignments in slot `bc_ids`, separations between barcode populations on the normalized scale in slot `deltas`, and normalized barcode intensities in slot `normed_bcs`. Measurement intensities are normalized by population such that each is scaled to the 95% quantile for asinh transformed measurement intensities of events assigned to the respective barcode.

```
sample_key
##    102 104 105 106 108 110
## A1   1   1   1   0   0   0
## A2   1   1   0   1   0   0
## A3   1   1   0   0   1   0
## A4   1   1   0   0   0   1
## A5   1   0   1   1   0   0
## B1   1   0   1   0   1   0
## B2   1   0   1   0   0   1
## B3   1   0   0   1   1   0
## B4   1   0   0   1   0   1
## B5   1   0   0   0   1   1
## C1   0   1   1   1   0   0
## C2   0   1   1   0   1   0
```

```
## C3   0   1   1   0   0   1
## C4   0   1   0   1   1   0
## C5   0   1   0   1   0   1
## D1   0   1   0   0   1   1
## D2   0   0   1   1   1   0
## D3   0   0   1   1   0   1
## D4   0   0   1   0   1   1
## D5   0   0   0   1   1   1
re <- assignPrelim(x = sample_ff, y = sample_key, verbose = FALSE)
re
## dbFrame objectect with
##  10000 events, 64 observables and 20 barcodes:
##
## Current assignments:
##       940 event(s) unassigned
## ID    B3   B4   B2   B5   B1   D3   D1   D4   D2   C4   D5   C3
## Count 1351 1257 1242 1232 1093  394  333  328  313  285  268  247
##
## ID    C1   C2   C5   A5   A2   A1   A4   A3
## Count  244  238  176   27   11    8    7    6
```

3.2.2 `estCutoffs`: Estimation of distance cutoffs

Here, the choice of thresholds for the distance between negative and positive barcode populations is *i) automated* and *ii) independent for each barcode*. As opposed to a single and global cutoff parameter, `estCutoffs` will estimate a cutoff value that is specific for each sample to deal with barcode population cell yields that decline in an asynchronous fashion. The function will update the `sep_cutoffs`, `counts` and `yields` slots of the input `dbFrame`.

For the estimation of cutoff parameters we concider yields upon debarcoding as a function of the applied cutoffs. Commonly, this function will characterized by an initial weak decline, where doublets are excluded, and subsequent rapid decline in yields to zero. Inbetween, low numbers of counts with intermediate barcode separation give rise to a plateau. The separation cutoff value should be chosen such that it appropriately balances confidence in barcode assignment and cell yield. We thus fit the yields function, its first and second derivative, and compute the first turning point, marking the on-set of the plateu regime, as an adequte cutoff estimate.

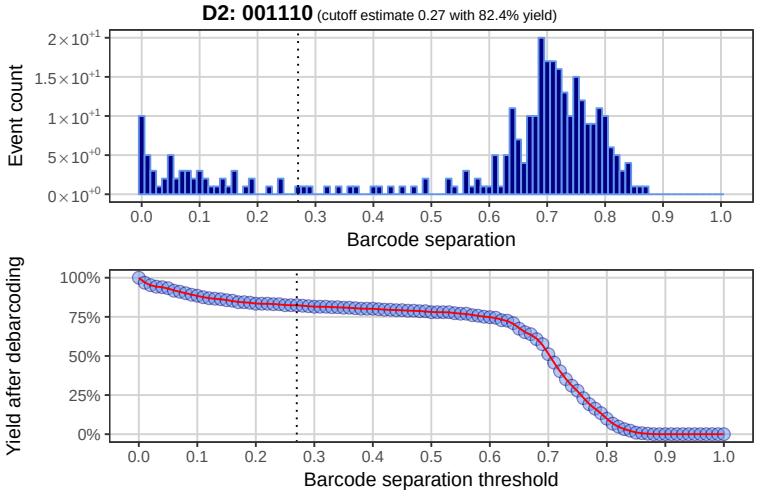
```
# estimate separation cutoffs, and
# get counts and yields as a function of barcode separations
re <- estCutoffs(x = re, verbose = FALSE)
re
## dbFrame objectect with
##  10000 events, 64 observables and 20 barcodes:
##
```

```
## Current assignments:
##       940 event(s) unassigned
## ID    B3   B4   B2   B5   B1   D3   D1   D4   D2   C4   D5   C3
## Count 1351 1257 1242 1232 1093  394  333  328  313  285  268  247
##
## ID    C1   C2   C5   A5   A2   A1   A4   A3
## Count  244  238  176   27   11    8    7    6
##
## Separation cutoffs:
## ID    B3   B4   B2   B5   B1   D3   D1   D4   D2   C4   D5   C3
## Yield 0.08 0.06 0.08 0.07 0.08 0.22 0.11 0.08 0.27 0.18 0.07 0.11
##
## ID    C1   C2   C5   A5   A2   A1   A4   A3
## Yield 0.09 0.17 0.08 0.16 0.11 0.10 0.09 0.10
##
## Yields upon debarcoding:
##       70.2% overall yield
## ID    B3  B4     B2     B5     B1     D3     D1     D4     D2
## Yield 98% 98.49% 97.83% 98.54% 97.62% 88.83% 92.49% 94.21% 82.43%
##
## ID    C4     D5    C3     C1     C2     C5     A5 A2     A1 A4 A3
## Yield 88.07% 95.9% 86.23% 89.34% 86.97% 90.91% 0% 18.18% 0% 0% 0%
```

