

Doublet detection in Pegasus

Bo Li

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 [5] 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 strategy used in Pegasus to identify and remove neotypic doublets. First, Pegasus calculates doublet scores per sample using a slightly modified Scrublet [5] 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. [3].

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

2 Doublet score calculation

Pegasus reimplements Scrublet [5] 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 replac a) and b) with Pegasus data normalization and log transformation $[\log(\text{TP100K}+1)]$, 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 $\log(\text{TP100K}+1)$ matrix.

For the doublet simulation and doublet score calculation steps, we exactly follow the Scrublet method, except that we built kNN graph using Pegasus’ kNN building functions [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 [5].

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

$$q(x) = \frac{P'_D(x)r}{P'_D(x)r + P_{obs}}. \quad (1)$$

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

$$\mathcal{L}(x) \approx \frac{P'_D(x)\hat{\rho}}{P'_D(x)\hat{\rho} + P_{obs}(x)(1 - \hat{\rho})}. \quad (2)$$

Reorganize equation (1), we get

$$P_{obs}(x) = P'_D(x) \cdot \frac{r(1 - q)}{q} \cdot (1 - \hat{\rho}). \quad (3)$$

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

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

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

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

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

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

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

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

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

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

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

$$\lambda(N) = c \cdot N. \quad (8)$$

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.

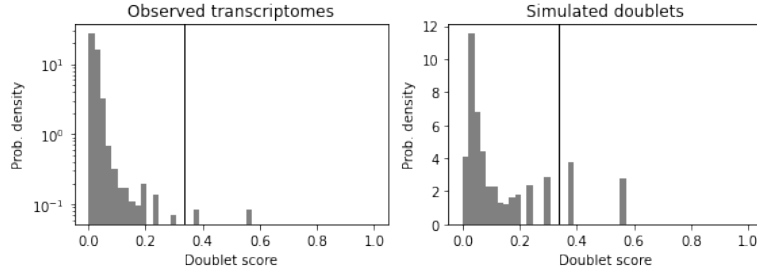


Figure 1: **Scrublet histograms for observed cells (left) and simulated doublets (right).** The vertical line indicates the cutoff.

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 in the following.

First, 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.

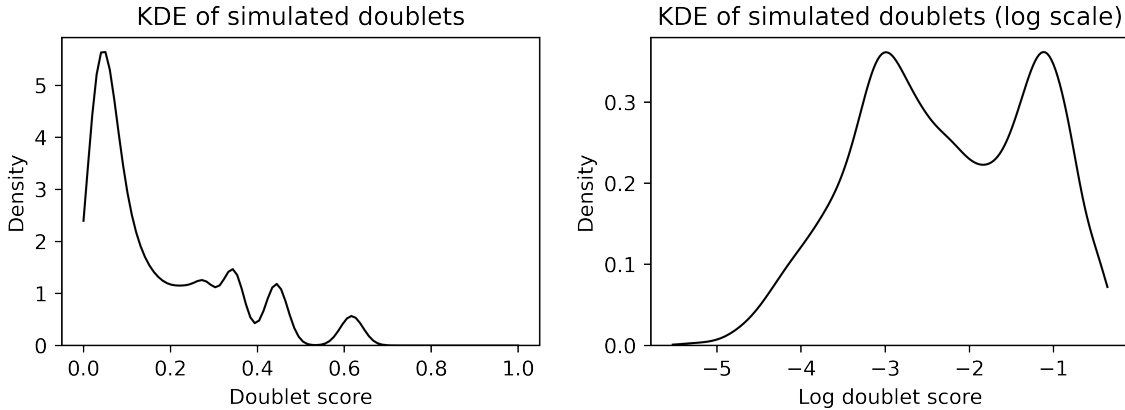


Figure 2: **KDE plots on doublet scores (left) and log-transformed doublet scores for simulated doublets.**

Input: **sim_scores**: doublet scores for all simulated doublets;
KDE(x): a Kernel density estimation function return densities at values in vector **x**;
RANGE(start, end, step): this function return a sequence of points in [start,end] with a increment of step between adjacent points;
 $\alpha_f = \frac{1}{3}$, $\alpha_m = 0.06$

