cellsnp-lite
Release v1.2.0

Xianjie Huang

Jul 04, 2021
cellsnp-lite is implemented in C. You can install it via conda or from this github repo.

1.1 Install via conda (latest stable version)

This is the recommended way to install cellsnp-lite. Lacking the potential issues of dependency, it’s simple and fast if conda is available on the machine.

1.1.1 Step 1: add config

```
conda config --add channels bioconda
conda config --add channels conda-forge
```

1.1.2 Step 2: install

To your current environment:

```
conda install cellsnp-lite
```

Or to a new environment:

```
conda create -n CSP cellsnp-lite  # you can replace 'CSP' with another env name.
```

1.2 Install from this Github Repo (latest stable/dev version)

We recommend installing cellsnp-lite via conda, as described above. The method of compiling from source code (ie., installing from this repo) is described below:
1.2.1 Step 0: install dependencies

cellsnp-lite mainly depends on zlib and HTSlib (v1.10.2+). Note that HTSlib has some extra dependencies: liblzma, libbz2, libcurl, and libcrypto. The whole list of dependencies of building cellsnp-lite is:

- gcc (have tested v4.8.5 on CentOS 7)
- autoreconf
- zlib
- HTSlib >= 1.10.2
- liblzma
- libbz2
- libcurl
- libcrypto

All dependencies should have been installed into the system before installing cellsnp-lite.

If you already have a pre-installed HTSlib then go to next step. Otherwise, a common way to install HTSlib is building from its Github repo following this HTSlib_instruction. When HTSlib is successfully installed, there should be some executable files (e.g., bgzip) and library files (eg., libhts.a and libhts.so) in htslib dir.

1.2.2 Step 1: compiling cellsnp-lite

Once all dependencies have been installed, it’s ready to compile cellsnp-lite,

```
git clone https://github.com/single-cell-genetics/cellsnp-lite.git
cd cellsnp-lite
autoreconf -iv
./configure    # Needed for choosing optional functionality
make
make install   # Optional, may need sudo privilege
```

By default, this will either build against a pre-installed HTSlib in ../htslib or build with a HTSlib in a system library path (eg., /usr/lib). You can alter this to a pre-installed HTSlib elsewhere by configuring with --with-htslib=DIR. The DIR should be either a dir containing a source tree with libhts.a/libhts.so being in DIR, or a dir containing include and 1ib subdir with libhts.a/libhts.so being in DIR/lib. Note that DIR must be absolute path and please use /home/<user> instead of ~ if needed.

Possible issue

Compilation in Step 1 prefers libhts.a than libhts.so for linking HTSlib. In rare cases that the libhts.a does not exist and libhts.so has to be used for linking, the issue error while loading shared libraries: libhts.so. 3 could happen when running cellsnp-lite, if HTSlib is not in system library path (eg., /usr/lib).

The issue, if happened, could be fixed by adding abspath to the dir containing libhts.so to the environment variable LD_LIBRARY_PATH,

```
echo 'export LD_LIBRARY_PATH=<abspath_to_hlslib_dir>:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc
```
2.1 Quick usage

Once installed, check all arguments by type cellsnp-lite -h. There are two modes of cellsnp-lite:

2.1.1 Mode 1: pileup with given SNPs

This mode genotypes single cells or bulk sample at a list of given SNPs, which could be common SNPs in human population (see compiled candidate SNPs), or called heterozygous variants from Mode 2b on its own.

Mode 1a: droplet-based single cells

Use both -R and -b to pileup droplet-based dataset (e.g., 10x Genomics) with given SNPs.

Require: a single BAM/SAM/CRAM file, e.g., from cellranger, a list of cell barcodes, a VCF file for common SNPs. This mode is recommended comparing to mode 2, if a list of common SNP is known, e.g., human (see Candidate_SNPs)

```
cellsnp-lite -s $BAM -b $BARCODE -0 $OUT_DIR -R $REGION_VCF -p 20 --minMAF 0.1 --minCOUNT 20 --gzip
```

As shown in the above command line, we recommend filtering SNPs with <20UMIs or <10% minor alleles for downstream donor deconvolution, by adding --minMAF 0.1 --minCOUNT 20

Besides, special care needs to be taken when filtering PCR duplicates for scRNA-seq data by including DUP bit in exclFLAG, for the upstream pipeline may mark each extra read sharing the same CB/UMI pair as PCR duplicate, which will result in most variant data being lost. Due to the reason above, cellsnp-lite by default uses a non-DUP exclFLAG value to include PCR duplicates for scRNA-seq data when UMItag is turned on.

Mode 1b: well-based single cells or bulk

Use -R but not -b to pileup well-based dataset (e.g., SMART-seq2) with given SNPs.

