Doublet detection in Pegasus
Version 3

Bo Li

July 12, 2021

1 Overview

Doublets consist of transcriptomes from two different cells. Doublets may be mistakenly considered as new biology (e.g. rare cell types) due to their distinction from singlets. In addition, doublets introduces noise in downstream analysis and can worsen the clustering and visualization quality. Thus, identifying and removing doublets becomes a critical data cleaning step in single-cell and single-nucleus RNA-seq (sc/snRNA-seq) data analysis. Based on Scrublet [6] paper's definition, we can classify doublets into two categories: embedded doublets and neotypic doublets. Embedded doublets are composed of highly similar cells and thus are hardly distinguishable from singlets. Neotypic doublets are composed of cells with dissimilar transcriptomes. They are the doublets that cause most trouble. Fortunately, they are also distinguishable from singlets.

Our goal is to identify and remove neotypic doublets. In this manuscript, we will describe a three-step stragety used in Pegasus to identify and remove neotypic doublets. First, Pegasus calculates doublet scores per sample using a slightly modified Scrublet [6] method. Second, Pegasus infers a doublet score cutoff between neotypic and embedded doublets per sample automatically using a method combining Gaussian mixture model and signed curvature scores. Lastly, Pegasus tests if any cluster consists of more neotypic doublet than expected using Fisher's exact test. Clustering should be performed on all samples after batch correction. Users can determine if they want to mark any statistically significant cluster as a neotypic cluster and all cells in a neotypic cluster would be marked as neotypic doublets. This last step is inspired by Pijuan-Sala et al. [4].

In the following sections, we will describe each of the three steps in details.

2 Doublet score calculation

Pegasus reimplements Scrublet [6] with slightly modifications. We reimplemented Scrublet for two reasons: 1) Scrublet source code was not maintained since July 2019; 2) a re-implementation allows us to cut many unnecessary dependencies and gives us more flexibility on future improvements.

Scrublet has three major steps: preprocessing, doublet simulation and doublet score calculation using a KNN classifier. The preprocessing step consists of 4 sub-steps (see Default Preprocessing section of the Scrublet paper): a) data normalization, b) highly variable gene selection, c) data standardization and d) PCA. In our reimplementation, we replace a) and b) with Pegasus data normalization and log transformation [log(TP100K+1)], followed by Pegasus-style highly variable gene selection [1]. It is also worth noting that we directly work on the TP100K matrix in c), instead of log(TP100K+1) matrix.

For the doublet simulation and doublet score calculation steps, we exactly follow the Scrublet method, except that we build kNN graphs using Pegasus' kNN building function [1], which utilizes the Hierarchical Navigable Small World algorithm [2]. For users' convenience, we also provide a brief derivation of how the doublet score is calculated below. More details can be found in the Scrublet paper [6].

Let r be the ratio between simulated doublets and observed doublets, $P_D^{'}(x)$ be the approximated density function of doublets and $P_{obs}(x)$ be the density function of observed cells, which can be used as an approximation of density function of singlets (assuming doublet rate is low). The probability of a simulated doublet appeared in the neighborhood of cell x becomes

$$q(x) = \frac{P_D^{'}(x)r}{P_D^{'}(x)r + P_{obs}}. \tag{1}$$

Let $\hat{\rho}$ be the expected doublet rate, the probability of x is a doublet becomes

$$\mathscr{L}(x) \approx \frac{P_D^{'}(x)\hat{\rho}}{P_D^{'}(x)\hat{\rho} + P_{obs}(x)(1-\hat{\rho})}. \tag{2}$$

Reorganize equation (1), we get

$$P_{obs}(x) = P_D^{'}(x) \cdot \frac{r(1-q)}{q} \cdot (1-\hat{\rho}). \tag{3}$$

Plug equation (3) into equation (2), we get

$$\mathscr{L}(x) = \frac{q(x)\hat{\rho}/r}{(1-\hat{\rho}) - q(x)(1-\hat{\rho}-\hat{\rho}/r)}. \tag{4}$$

Following Scrublet notations, we denote k as the average number of observed cell neighbors and k_{adj} as the total number of neighbors. By default, we have

$$\begin{aligned} k &= \lfloor 0.5 \cdot \sqrt{\text{number of cells}} \rceil, \\ k_{adj} &= \lfloor k \cdot (1+r) \rceil. \end{aligned}$$

If we put a non-informative prior $Beta(1,1)$ on $q(x)$, the expectation of $q(x)$ becomes

$$\langle q(x) \rangle = \frac{k_d(x)+1}{k_{adj}+2}, \tag{5}$$

where $k_d(x)$ is the number of simulated doublets in cell x's neighborhood. Note that the neighborhood here does not include x itself.

Plug equation (5) into equation (4), we get the formula for doublet score as

$$\langle \mathscr{L}(x) \rangle \approx \frac{\langle q(x) \rangle \hat{\rho}/r}{(1-\hat{\rho}) - \langle q(x) \rangle (1-\hat{\rho}-\hat{\rho}/r)}. \tag{6}$$

2.1 Estimate doublet rate prior automatically for 10x Genomics data

In Scrublet, users need to set a doublet rate prior parameter manually, which might be challenging. In Pegasus, we have developed a method to **automatically** estimate this prior based on total number of cells.