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;
x: discrete data points in log doublet score space;
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( $\lceil (\text{max\_score} - \text{min\_score}) / \text{min\_gap} \rceil$ , 200);
gap  $\leftarrow$  (max_score - min_score) / n_gap;
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
y  $\leftarrow$  KDE(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 |x| - 2 do // index starts from 1
| if y[i - 1] = y[i] and y[i - 2] < y[i - 1] and y[i] > y[i + 1]
| or y[i - 2] < y[i - 1] and y[i - 1] < y[i] and y[i] > y[i + 1] and y[i + 1] > y[i + 2] then
| | // Determine if major peak or minor peak
| | if y[i]  $\geq$  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(y[major_peaks[i] + 1 : major_peaks[i + 1]]);
| max_value  $\leftarrow$  max(y[major_peaks[i]], 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  $\cup$  {argmaxj  $\in$  curr_peak y[j]};
| | curr_peak  $\leftarrow$  {i}
| end
end
merged_peaks  $\leftarrow$  merged_peaks  $\cup$  {argmaxj  $\in$  curr_peak y[j]};

```

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

Secondly, 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 or equal to α_f fraction ($\alpha_f = \frac{1}{3}$) of the global maximum value of the KDE plot; and 2) minor peaks are peaks with heights smaller than α_f . Major peaks are likely to represent embedded or neotypic doublet groups and are used to determine the cutoff between peaks. Minor peaks are more likely to represent a small fraction of one doublet group and are used to determine the cutoff between peaks. We also observed that in some cases, one major peak may be splitted into two major "peaks" due to noise in data (Figure 3B). To make our method robust to this issue, we merge two adjacent major peaks if $\frac{\max(y_1, y_2) - y_{\min}}{\max(y_1, y_2)} \leq \alpha_m$ ($\alpha_m = 0.06$), where y_1 and y_2 are the peak density value at the two major peaks and y_{\min} is the minimal density value between the two peaks. We describe the detailed method for finding all peaks in Algorithm 1.

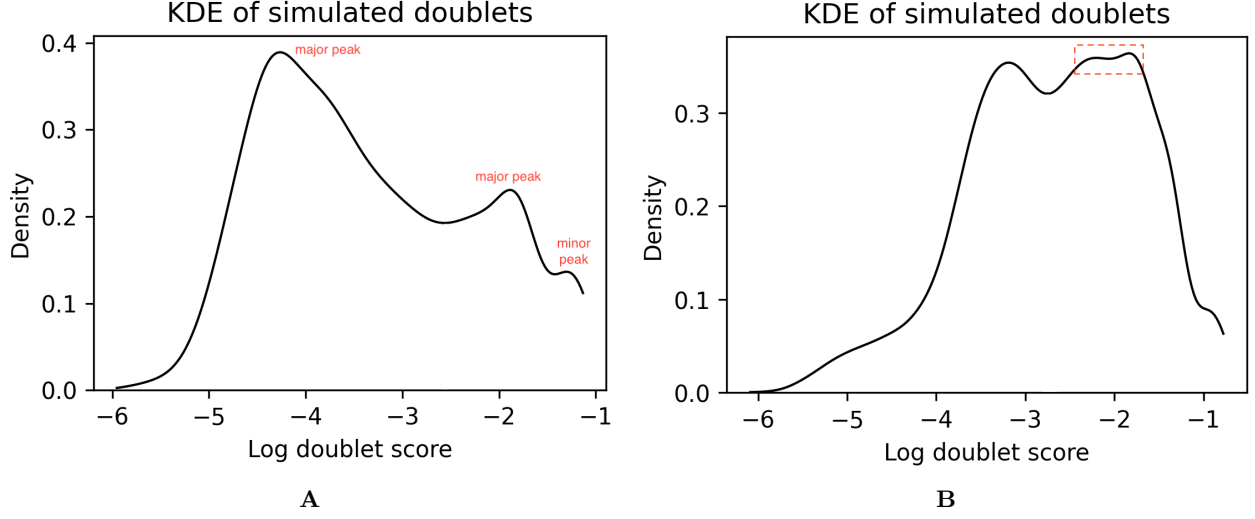


Figure 3: **Example of peaks.** **A.** KDE plot estimated from a human heart sample. This plot has two major peaks and one minor peaks. **B.** KDE plot estimated from a human peripheral blood sample. The red rectangle indicates the "splitted" major peak.