3.2.3 `plotYields`: Distribution of population separations and yields upon debarcoding

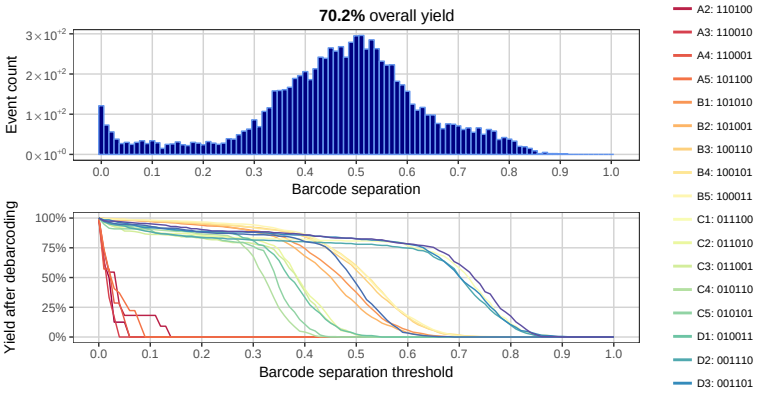
For each barcode, `plotYields` will generate a histogram of events separated by a given distance, as well as yields upon debarcoding as a function of separation cutoffs. The currently used separation cutoff as well as its resulting yield within the population is indicated in the plot's main title.

```
# generate yields plot for barcode 174
plotYields(x = re, which = "D2")
```



Option `which` will render a summary plot of all barcodes. Here, the overall yield achieved by applying the current set of cutoff values will be shown. All yield functions should behave as described above: decline, stagnation, decline. Convergence to 0 yield at low cutoffs is a strong indicator that staining in this channel did not work, and excluding the channel entirely is sensible in this case. It is thus recommended to always view the all-barcodes yield plot to eliminate uninformative populations as a too small population size may cause difficulties, especially when computing spill estimates.

```
# generate summary yields plot
plotYields(x = re, which = 0)
```



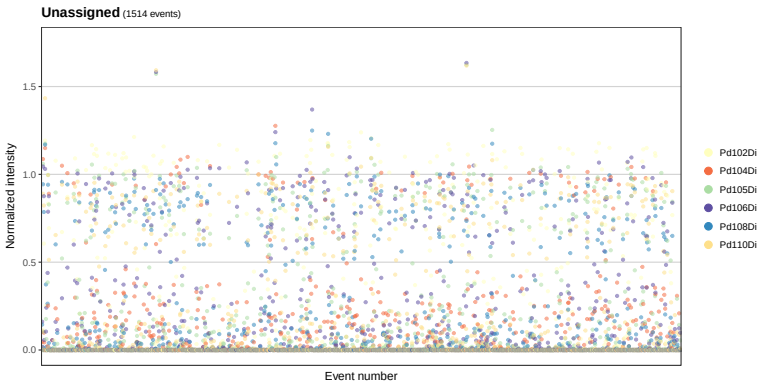
3.2.4 `applyCutoffs`: Applying deconvolution parameters

```
# apply separation and Mahalanobis distance thresholds
re <- applyCutoffs(x = re)
re
## dbFrame objectect with
```