We assume the number of cells n entering a droplet or microwell follow a Poisson distribution parameterized by λ, i.e. $n \sim Pois(\lambda)$. Then we can estimate the doublet rate ρ as

$$\rho = \frac{P(n>1)}{P(n>0)} = \frac{(1.0 - e^{-\lambda} - \lambda e^{-\lambda})}{1.0 - e^{-\lambda}}. \tag{7}$$

λ can be interpreted as the rate of an event happening in an interval of time, where the event is a cell entering the droplet or microwell. If we denote N as the total number of cells, it is intuitive to assume that $\lambda(N)$, the rate parameter for capturing N cell in one channel, is proportional to N, or

$$\lambda(N) = c \cdot N. \tag{8}$$

Based on equations (7) and (8), we can estimate $\lambda(N)$ for 10x Genomics data from the multiplet rate table available at 10x Genomics website. Based on the table, we estimated

$$\hat{\lambda}(N) = \frac{0.00785}{500} \cdot N,$$

where 0.00785 is the estimated λ for 500 cells.

If other protocols also provide multiplet rate tables similar to 10x Genomics, we can easily estimate $\lambda(N)$ using equations (7) and (8).

In Pegasus, if users do not provide a doublet rate prior value, we automatically set $\hat{\rho}$ as

$$\hat{\rho} = \frac{(1.0 - e^{-\hat{\lambda}(N)} - \hat{\lambda}(N)e^{-\hat{\lambda}(N)})}{1.0 - e^{-\hat{\lambda}(N)}}.$$

Note that if the data are not 10x Genomics, users may consider to provide a prior value instead of using this automatic feature.

3 Doublet cutoff inference

Scrublet provides a method to determine doublet score cutoff between embedded and neotypic doublets based on simulated doublets. However, this method is far from ideal. Figure 1 showed the Scrublet histograms generated for bone marrow donor 3, channel 1 from the Immune Cell Atlas dataset. We ran Scrublet using default parameters except setting $\hat{rho} = 0.0031$. We can observe that the "ideal" cutoff should be around 0.2, while Scrublet set the threshold in the middle of the "neotypic" doublet peak.

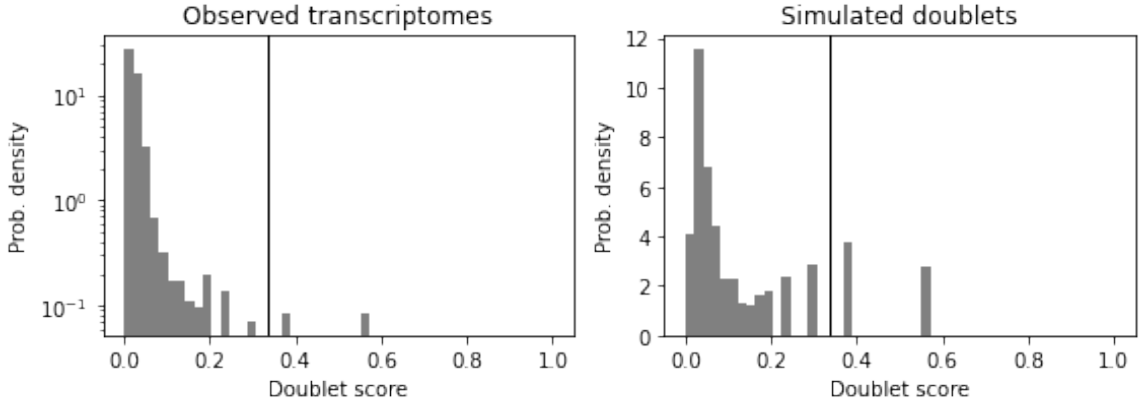

Figure 1: **Scrublet histograms for observed cells (left) and simulated doublets (right).** The vertical line indicates the cutoff.

Thus, we developed a novel method to automatically determine the cutoff in Pegasus. Our method is based on several observations from real data, which we will describe below.

3.1 Log-transformation

We observed that log-transforming the doublet scores helps us to push neotypic doublets together and have a clearer distinction between embedded and neotypic doublets. For example, we performed Kernel density estimation (KDE) on both doublet scores and log-transformed doublet scores for simulated doublets (Figure 2). We can clearly observe two peaks on the KDE plot generated from log-transformed scores. Thus, we will work on log-transformed ($\log x$) doublet scores for determining the cutoff.