Thirdly, there are cases where we can only observe one major peak (Figure 4). When we only observe one major peak, this peak might represent either embedded doublets (Figure 4A) or neotypic doublets (Figure 4B). 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. Denote the position of the major peak in \mathbf{x} is $peak_pos$, we can calculate $frac_right$, the fraction of simulated doublets at the right side of the peak as follows:

$$frac_right = \frac{|\{i | \mathbf{sim_scores_log}[i] > \mathbf{x}[peak_pos]\}|}{|\mathbf{sim_scores_log}|},$$

where $|\cdot|$ denotes the size of a set or vector. To avoid calling false positive neotypic doublets, we only consider the peak representing neotypic doublets if

$$frac_right < \beta_r, \quad \text{where} \quad \beta_r = 0.42.$$

Otherwise, the peak presents embedded doublets.

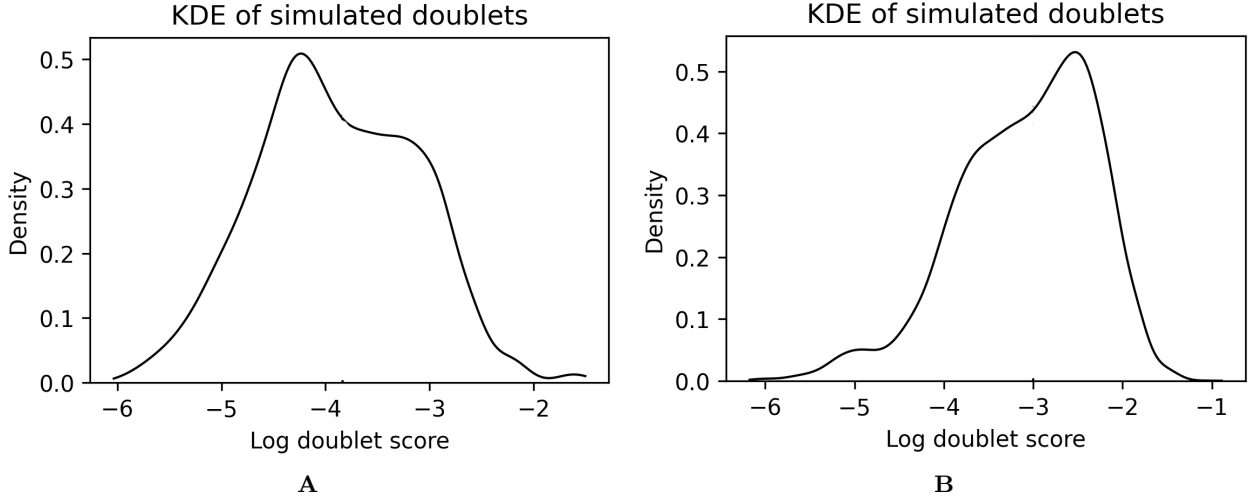


Figure 4: **Two types of single peak cases** **A.** KDE plot estimated from a human bone marrow sample. This plot has one major peak that is likely to represent embedded doublets. **B.** KDE plot estimated from a human peripheral blood sample. This plot has one major peak that is likely to represent neotypic doublets.

In addition, we adapt an addition criterion to safeguard us from calling false positive neotypic doublets. Let us assume the peak represents neotypic doublets, the peak position is $peak_pos$, and the cutoff position at the left side of the peak is $cutoff_pos$. We define $frac_left$, the fraction of simulated doublets with log scores smaller than the log score at the cutoff $\mathbf{x}[cutoff_pos]$ as

$$frac_left = \frac{|\{i | \mathbf{sim_scores_log}[i] < \mathbf{x}[cutoff_pos]\}|}{|\mathbf{sim_scores_log}|}.$$

We predict any doublets at the right side of $\mathbf{x}[cutoff_pos]$ as neotypic doublets if and only if

$$frac_left \geq \beta_l, \quad \text{where} \quad \beta_l = 0.4.$$

Otherwise, we would set the cutoff at the major peak (i. e. $peak_pos$) instead.

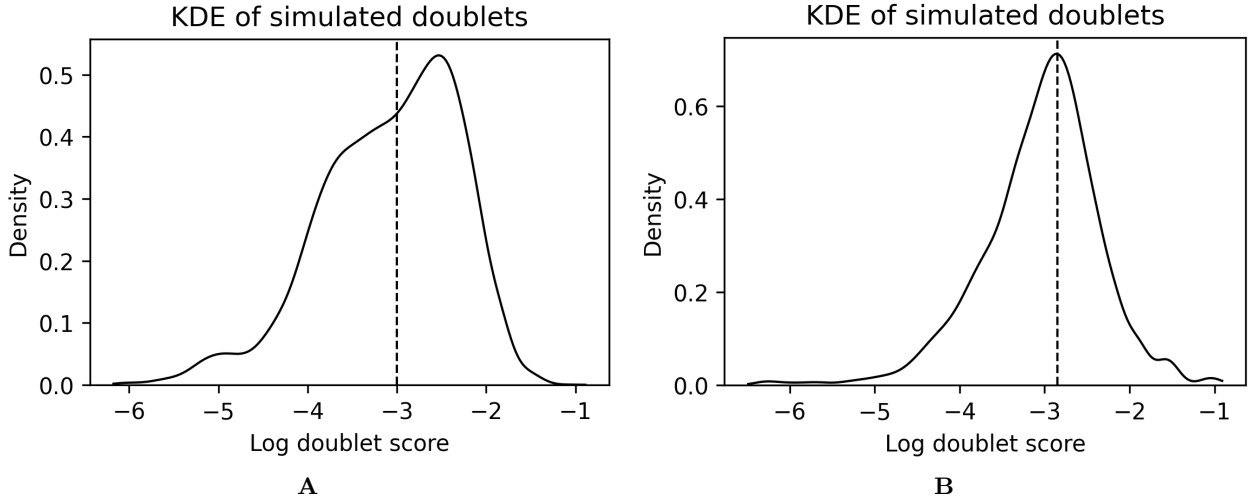


Figure 5: **Two scenarios of determining cutoffs for neotypic major peak** **A.** Same KDE plot as in Figure 4B, with the cutoff indicated as a dashed line. **B.** KDE plot estimated from a human peripheral blood sample. The cutoff is set as the peak position of the major peak.

Suppose we already have a function `LocateCutoff` to determine the cutoff from one side, we summarize the overall algorithm described so far in Algorithm 2.