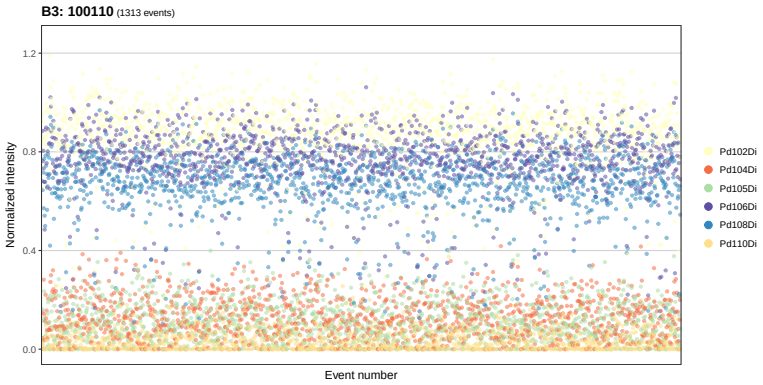
```
##  10000 events, 64 observables and 20 barcodes:
##
## Current assignments:
##       1514 event(s) unassigned
## ID    B3   B4   B5   B2   B1   D3   D1   D4   D2   D5   C4   C1
## Count 1313 1224 1208 1204 1055  345  303  302  257  252  245  213
##
## ID    C3   C2   C5   A2
## Count  206  203  154    2
##
## Separation cutoffs:
## ID    B3   B4   B5   B2   B1   D3   D1   D4   D2   D5   C4   C1
## Yield 0.08 0.06 0.07 0.08 0.08 0.22 0.11 0.08 0.27 0.07 0.18 0.09
##
## ID    C3   C2   C5   A2
## Yield 0.11 0.17 0.08 0.11
##
## Yields upon debarcoding:
##       87.75% overall yield
## ID    B3  B4     B5     B2     B1     D3     D1     D4     D2
## Yield 98% 98.49% 98.54% 97.83% 97.62% 88.83% 92.49% 94.21% 82.43%
##
## ID    D5    C4     C1     C3     C2     C5     A2
## Yield 95.9% 88.07% 89.34% 86.23% 86.97% 90.91% 18.18%
```

3.2.5 `plotEvents`: Normalized intensities for each sample

```
# generate event plot for unassigned events
set.seed(42)
plotEvents(x = re, which = 0, n_events = "all")
```



```
# generate event plot for sample B3: 100110
plotEvents(x = re, which = "B3", n_events = "all")
```



4 Compensation

CATALYST performs compensation via a two-step approach comprising:

i. identification of single positive populations via single-cell debarcoding (SCD) of single-stained beads (or cells); and,
ii. estimation of a spillover matrix (SM) from the populations identified, followed by compensation via multiplication of measurement intensities by its inverse, the compensation matrix (CM).

As shown in [REF], we can model spillover linearly, with the channel stained for as predictor, and spill-effected channels as response. Thus, the intensity observed in a given channel j are a linear combination of its real signal and contributions of other channels that spill into it. Let s_ij denote the proportion of channel j signal that is due to channel i, and w_j the set of channels that spill into channel j. Then

$$I_{j,observed} = I_{j,real} + \sum_{i \in w_j} s_{ij}$$

In matrix notation, measurement intensities may be viewed as the convolution of real intensities and a spillover matrix with dimensions number of events times number of measurement parameters

$$I_{observed} = I_{real} \cdot SM$$

Therefore, we can estimate the real signal, I_{real}, as:

$$I_{real} = I_{observed} \cdot SM^{-1} = I_{observed} \cdot CM$$

where SM^{-1} is termed compensation matrix (CM).