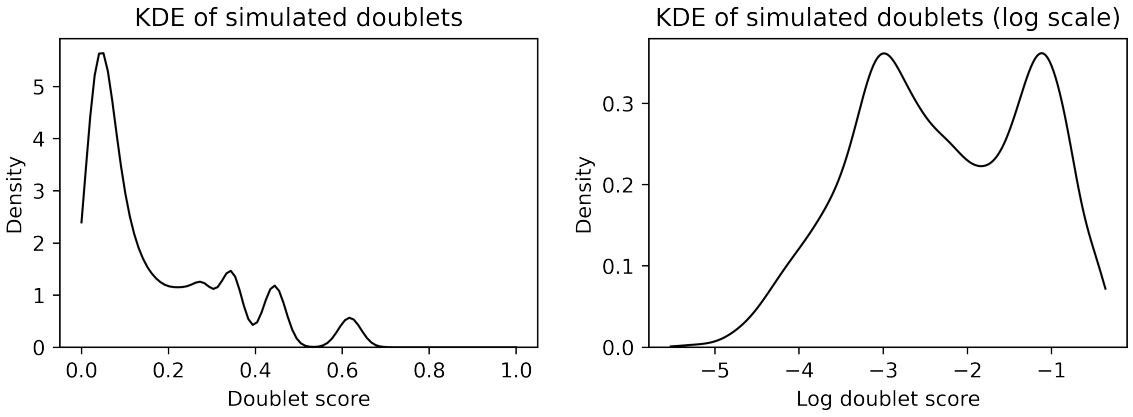

Figure 2: **KDE plots on doublet scores (left) and log-transformed doublet scores for simulated doublets.**

3.2 Peaks in the KDE plot

We observed that clear two-peaks structure in the log-transformed KDE plot (e.g. Figure 2, right panel) for many cases. For these cases, it is intuitive to set the cutoff at the position with minimal density value between the two peaks. In order to determine the appropriate cutoff, we need to identify all peaks (local maxima) from the log-transformed KDE plot. To do so, we first need to discretize the x axis (log doublet score) into a series of points. We then check each point to determine if it is a local maximum (i. e. larger than its neighbor points).

We categorize peaks into two groups (Figure 3A) based on their heights: 1) major peaks are the peaks with heights larger than α_f fraction ($\alpha_f = 0.25$) of the global maximum; and 2) minor peaks are any other peaks. Major peaks are likely to represent embedded or neotypic doublet groups and minor peaks are likely to be outliers representing only a small fraction of doublets in one doublet group. Thus, we use major peaks to determine the cutoff.

In some cases, one major peak may be splitted into two major "peaks" due to noise in data (Figure 3B). To handle this issue, we merge two adjacent major peaks if $\frac{\max(y_i, y_{i+1}) - y^{\min}_{i,i+1}}{\max(y_i, y_{i+1})} \leq \alpha_m$. Here y_i and y_{i+1} are the peak density values at the two peaks, $y^{\min}_{i,i+1}$ is the minimal density value between the peaks and α_m is the peak merging threshold ($\alpha_m = 0.06$). We describe the detailed method for finding all peaks in Algorithm 1.

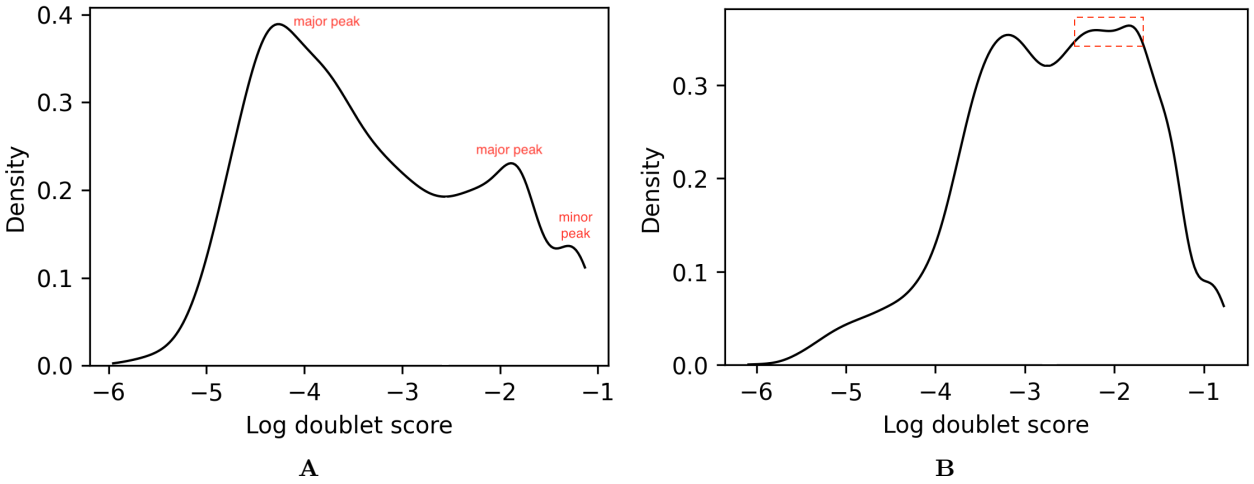

Figure 3: **Example peaks.** KDE plots estimated from a human heart sample (**A**) and a human peripheral blood sample (**B**). **A** shows two major peaks and one minor peak. **B** indicates the "splitted" major peak using a red rectangle.