Require: one or multiple BAM/SAM/CRAM files (bulk or smart-seq), their according sample ids (optional), and a VCF file for a list of common SNPs. BAM/SAM/CRAM files can be input in comma separated way (-s) or in a list file (-S).

```
cellsnp-lite -s $BAM1,$BAM2 -I sample_id1,sample_id2 -0 $OUT_DIR -R $REGION_VCF -p 20 --cellTAG None --UMItag None --gzip
```

```
cellsnp-lite -S $BAM_list_file -i sample_list_file -0 $OUT_DIR -R $REGION_VCF -p 20 --cellTAG None --UMItag None --gzip
```

(continues on next page)
Set filtering thresholds according to the downstream analysis. Please add --UMItag None if your bam file does not have UMIs, e.g., smart-seq and bulk RNA-seq.

## 2.1.2 Mode 2: pileup whole chromosome(s) without given SNPs

Recommend filtering SNPs with <100UMIs or <10% minor alleles for saving space and speed up inference when pileup whole genome: --minMAF 0.1 --minCOUNT 100.

**Note:** This mode may output false positive SNPs, for example somatic variants or false caused by RNA editing. These false SNPs are probably not consistent in all cells within one individual, hence confounding the demultiplexing. Nevertheless, for species, e.g., zebrafish, without a good list of common SNPs, this strategy is still worth a good try.

### Mode 2a: droplet based single cells without given SNPs

Don’t use -R but use -b to pileup whole chromosome(s) without given SNPs for droplet-based dataset (e.g., 10x Genomics).

This mode requires inputting a single BAM/SAM/CRAM file with cell barcoded (add -b):

```
# 10x sample with cell barcodes
cellsnp-lite -s $BAM -b $BARCODE -O $OUT_DIR -p 22 --minMAF 0.1 --minCOUNT 100 --gzip
```

Add --chrom if you only want to genotype specific chromosomes, e.g., 1, 2, or chrMT.

**Note:** Mode 2a does joint calling and genotyping, but it is substantially slower than calling first in a bulk manner by Mode 2b followed by genotyping in Mode 1a. Otherwise, it is handy for small chromosomes, e.g., mitochondrial.

### Mode 2b: well-based single cells or bulk without SNPs

Don’t use -R and -b to pileup whole chromosome(s) without given SNPs for well-based dataset (e.g., SMART-seq2).

This mode requires inputting one or multiple BAM/SAM/CRAM file(s) of bulk or smart-seq.

```
# a bulk sample without cell barcodes and UMI tag
cellsnp-lite -s $bulkBAM -I Sample0 -O $OUT_DIR -p 22 --minMAF 0.1 --minCOUNT 100 --cellTAG None --UMItag None --gzip
```

Add --chrom if you only want to genotype specific chromosomes, e.g., 1, 2, or chrMT.
### 2.2 Notes

The **Too many open files** issue has been fixed (since v1.2.0). The issue is commonly caused by exceeding the `RLIMIT_NOFILE` resource limit (ie. the max number of files allowed to be opened by system for single process), which is typically 1024. Specifically, in the case of `M` input files and `N` threads, cellsnp-lite would open in total about `M*N` files. So the issue would more likely happen when large `M` or `N` is given. In order to fix it, cellsnp-lite would firstly try to increase the limit to the max possible value (which is typically 4096) and then use a fail-retry strategy to auto detect the most suitable number of threads (which could be smaller than the original nthreads specified by user).

The command line option `--maxFLAG` is now deprecated (since v1.0.0), please use `--inclFLAG` and `--exclFLAG` instead, which are more flexible for reads filtering. You could refer to the explain_flags page to easily aggregate and convert all flag bits into one integer. One example is that the default exclFLAG value (without using UMIs) is 1796, which is calculated by adding four flag bits: UNMAP (4), SECONDARY (256), QCFAIL (512) and DUP (1024).

### 2.3 Full parameters

Here is a list of full parameters for setting (`cellsnp-lite -V` always give the version you are using):