```

Input: sim_scores: doublet scores for all simulated doublets;
         LocateCutoff(major_peaks, minor_peaks, dir, x, y, args): return cutoff position;
          $\beta_r = 0.42$ ,  $\beta_l = 0.4$ 
Output: cutoff: a cutoff score applicable to observed data

Run Algorithm 1;
if |merged_peaks|  $\geq 2$  then
     $i_1 \leftarrow \operatorname{argmax}_{i \in \text{merged\_peaks}} \mathbf{y}[i]$  // position of the largest peak;
     $i_2 \leftarrow \operatorname{argmax}_{i \neq i_1, i \in \text{merged\_peaks}} \mathbf{y}[i]$  // position of the second largest peak;
    if  $i_1 > i_2$  then // make sure  $i_1$  represents the left peak
        | swap  $i_1$  and  $i_2$ 
    end
    cutoff_pos  $\leftarrow \operatorname{argmin}_{i_1 < i < i_2} \mathbf{y}[i]$ ;
else
    // only one merged major peak; however, this peak might correspond to multiple
    // major peaks before merging
    cutoff_pos  $\leftarrow -1$ ;
    for  $i \leftarrow 1$  to |major_peaks| do
        frac_right  $\leftarrow \frac{|\{j | \text{sim\_scores\_log}[j] > \mathbf{x}[\text{major\_peaks}[i]]\}|}{|\text{sim\_scores\_log}|}$ ;
        if frac_right  $< \beta_r$  then
            if  $i = 1$  then // left most major peak before merging
                | cutoff_pos  $\leftarrow \text{LocateCutoff}(\text{major\_peaks}, \text{minor\_peaks}, -, \mathbf{x}, \mathbf{y}, \dots)$ 
            else
                | cutoff_pos  $\leftarrow \operatorname{argmin}_{\text{major\_peaks}[i-1] < j < \text{major\_peaks}[i]} \mathbf{y}[j]$ 
            end
            frac_left  $\leftarrow \frac{|\{j | \text{sim\_scores\_log}[j] < \mathbf{x}[\text{cutoff\_pos}]\}|}{|\text{sim\_scores\_log}|}$ ;
            if frac_left  $< \beta_l$  then
                | cutoff_pos  $\leftarrow \text{major\_peaks}[i]$ 
            end
            break;
        end
    end
    if cutoff_pos  $< 0$  then // major peak represents embedded doublets
        | cutoff_pos  $\leftarrow \text{LocateCutoff}(\text{major\_peaks}, \text{minor\_peaks}, +, \mathbf{x}, \mathbf{y}, \dots)$ 
    end
end
cutoff  $\leftarrow \exp(\mathbf{x}[\text{cutoff\_pos}])$ 

```

Algorithm 2: Overall cutoff determination algorithm.

Now let us focus on how to determine the actual cutoff from one side of the peak using function `LocateCutoff`. Let us assume we only have one major peak and the peak represents the embedded doublets (Figure 6, left). We need to find the cutoff at the right side of the major peak. In particular, we will use signed curvatures [4] of the KDE plot to determine the cutoff. 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 derivatives 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).

Let us only focus on the right side of the peak for now. 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 need to define 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 ($\gamma_r = 0.4$) of $curv_right$. Note that if $curv_glob \geq \gamma_d$ ($\gamma_d = -0.25$), we have no major concave bump.