Because any signal not in a single stain experiment's primary channel j results from channel crosstalk, each spill entry s_{ij} can be approximated by the slope of a linear regression with channel j signal as the response, and channel i signals as the predictors, where $i \in w_j$. To facilitate robust estimates, we calculate this as the slope of a line through the medians (or trimmed means) of stained and unstained populations, m_j^+ and m_i^+, respectively. The medians (or trimmed means) computed from events that are i) negative in the respective channels; and, ii) not assigned to interacting channels; and, iii) not unassigned, m_j^- and m_i^-, respectively, are subtracted as to account for background according to:

$$s_{ij} = \frac{m_j^+ - m_j^-}{m_i^+ - m_i^-}$$

On the basis of their additive nature, spill values are estimated independently for every pair of interacting channels. The current framework exclusively takes account of interactions that are sensible from a chemical and physical point of view. As reasoned before, +/-1M channels (abundance sensitivity), the +16M channel (oxide formation) and channels measuring isotopes (impurities) are taken into consideration. The SM's diagonal entries sii are set to 1 so that spill is relative to the total signal measured in a given channel.

4.1 Compensation work-flow with *CATALYST*

4.1.1 `computeSpillmat`: Estimation of the spillover matrix

Given a flowFrame of single-stained beads (or cells) and a numeric vector of barcode masses and barcode IDs, `computeSpillmat` estimates the spillover matrix as follows. Let s_{ij} denote the portion of signal measured in channel j that is due to spill of channel i. Assuming spillover to be a linear phenomenon, we compute this fraction as the median intensity measured in the affected channel, m_j^+, over the median intensity of the spilling channel, m_i^+. The median signal of events that are i) negative in the given channel, ii) are not assigned to potentially interacting channels, and iii) are not unassigned, m_j^- and m_i^-, respectively, are substracted as to account for background:

$$s_{ij} = \frac{m_j^+ - m_j^-}{m_i^+ - m_i^-}$$

On the basis of their additive nature, spill values are estimated independently for every pair of interacting channels. The current framework exclusively takes account of interactions that are sensible from a chemical and physical point of view. Precisely, $M \pm 1$ channels (*abundance sensitivity*), the $M + 16$ channel (*oxide formation*) and channels that measure potentially contaminated metals (*isotopic impurities*; **??**) are taken into consideration. The list of mass channels that may contain isotopic contaminatons are shown below. By default, the SM's diagonal entries s_{ii} are set to 1 to make spill relative to the total.

Metal	Isotope masses
La	138, 139
Pr	141
Nd	142, 143, 144, 145, 146, 148, 150
Sm	144, 147, 148, 149, 150, 152, 154
Eu	151, 153
Gd	152, 154, 155, 156, 157, 158, 160
Dy	156, 158, 160, 161, 162, 163, 164
Tb	159
Er	162, 164, 166, 167, 168, 170
Ho	165
Yb	168, 170, 171, 172, 173, 174, 176
Tm	169
Lu	175, 176

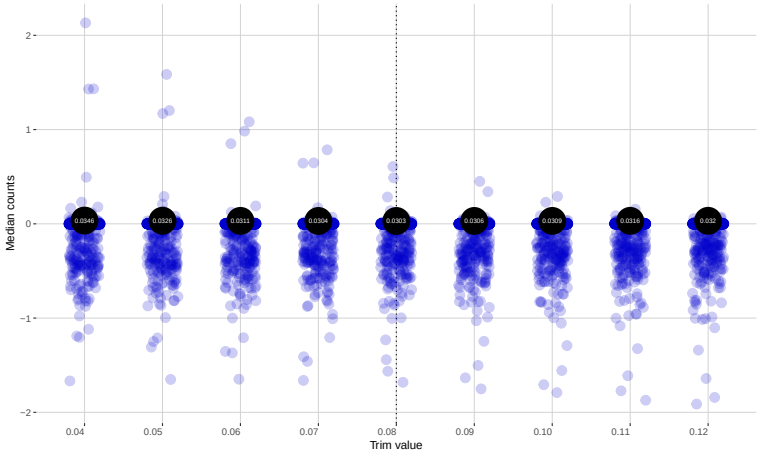
Table 1: List of isotopes available for each metal used in CyTOF. In addition to $M + 1$ and $M + 16$ channels, these mass channels are considered during estimation of spill to capture channel crosstalk that is due to isotopic contanimations.

```
# load single staining experiment
data(ss_exp)
# specify mass channels stained for
bc_ms <- c(139, 141:156, 158:176)
# debarcode
re <- assignPrelim(x = ss_exp, y = bc_ms, verbose = FALSE)
re <- estCutoffs(x = re, verbose = FALSE)
re <- applyCutoffs(x = re)
# compute compensation matrix
spillMat <- computeSpillmat(x = re)
```