Input: **sim_scores**: doublet scores for all simulated doublets;
KDE($\mathbf{x}$): a Kernel density estimation function return densities at values in vector $\mathbf{x}$;
RANGE(start, end, step): this function return a seqeunce of points in [start, end] with a increment of step between adjacent points;
$\alpha_f = 0.25$, $\alpha_m = 0.06$
Output: *merged_peaks*: merged major peaks;
major_peaks: all major peaks before merging;
minor_peaks: minor peaks;
sim_scores_log: log transformed simulated doublet scores;
$\mathbf{x}$: discrete data points in log doublet score space;
$\mathbf{y}$: density value of discrete data points

sim_scores_log $\leftarrow \log$ **sim_scores** `// log transform doublet scores;`
$min\_score \leftarrow \min($**sim_scores_log**$)$;
$max\_score \leftarrow \max($**sim_scores_log**$)$;
`// generate discrete data points for evaluating local maxima`
`// adjacent data points only contain one unique value in sim_scores_log`
$min\_gap \leftarrow$ minimum gap between adjacent scores in **sim_scores_log**;
$n\_gap \leftarrow \max(\lceil (max\_score - min\_score)/min\_gap \rceil, 200)$;
$gap \leftarrow (max\_score - min\_score)/n\_gap$;
$\mathbf{x} \leftarrow$ **RANGE**$(min\_score - gap \times 5, max\_score + gap \times 5, gap)$ `// add a margin of` $gap \times 5$ `at both sides;`
`// calculate densities at data points in x`
$\mathbf{y} \leftarrow$ **KDE**$(\mathbf{x})$;
`// search for local maxima`
$lower\_bound \leftarrow \alpha_f \cdot \max(\mathbf{y})$, $major\_peaks \leftarrow \emptyset$, $minor\_peaks \leftarrow \emptyset$;
for $i \leftarrow 3$ **to** $|\mathbf{x}| - 2$ **do** `// index starts from 1`
  if $\mathbf{y}[i-1] = \mathbf{y}[i]$ *and* $\mathbf{y}[i-2] < \mathbf{y}[i-1]$ *and* $\mathbf{y}[i] > \mathbf{y}[i+1]$
  or $\mathbf{y}[i-2] < \mathbf{y}[i-1]$ *and* $\mathbf{y}[i-1] < \mathbf{y}[i]$ *and* $\mathbf{y}[i] > \mathbf{y}[i+1]$ *and* $\mathbf{y}[i+1] > \mathbf{y}[i+2]$ **then**
    `// Determine if major peak or minor peak`
    if $\mathbf{y}[i] > lower\_bound$ **then**
      $major\_peaks \leftarrow major\_peaks \cup \{i\}$
    else
      $minor\_peaks \leftarrow minor\_peaks \cup \{i\}$
    end
  end
end
`// merge major peaks that might be produced by noise`
$curr\_peak \leftarrow \{1\}$, $merged\_peaks \leftarrow \emptyset$;
for $i \leftarrow 2$ **to** $|major\_peaks|$ **do**
  $min\_value \leftarrow \min(\mathbf{y}[major\_peaks[i] + 1 : major\_peaks[i+1]])$;
  $max\_value \leftarrow \max(\mathbf{y}[major\_peaks[i]], \mathbf{y}[major\_peaks[i+1]])$;
  if $(max\_value - min\_value)/max\_value \leq \alpha_m$ **then** `// merge peaks`
    $curr\_peak \leftarrow curr\_peak \cup \{i\}$
  else
    $merged\_peaks \leftarrow merged\_peaks \cup \{\text{argmax}_{j \in curr\_peak} \mathbf{y}[j]\}$;
    $curr\_peak \leftarrow \{i\}$
  end
end
$merged\_peaks \leftarrow merged\_peaks \cup \{\text{argmax}_{j \in curr\_peak} \mathbf{y}[j]\}$;

Algorithm 1: Algorithm to collect all merged and unmerged major peaks and all minor peaks.

3.3 Cutoff determination when multiple major peaks exist

If we only have 2 major peaks, it is straightforward to set the cutoff at the position with minimal density value between the two peaks. However, in many cases, we may have more than 2 major peaks. In this case, setting cutoff between the first two major peaks (ranked by peak density value) might not be always optimal (Figure 4). Fortunately, we can estiamte a theoritical cutoff between embedded and neotypic doublets by clustering, which can in turn guide us to find the best cutoff.

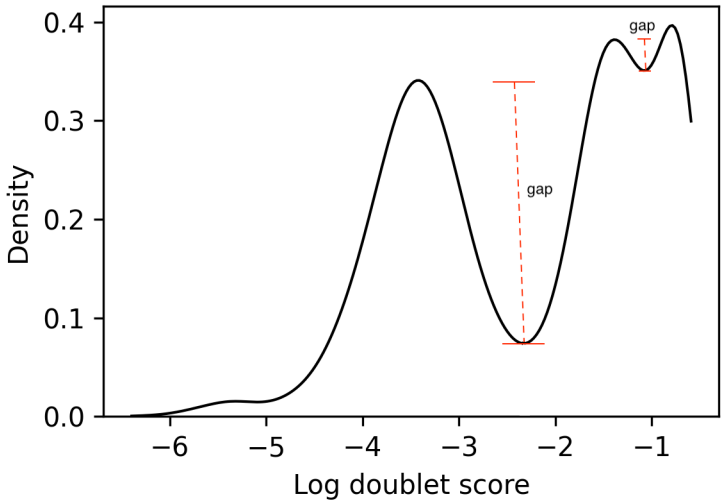

Figure 4: **Multi-major-peak examples.** KDE plots estimated from a human cord blood sample, showing three major peaks. Cutoff between the first and third peaks looks the best choice with the largest gap size.

