iScience, Volume *25*

Supplemental information

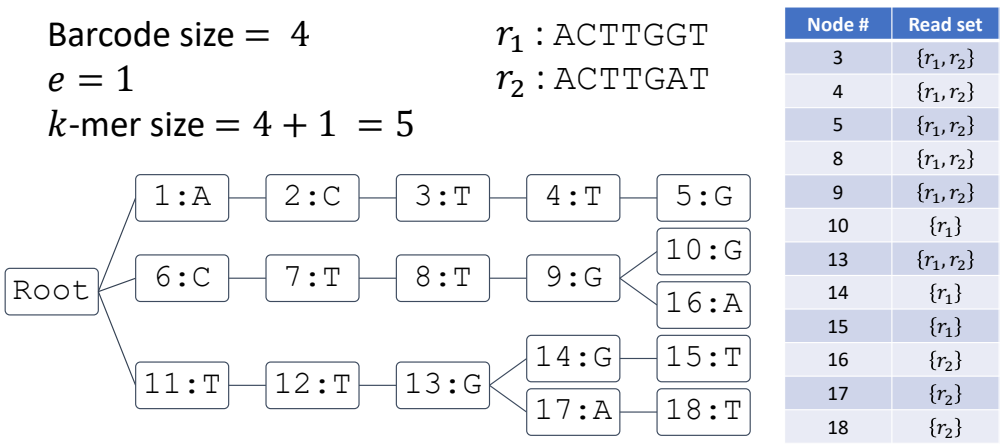
Fast and accurate matching of cellular barcodes across short-reads and long-reads of single-cell RNA-seq experiments

Ghazal Ebrahimi, Baraa Orabi, Meghan Robinson, Cedric Chauve, Ryan Flannigan, and Faraz Hach

Supplementary Material

Table S1: Statistics comparing the matches of FLAMES and scTagger against the Brute-force method, related to Figure 6.

Sample	Brute matched?	Brute match unique?	scTagger = Brute?	FLAMES = Brute?	scTagger = FLAMES?	% of LRs	# of LRs
N	✓	✓	✓	✓	✓	23.5%	1,734,479
	✓	✓	✓	✗	✗	26.7%	1,965,586
	✓	✓	✗	✓	✗	0.1%	3,742
	✓	✓	✗	✗	✗	1.5%	111,101
	✓	✗	✓	✗	✗	3.2%	235,318
	✓	✗	✗	✗	✗	20.6%	1,521,474
	✗	✗	✗	✗	✗	24.4%	1,799,495
NOA1	✓	✓	✓	✓	✓	28.4%	1,606,970
	✓	✓	✓	✗	✗	28.2%	1,597,480
	✓	✓	✗	✓	✗	0.0%	1,586
	✓	✓	✗	✗	✗	0.7%	42,423
	✓	✗	✓	✗	✗	4.0%	228,219
	✓	✗	✗	✗	✗	14.2%	803,268
	✗	✗	✗	✗	✗	24.5%	1,385,299
NOA2	✓	✓	✓	✓	✓	28.0%	1,321,345
	✓	✓	✓	✗	✗	28.4%	1,337,904
	✓	✓	✗	✓	✗	0.1%	4,581
	✓	✓	✗	✗	✗	2.1%	99,454
	✓	✗	✓	✗	✗	2.0%	93,328
	✓	✗	✗	✗	✗	15.3%	720,629
	✗	✗	✗	✗	✗	24.1%	1,136,768



Node #	Read set
3	$\{r_1, r_2\}$
4	$\{r_1, r_2\}$
5	$\{r_1, r_2\}$
8	$\{r_1, r_2\}$
9	$\{r_1, r_2\}$
10	$\{r_1\}$
13	$\{r_1, r_2\}$
14	$\{r_1\}$
15	$\{r_1\}$
16	$\{r_2\}$
17	$\{r_2\}$
18	$\{r_2\}$

Figure S1: Illustrative example of trie construction in scTagger, related to Figure 2. Assuming barcode size of 4 and maximum allowed error $e = 1$, the k-mer size is 5. The k-mers of the two LR barcode segments, r_1 and r_2, are inserted into the trie. The nodes IDs correspond to the order by which they were inserted into the trie. Nodes at layers $k - e =$ to $k + e =$ (i.e. the deepest three layers in the example) contain nodes that barcode alignment can successfully terminate within the allowed error. A map with these nodes as keys is maintained. Each node maps to a set of LR segments from which we extracted k-mers that threaded through the node at insertion time.