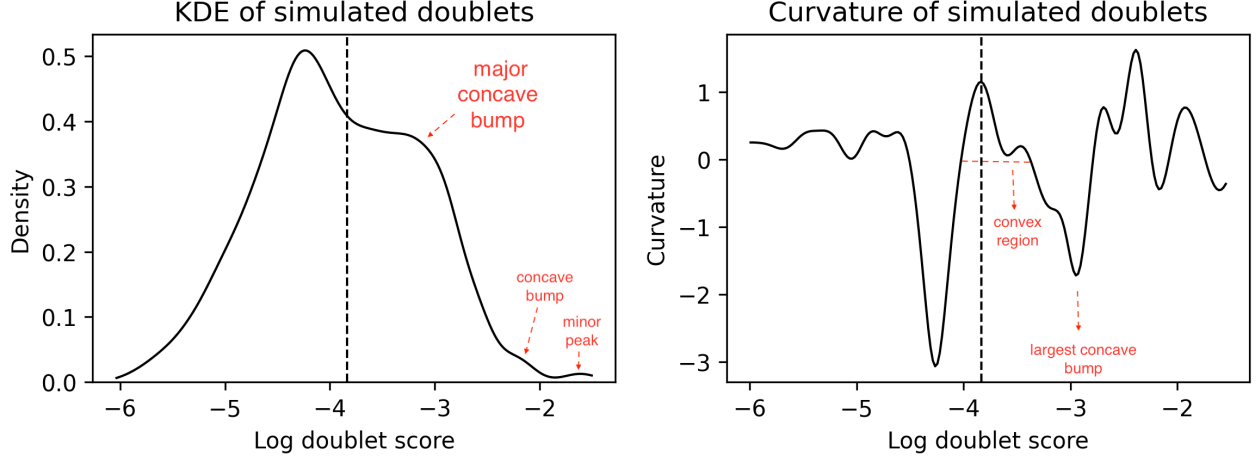


Figure 6: **KDE (left) and signed Curvature (right) plots of a human bone marrow sample.** The dashed lines indicate the cutoff. The red arrows highlight important concepts, such as concave bump, major concave bump, minor peak, convex region and largest concave bump.

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 ($\gamma_p = 0.06$). 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 [4] (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 LocateCutoff:      // Locate the doublet cutoff for the one major peak case
  Argument: major_peaks: all major peaks;
               minor_peaks: all minor peaks;
               dir: direction of search, '+' searches towards right side and '-' searches towards left side;
               x, y: x-axis and y-axis of the KDE plot;
                $\gamma_p = 0.06$ ,  $\gamma_r = 0.4$ ;
               calc_curv(x, y): this function return curvature values using the five-point stencil method;
  Return: cutoff_pos: the cutoff position in x

  curv  $\leftarrow$  calc_curv(x, y) ;
  if dir = + then
    // the major peak represents embedded doublets
    // calculate curv_peak
    s  $\leftarrow$  max{i | i < max(major_peaks) and curv[i]  $\geq$  0.0};
    t  $\leftarrow$  min{i | i > max(major_peaks) and curv[i]  $\geq$  0.0};
    curv_peak  $\leftarrow$  mins < i < t curv[i];
    // calculate curv_glob
    s  $\leftarrow$  min{i | i > max(major_peaks) and curv[i] >  $\gamma_p$ };
    il  $\leftarrow$  min{i | i  $\in$  minor_peaks and i > max(major_peaks)} // locate leftmost minor peak;
    t  $\leftarrow$  max{i | i < il and curv[i] >  $\gamma_p$ };
    curv_glob  $\leftarrow$  mins < i < t curv[i];
    if curv_glob <  $\gamma_d$  then                                     // locate leftmost major concave bump
      curv_right  $\leftarrow$  max(curv_peak, curv_glob);
      i  $\leftarrow$  s + 1;
      while i < t and not (curv[i] < curv_right  $\cdot$   $\gamma_r$  and curv[i - 1] > curv[i] and
        curv[i] < curv[i + 1]) do
        | i  $\leftarrow$  i + 1
      end
      t  $\leftarrow$  max{j | j < i and curv[j] >  $\gamma_p$ };
    end
    cutoff_pos  $\leftarrow$  argmaxs  $\leq$  i  $\leq$  t curv[i]
  else
    // the major peak represents neotypic doublets
    Calculate cutoff_pos similarly as in dir = +, use min(major_peaks) as the major peak instead
  end
end

```

Algorithm 3: Algorithm to determine cutoff for the one major peak case.

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] 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.
- [4] 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.
- [5] 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, 2019.