Suppose there are n major cell types in ground truth and each cell type i occupies f_i proportion of the overal population and $\sum_{i=1}^{n} f_i = 1.0$. In the simulation process, the probability of selecting either a singlet or a embedded doublet of cell type i is

$$P_i = (1 - \hat{\rho}) * f_i + \hat{\rho} * f_i^2,$$

where $\hat{\rho}$ is the doublet prior estimated using 10x Genomics table in the previous section. The probability of generating an embedded doublet from any of the major cell types becomes

$$P_{\text{emb}} = \sum_{i=1}^{n} P_i^2 = \sum_{i=1}^{n} ((1 - \hat{\rho}) * f_i + \hat{\rho} * f_i^2)^2,$$

and the probability of generating a neotypic doublet becomes

$$P_{\text{neo}} = 1.0 - P_{\text{emb}}.$$

Note that [3] also utilized a similar technic to estimate the proportion of homotypic doublets in real data. However, their analysis did not take doublets into consideration.

In reality, we do not know the ground truth major cell types. Thus, we use the following approximations: we set $n = 5$ and perform a KMeans clustering on the PCA coordinates computed in the previous section. The resulting 5 clusters are used as a proxy of the ground truth cell types. With the theoritical neotypic doublet rate P_{neo} estimated this way, we can compute cutoffs between the first major peak and the rest major peaks and pick the cutoff that is closest to P_{neo} as the final cutoff. Please refer to Algorithm 2 for details.

In some cases, the determined cutoff results in a very small fraction (e.g. $< 10\%$) of simulated doublets identified as neotypic. In this case, we will marke all major peaks at the right of the cutoff as minor peaks and apply the rules in next subsection to find an appropriate cutoff. The fraction of neotypic doublets in simulation can be calculated as

$$P_{\text{prac}} = \frac{|\{i | \textbf{sim\_scores\_log}[i] > \textbf{x}[\textit{cutoff\_pos}]\}|}{|\textbf{sim\_scores\_log}|}.$$

Function `LocateCutoffMulti:` `// Locate the cutoff for the multi-major peaks case`

Argument: *merged_peaks*: all merged major peaks;
x: x-axis of the KDE plot;
y: y-axis of the KDE plot;
sim_scores_log: log transformed simulated doublet scores;
P_{neo}: estimated theoritical neotypica doublet rate;
Return: *cutoff_pos*: the cutoff position in **x**

$i_1 \leftarrow \text{argmax}_{i \in merged\_peaks} \mathbf{y}[i]$ `// position of the largest peak;`
$posvec \leftarrow \emptyset$;
$deltavec \leftarrow \emptyset$;
for $i \in merged\_peaks, i \neq i_1$ **do**
  $pos \leftarrow \text{argmin}_{s<j<t} \mathbf{y}[j]$;
  $P_{\text{prac}} \leftarrow \frac{|\{j|\mathbf{sim\_scores\_log}[j] > \mathbf{x}[pos]\}|}{|\mathbf{sim\_scores\_log}|}$; `// |·| refers to the size of set or array`
  $delta \leftarrow |P_{\text{neo}} - P_{\text{prac}}|$; `// |·| refers to absolute value`
  $posvec \leftarrow posvec \cup \{pos\}$;
  $deltavec \leftarrow deltavec \cup \{delta\}$;
end
$cutoff\_pos \leftarrow posvec[\text{argmin}_i\ deltavec[i]]$;
end

Algorithm 2: Algorithm to determine cutoff for the multiple major peaks case.

3.4 Cutoff determination when only one major peak exists

There are cases where we can only observe one major peak (Figure 5). When we only observe one major peak, this peak might represent either embedded doublets (Figure 5A) or neotypic doublets (Figure 5B). If the peak represents embedded doublets, we need to find the cutoff at the right side of the peak; otherwise, we find the cutoff at the left side of the peak. Thus, in order to determine an appropriate cutoff, we need to first decide which doublet group the major peak represents.