4.1.2 `estTrim`: Estimation of optimal trim value

Spill value are effected my the method chosen for their estimation, that is median or mean, and, in the latter case, the specified trim percentage. To optimize results achieven upon compensation, `estTrim` will estimate the CM for a range of trim values, and evaluate, for each barcode population, the sum over squared medians of each negative channel. Along with an **optimal** trim value, the function will return population- and channel-wise median counts for each trim value. The returned value is the one that minimizes this sum. Nevertheless, it may be worth chosing a trim value that gives rise to less negative compensated data at the cost of a higher sum of squared medians. It is thus recommended to view the diagnostic plot to check the selected value, and potentially choose another.

```
# estimate trim value minimizing sum of squared
# population- and channel-wise medians upon compensation
estTrim(x = re, min = 0.04, max = 0.12, step = 0.01)
```

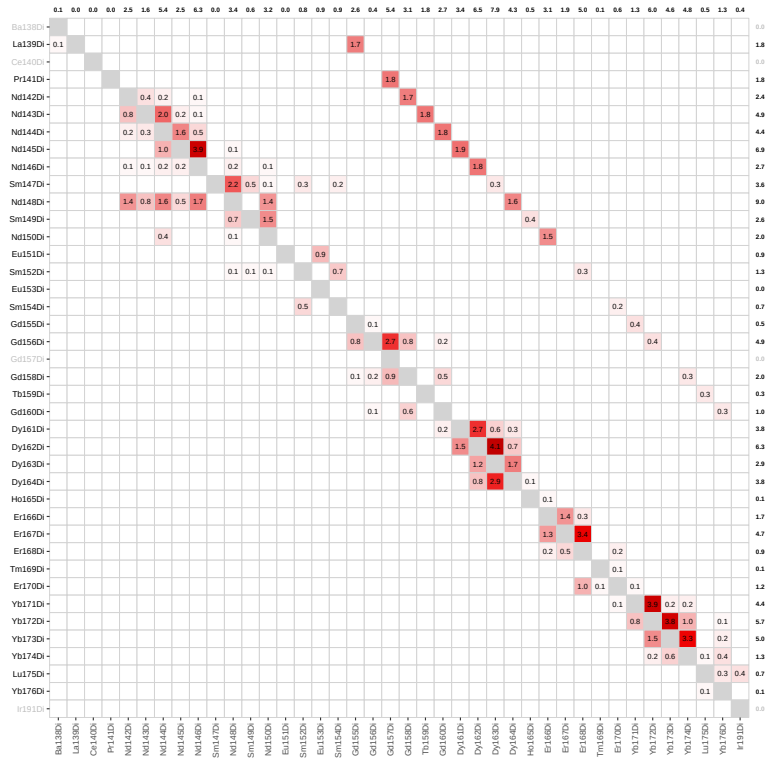

Median counts
2
1
0
−1
−2
0.0346
0.0326
0.0313
0.0304
0.0303
0.0306
0.0309
0.0316
0.032
0.04
0.05
0.06
0.07
0.08
0.09
0.10
0.11
0.12
Trim value

```
## [1] 0.08
```

4.1.3 `plotSpillmat`: Spillover matrix heat map

`plotSpillmat` provides a visualization of estimated spill percentages as a heat map. Channels not corresponding to a barcode are annotated in grey, and colours are ramped to the highest spillover value present.

```
# plot spillover matrix heat map
plotSpillmat(bc_ms = bc_ms, SM = spillMat)
```



5 References