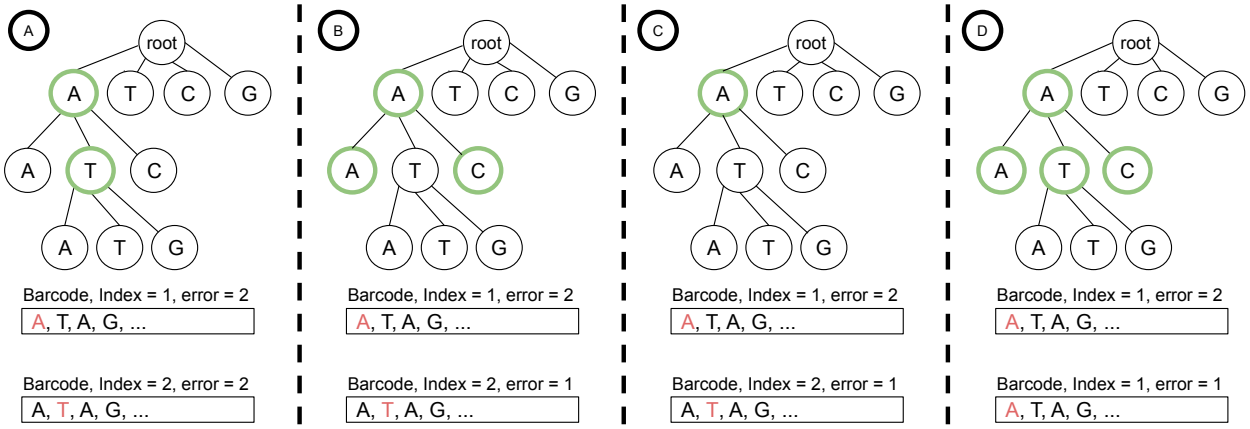


Figure S2: Illustrative example of trie querying, related to Figure 2. The different sub-queries of the search at a given node: A) Matching: If the a child node's character matches the next barcode character, we explore this branch while incrementing the barcode index. B) Mismatch: If the a child node's character does not match the next barcode character, we explore this branch while incrementing the barcode index and the total error incurred. C) Deletion: If a character is deleted from the long-read segment, we skip that corresponding barcode index by incrementing the index while stay on the same node in the trie and increment the total error incurred. D) Insertion: If a character is inserted in the long-read segment, we skip that node and to its children without increasing the barcode index while incrementing the total error incurred.

Algorithm S1 Automatic detection of the ranges of the adapter alignments on the LRs, related to Figure 7.

```
function GETRANGES(F)                    ▷ F[i] is the number adapter alignments on the i-th position
    ranges = list()
    while Q.size > 0 do
        S = sum(F)
        if S < 0.01*T  then
            break
        end if
        Q = Queue()
        Q.enqueue(P)
        first = P                                          ▷ first location in the range
        last = P                                           ▷ last location in the range
        while Q.size > 0 do
            i = Q.deque()
            if i≤ P and F[i-1] > 0.001*T then              ▷ Try expanding range to left
                Q.enqueue(i-1)
                first=i-1
            end if
            if i≥ P and F[i+1] > 0.001*T then              ▷ Try expanding range to right
                Q.enqueue(i+1)
                last=i+1
            end if
            F[i]=0
        end while
        F[first-20:last+20] = 0                            ▷ Set values in range's neighborhood to 0
        ranges.append((first,last))
    end while
    return ranges
end function
```

Algorithm S2 Depth-first search in the trie, related to Figure 2.

```
function DFS(node, error_budget, index, barcode)
    if index = barcode.length then
        edit_distance = MAX_ALLOWED_ERRORS - error_budget
        output node.LRs,edit_distance
    end if
    if error_budget > 0 then
        DFS(node, error_budget - 1 , index + 1, barcode)                ▷ Deletion
    end if
    for child in node.children do
        if barcode[index] = child.char then
            DFS(child, error_budget, index +1, barcode)                 ▷ Match
        end if
        if barcode[index] != child.char then
            DFS(child, error_budget - 1, index +1, barcode)             ▷ Mismatch
        end if
        if error_budget > 0 then
            DFS(child, error_budget - 1, index, barcode)                ▷ Insertion
        end if
    end for
end function
function QueryTrie(barcode)
    DFS(root, MAX_ALLOWED_ERRORS, 0, barcode)
end function
```