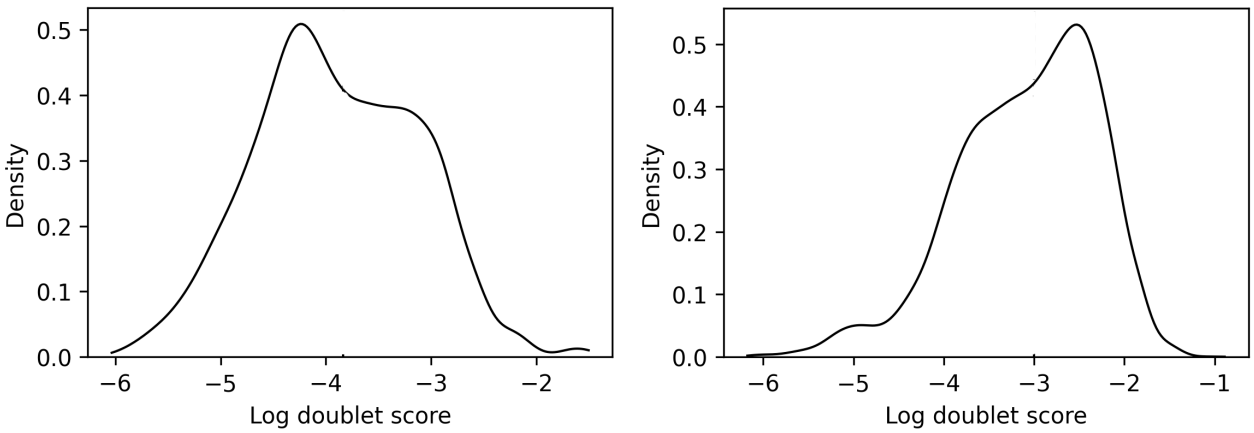

Figure 5: **Two types of single peak cases A.** KDE plot estimated from a human bone marrow sample. This plot has one major peak that is likely to represent embedded doublets. **B.** KDE plot estimated from a human peripheral blood sample. This plot has one major peak that is likely to represent neotypic doublets.

In order to determine the doublet group of the major peak, we first need to calculate $frac\_right$, the fraction of simulated doublets at the right side of the major peak. Denote the rightmost position in *major_peaks* as *peak_pos*, $frac\_right$ can be calculated as

$$frac\_right = \frac{|\{i|\mathbf{sim\_scores\_log}[i] > \mathbf{x}[peak\_pos]\}|}{|\mathbf{sim\_scores\_log}|},$$

where $|\cdot|$ denotes the size of a set or vector. If the major peak represents neotypic doublets, there must be

some embedded doublets at the leftside of the peak and thus $frac\_right$ must be smaller than 0.5. To avoid calling false positive neotypic doublets, we only consider the peak representing neotypic doublets and locate cutoff at the leftside of the peak if

$$frac\_right < 0.41, \qquad \text{or} \qquad frac\_right < 0.5 \text{ and } x\_theory + 0.05 < x[peak\_pos].$$

Otherwise, the peak presents embedded doublets and we locate cutoff at the rightside of the peak. x_theory in the above formula denotes the x coordinate of the theoritical cutoff (P_{neo}). The above criteria suggest if it is apparent that there is some embedded doublet mass at the leftside (i.e. $frac\_righ$ is small enough), we consider the peak as representing neotypic doublets. Otherwise, even if $frac\_right < 0.5$, we still need to make sure the theoritical cutoff is at the leftside of the peak ($x\_theory + 0.05 < x[peak\_pos]$).

3.4.1 Peak is embedded

If the major peak represents embedded doublets (Figure 6, left), we need to find the cutoff at the right side of the major peak. To facilitate cutoff determination, we need to calculate signed curvatures [5] of the KDE plot. The signed curvature $K_f(x)$ can be calculated as

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{1.5}},$$

where f'' and f' are the second and first derivaties of the function f (i.e. **KDE** function). We can approximate f' and f'' for data points in $\mathbf{x}$ using the five-point stencil method and calculate the signed curvature using approximated f' and f'' values (Figure 6, right).

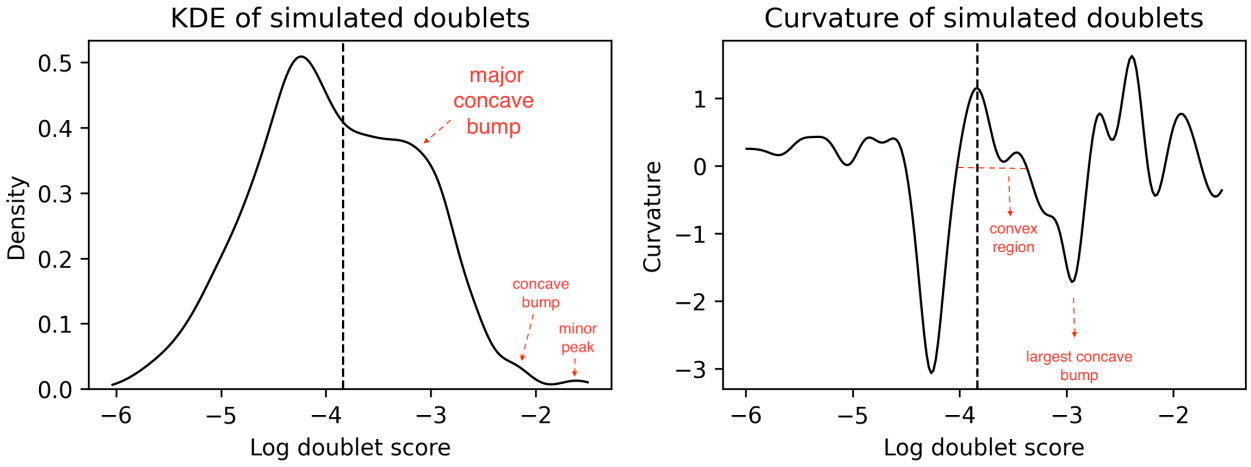

Figure 6: **KDE (left) and signed Curvature (right) plots of a human bone marrow sample.** The dashed lines indicate the cutoff. The red arrows highlight important concepts, such as concave bump, major concave bump, minor peak, convex region and largest concave bump.

