detect bead substitution error - report

tsztank

05/Februar/2020

Detecting bead substitution

NOTE: please read carefully the original Dropseq pipeline .pdf file, before reading this document. The original pipeline description can be foud here: https://github.com/broadinstitute/Drop-seq/blob/master/doc/Drop-seq_Alignment_Cookbook.pdf

It was reported previously () that sometimes two barcodes which are different are related, as one of the nucleotides changed during the substitution or even later during sequencing. This is problem is tackled by the DetectBeadSubstitutionErrors function of the Dropseq_tools. First, the identify barcode pairs of hamming-distance=1 (where each barcode which has less than 20 UMIs is filtered out). Errors at the synthesis step should be systemic, so a change pattern should be seen across all barcodes. That's why, their second step is to look for these systemic changes in the pairs of barcodes (with hamming=1) and join two barcodes if this systemic pattern is seen.

It might be the case, however, that substitution and synthesis error events do occur, but the search space (ie number of barcode) is so big, that these events will go unnoticed (as they do not occure more than by chance). In this case this problem is not solved, as there is no way to know if a substitution is indeed real, or if we have just an another barcode with hamming-distance 1.

More info in the original document, and in the source code.

Our case

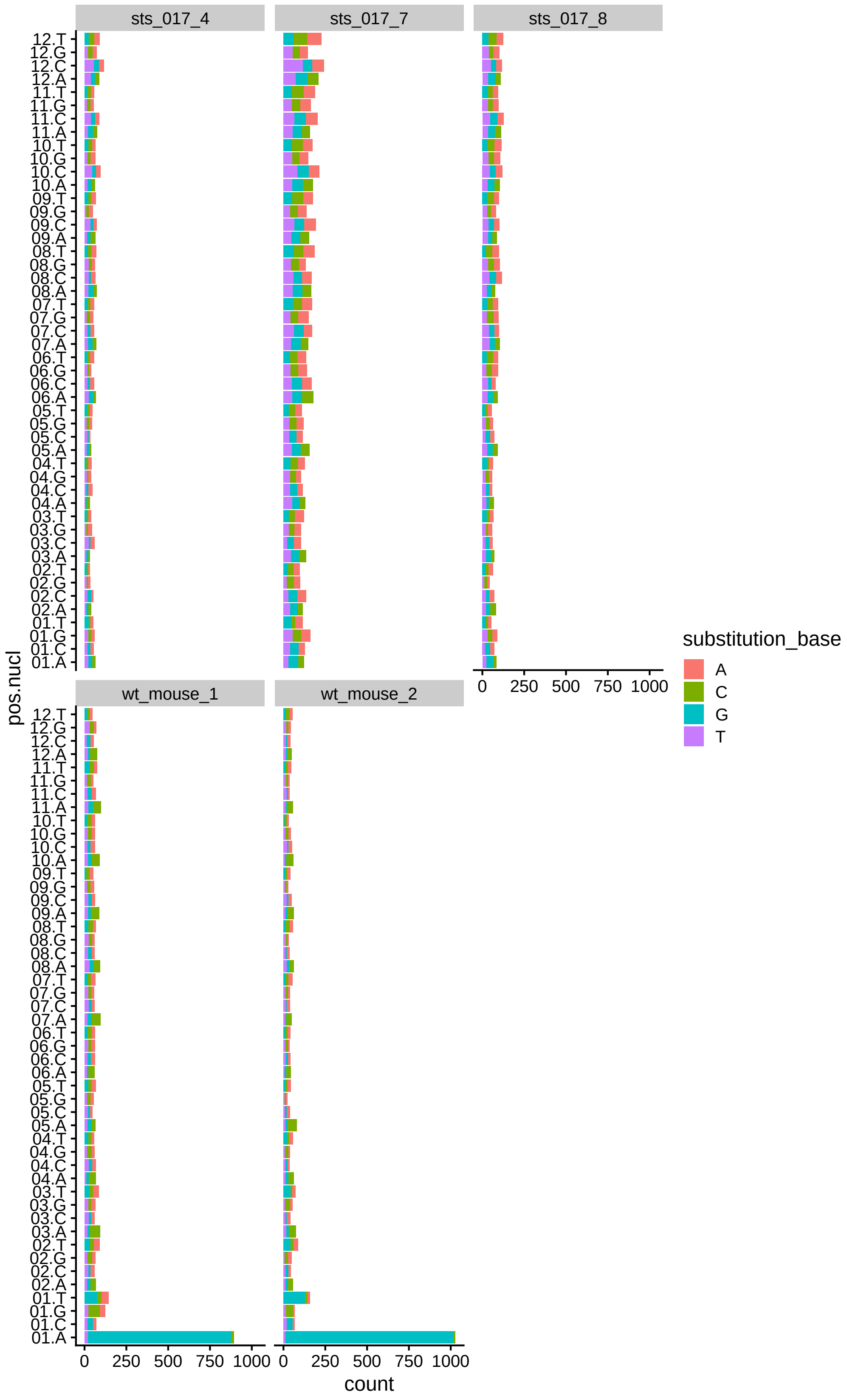
One difference with our illumina sequencing for the spacial transcriptomics project is that we use 10-times more barcodes (~100k vs ~10k), so the search space is also bigger. Ultimately this means, that the above mentioned DetectBeadSubstitutionErrors function will detect some systemic barcode switching, however these events will not occur more frequently than random. This basically means, that even if the substitution error is present, we cannot tell it apart from just a 'chance' event of getting barcodes with hamming-distance 1.

See below plot: on the y-axis you have 12 nucleotide positions for each nucleotide, and on the x axis you have the number of barcodes which have this switch (as we are looking for systemic substitution errors).

For the sts samples, you see no swap between any two nucleotides for any position, which would stand out for the rest. As a negative control, below we also plotted two examples from an another dropseq project (but this time with 10k barcodes). You can see, that there there are a few systemic conversions, which then will also be fixed by the script.

Nucleotide substitutions per position

we do not observe any systemic substitutions in the STS samples



Conclusion

In our case, it seems that the search space of the barcodes (~100k) is too big, so even if there are some real substitution erros with the barcodes, this would not be found.

This means in practice, that the DetectBeadSubstitutionErrors function will parse the .bam file, but then will not join/remove any barcodes from the analysis. So we decided to remove this function from the pipeline, at least for now.

Also: The other important Dropseq function is the DetectBeadSynthesisErrors. This occurs, when the bead is not synthesised correctly, and one nucleotide of the UMI is read as part of the Cell Barcode, and the last nucleotide of the UMI will be a T from the polyT. However, this problem is a subset of the previous problem (as the synthesis is also ought to be systemic) so this function is also not needed.