Once singed curvatures are calculated, we can utilize concave bumps (Figure 6) and minor peaks to locate the cutoff. Concave bumps are local minima with negative curvature value in the signed curvature plot (Figure 6, right). We further define the concept of major concave bumps (Figure 6, left), which are concave bumps with large enough absolute curvature values compared to either the major peak or the largest concave bump (Figure 6, right). The largest concave bump is the concave bump that has the largest absolute curvature value among concave bumps between the major peak and minor peaks. Let us denote $curv\_peak$ and $curv\_glob$ as the minimal curvature value in the major peak and the largest concave bump respectively, and define $curv\_right = \max(curv\_glob, curv\_peak)$. The major concave bumps are defined as any concave bumps with the minimal curvature values smaller than γ_r fraction ($\gamma_r = 0.45$) of $curv\_right$. Note that major concave bumps should also have their minimal curvature values smaller than γ_d (an upper bound on minimal curvature values, $\gamma_d = -0.25$).

We additionally define a convex region (Figure 6, right) as an interval that the minimal curvature value within the interval is larger than γ_p ($\gamma_p = 0.05$). The cutoff should locate at a convex region between the

major peak and the leftmost major concave bump or minor peak. In particular, we pick the cutoff as the elbow point [5] (the point with maximal curvature value, see dashed vertical lines in Figure 6) among all convex regions between the major peak and the leftmost major concave bump/minor peak. The algorithm to find a cutoff for the one major peak case is illustrated in Algorithm 3.

Function `LocateCutoffRight:` `// Locate the doublet cutoff at the rightside`
  Argument: *major_peaks*: all major peaks;
    minor_peaks: all minor peaks;
    $\mathbf{x}, \mathbf{y}$: x-axis and y-axis of the KDE plot;
    $\gamma_p = 0.05$, $\gamma_r = 0.45$, $\gamma_d = -0.25$;
    calc_curv($\mathbf{x}$, $\mathbf{y}$): this function return curvature values using the five-point stencil method;
  Return: *cutoff_pos*: the cutoff position in $\mathbf{x}$

  $\mathbf{curv} \leftarrow \mathbf{calc\_curv}(\mathbf{x}, \mathbf{y})$;
  `// the major peak represents embedded doublets`
  `// calculate curv_peak`
  $s \leftarrow \max\{i | i < \max(major\_peaks) \text{ and } \mathbf{curv}[i] \geq 0.0\}$;
  $t \leftarrow \min\{i | i > \max(major\_peaks) \text{ and } \mathbf{curv}[i] \geq 0.0\}$;
  $curv\_peak \leftarrow \min_{s<i<t} \mathbf{curv}[i]$;
  `// calculate curv_glob`
  $s \leftarrow \min\{i | i > \max(major\_peaks) \text{ and } \mathbf{curv}[i] > \gamma_p\}$;
  $i_l \leftarrow \min\{i | i \in minor\_peaks \text{ and } i > \max(major\_peaks)\}$ `// locate leftmost minor peak`;
  $t \leftarrow \max\{i | i < i_l \text{ and } \mathbf{curv}[i] > \gamma_p\}$;
  $curv\_glob \leftarrow \min_{s<i<t} \mathbf{curv}[i]$;
  if $curv\_glob < \gamma_d$ **then** `// locate leftmost major concave bump`
    $curv\_right \leftarrow \max(curv\_peak, curv\_glob)$;
    $thre \leftarrow \min(curv\_right \cdot \gamma_r, \gamma_d)$;
    $i \leftarrow s + 1$;
    while $i < t$ *and not (*$\mathbf{curv}[i] < thre$ *and* $\mathbf{curv}[i-1] > \mathbf{curv}[i]$ *and* $\mathbf{curv}[i] < \mathbf{curv}[i+1]$*)* **do**
      $i \leftarrow i + 1$
    end
    $t \leftarrow \max\{j | j < i \text{ and } \mathbf{curv}[j] > \gamma_p\}$;
  end
  $cutoff\_pos \leftarrow \text{argmax}_{s \leq i \leq t} \mathbf{curv}[i]$
end

Algorithm 3: Algorithm to determine cutoff at the right side of the major peak.

3.4.2 Peak is neotypic

If the major peak consists of multiple peaks that are merged according to the criteria in the Peaks section, we collect a set of cutoffs between any adjacent peak pairs and pick the final cutoff as the cutoff closest to the theoritical cutoff *x_theory*. Otherwise, we select the cutoff as the position with maximal curvature value in the convex region between the first curvature local minima at the leftside of the theoritical cutoff *x_theory*

and peak. See Algorithm 4 for details.

Function `LocateCutoffLeft:` `// Locate the doublet cutoff at the leftside`

Argument: *major_peaks*: all major peaks;
$\mathbf{x}, \mathbf{y}$: x-axis and y-axis of the KDE plot;
$\gamma_p = 0.05$;
calc_curv($\mathbf{x}$, $\mathbf{y}$): this function return curvature values using the five-point stencil method;
Return: *cutoff_pos*: the cutoff position in $\mathbf{x}$

$\mathbf{curv} \leftarrow \mathbf{calc\_curv}(\mathbf{x}, \mathbf{y})$;
`// the major peak represents neotypic doublets`
if $|major\_peaks| > 1$ **then** `// the peak consists of multiple peaks merged`
$posvec \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $|major\_peaks| - 1$ **do**
$pos \leftarrow \text{argmin}_{major\_peaks[i]<j<major\_peaks[i+1]} \mathbf{y}[j]$;
$posvec \leftarrow posvec \cup \{pos\}$;
end
cutoff _pos $\leftarrow \text{argmin}_{pos \in posvec} |\mathbf{x}[pos] - x\_theory|$;
end
else
$t \leftarrow \max\{i | i < \min(major\_peaks) \text{ and } \mathbf{curv}[i] \geq \gamma_p\}$;
$s \leftarrow \max\{i | i < t \text{ and } \mathbf{x}[i] < x\_theory \text{ and } \mathbf{curv}[i-1] > \mathbf{curv}[i] \text{ and } \mathbf{curv}[i] < \mathbf{curv}[i+1]\}$;
cutoff_pos $\leftarrow \text{argmax}_{s \leq i \leq t} \mathbf{curv}[i]$;
end
end

Algorithm 4: Algorithm to determine cutoff at the left side of the major peak.

3.5 The overall cutoff determination algorithm

We summarize the overall algorithm described so far in Algorithm 5. Note that all parameters (α_f, α_m, γ_p, γ_r and γ_d) and thresholds are determined empirically from real data.

Input: **sim_scores**: doublet scores for all simulated doublets;
`LocateCutoffMulti`(*merged_peaks*, **x**, **y**, **sim_scores_log**, P_{neo}): return cutoff position for multiple peaks case;
`LocateCutoffRight`(*major_peaks*, *minor_peaks*, **x**, **y**, args): return cutoff position for one embedded peak;
`LocateCutoffLeft`(*major_peaks*, **x**, **y**, args): return cutoff position for one embedded peak;
Output: *cutoff*: a cutoff score applicable to observed data

Run Algorithm 1;
Calculate P_{neo} and *x_theory*;
cutoff_pos $\leftarrow -1$;
if $|merged\_peaks| \geq 2$ **then**
  cutoff_pos $\leftarrow$ `LocateCutoffMulti`(*merged_peaks*, **x**, **y**, ***sim_scores_log***, P_{neo});
  Calculate P_{prac} based on *cutoff_pos*;
  if $P_{prac} < 0.1$ **then**
    Move all major peaks on right side of *cutoff_pos* from *major_peaks* to *minor_peaks*;
    cutoff_pos $\leftarrow -1$;
  end
end
if *cutoff_pos* $== -1$ **then**
  $i \leftarrow |major\_peaks| - 1$;
  $frac\_right \leftarrow \frac{|\{j|\mathbf{sim\_scores\_log}[j] > \mathbf{x}[major\_peaks[i]]\}|}{|\mathbf{sim\_scores\_log}|}$;
  if $frac\_right < 0.41$ **or** $frac\_right < 0.5$ **and** $x\_theory + 0.05 < \mathbf{x}[major\_peaks[i]]$ **then**
    cutoff_pos $\leftarrow$ `LocateCutoffLeft`(*major_peaks*, **x**, **y**, $\cdots$) ;
  else
    cutoff_pos $\leftarrow$ `LocateCutoffRight`(*major_peaks*, *minor_peaks*, **x**, **y**, $\cdots$) ;
  end
end

Algorithm 5: Overall cutoff determination algorithm.

4 Doublet cluster identification

Once we have identified neotypic doublets, we can assess if a cluster is significantly enriched for doublets using Fisher's exact test by constructing the follow data table. We conduct Fisher's exact test for all clusters and control the False Discover Rate at $\alpha = 0.05$. Among clusters that are significantly enriched for doublets, users can determine if they want to mark some clusters in whole as doublets.

	Within cluster	**Outside cluster**	*Row total*
Singlets	**a**	**b**	a + b
Doublets	**c**	**d**	c + d
Column total	a + c	b + d	a + b + c + d

Table 1: **Data table for Fisher's exact test.** $c + d$ is the total number of identified (neotypic) doublets.

References

[1] B. Li, J. Gould, Y. Yang, S. Sarkizova, M. Tabaka, O. Ashenberg, Y. Rosen, M. Slyper, M. Kowalczyk, A.-C. Villani, T. Tickle, N. Hacohen, O. Rozenblatt-Rosen, and A. Regev. Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq. *Nature Methods*, 17(8):793–798, 2020.

[2] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[3] C. S. McGinnis, L. M. Murrow, and Z. J. Gartner. Doubletfinder: Doublet detection in single-cell rna sequencing data using artificial nearest neighbors. *Cell Systems*, 8(4):329–337.e4, 2019.

[4] B. Pijuan-Sala, J. A. Griffiths, C. Guibentif, T. W. Hiscock, W. Jawaid, F. J. Calero-Nieto, C. Mulas, X. Ibarra-Soria, R. C. V. Tyser, D. L. L. Ho, W. Reik, S. Srinivas, B. D. Simons, J. Nichols, J. C. Marioni, and B. Göttgens. A single-cell molecular map of mouse gastrulation and early organogenesis. *Nature*, 566(7745):490–495, 2019.

[5] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, 2011.

[6] S. L. Wolock, R. Lopez, and A. M. Klein. Scrublet: Computational identification of cell doublets in single-cell transcriptomic data. *Cell Systems*, 8(4):281–291.e9, 2019.