<table>
<thead>
<tr>
<th>Usage:</th>
<th>cellsnp-lite [options]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options:</td>
<td></td>
</tr>
<tr>
<td><code>-s, --samFile</code></td>
<td>Indexed sam/bam file(s), comma separated multiple samples.</td>
</tr>
<tr>
<td></td>
<td>Mode 1&amp;2: one sam/bam file with single cell.</td>
</tr>
<tr>
<td></td>
<td>Mode 3: one or multiple bulk sam/bam files, no barcodes needed, but sample ids and regionsVCF.</td>
</tr>
<tr>
<td><code>-S, --samFileList</code></td>
<td>A list file containing bam files, each per line, for Mode 3.</td>
</tr>
<tr>
<td><code>-O, --outDir</code></td>
<td>Output directory for VCF and sparse matrices.</td>
</tr>
<tr>
<td><code>-R, --regionsVCF</code></td>
<td>A vcf file listing all candidate SNPs, for fetch each variants.</td>
</tr>
<tr>
<td><code>-T, --targetsVCF</code></td>
<td>Similar as -R, but the next position is accessed by streaming rather than indexing/jumping (like -T in samtools/bcftools mpileup).</td>
</tr>
<tr>
<td><code>-b, --barcodeFile</code></td>
<td>A plain file listing all effective cell barcode.</td>
</tr>
<tr>
<td><code>-i, --sampleList</code></td>
<td>A list file containing sample IDs, each per line.</td>
</tr>
<tr>
<td><code>-I, --sampleIDs</code></td>
<td>Comma separated sample ids.</td>
</tr>
<tr>
<td><code>-V, --version</code></td>
<td>Print software version and exit.</td>
</tr>
<tr>
<td><code>-h, --help</code></td>
<td>Show this help message and exit.</td>
</tr>
<tr>
<td>Optional arguments:</td>
<td></td>
</tr>
<tr>
<td><code>--genotype</code></td>
<td>If use, do genotyping in addition to counting.</td>
</tr>
<tr>
<td><code>--gzip</code></td>
<td>If use, the output files will be zipped into BGZF format.</td>
</tr>
<tr>
<td><code>--printSkipSNPs</code></td>
<td>If use, the SNPs skipped when loading VCF will be printed.</td>
</tr>
<tr>
<td><code>--chrom</code></td>
<td>Number of subprocesses [1]</td>
</tr>
<tr>
<td><code>--cellTAG</code></td>
<td>The chromosomes to use, comma separated [1 to 22]</td>
</tr>
<tr>
<td><code>--cellTAG</code></td>
<td>Tag for cell barcodes, turn off with None [CB]</td>
</tr>
<tr>
<td><code>--UMItag</code></td>
<td>Tag for UMI: UR, Auto, None. For Auto mode, use UR if barcodes is inputted, otherwise use None. None mode means no UMI but read counts [Auto]</td>
</tr>
<tr>
<td><code>--minCOUNT</code></td>
<td>Minimum aggregated count [20]</td>
</tr>
<tr>
<td><code>--minMAF</code></td>
<td>Minimum minor allele frequency [0.00]</td>
</tr>
<tr>
<td><code>--doubletGL</code></td>
<td>If use, keep doublet GT likelihood, i.e., GT=0.5 and GT=1.5.</td>
</tr>
</tbody>
</table>

(continues on next page)
Read filtering:

- **--inclFLAG STR|INT**  Required flags: skip reads with all mask bits unset []
- **--exclFLAG STR|INT**  Filter flags: skip reads with any mask bits set [UNMAP,SECONDARY, QCFAIL]

  (when use UMI) or UNMAP,SECONDARY,QCFAIL,DUP (otherwise)]
- **--minLEN INT**  Minimum mapped length for read filtering [30]
- **--minMAPQ INT**  Minimum MAPQ for read filtering [20]
- **--countORPHAN**  If use, do not skip anomalous read pairs.

Note that the "--maxFLAG" option is now deprecated, please use "--inclFLAG" or "--exclFLAG" instead.

You can easily aggregate and convert the flag mask bits to an integer by referring to: 
https://broadinstitute.github.io/picard/explain-flags.html
A quality list of candidate SNPs (usually common SNPs) are important for mode 1. If a list of genotyped SNPs is available, it can be used to pile up. Alternatively, for human, common SNPs in population that have been identified from consortia can also be very good candidates, e.g., gnomAD and 1000_Genome_Project. For the latter, we have compiled a list of 7.4 million common variants (AF>5%) with this bash script and stored in this folder.

In case you want to lift over SNP positions in vcf file from one genome build to another, see our LiftOver_vcf wrap function.
If you find cellsnp-lite is useful for your research, please cite:

Xianjie Huang, Yuanhua Huang, Cellsnp-lite: an efficient tool for genotyping single cells, Bioinformatics, 2021; btab358, https://doi.org/10.1093/bioinformatics/btab358
5.1 Release v1.2.0 (04/12/2020)

• add -T cmdline option, which is similar with samtools/bcftools mpileup -T for the next position is accessed by streaming rather than index-jumping, which is different from -R. Similar with -R, cellsnp-lite -T would use REF & ALT of the input vcf.
• fix a bug of not resetting thpool for too many open files issue.

5.2 Release v1.1.2 (02/12/2020)

• fix the issue of too many open files by 1) increase the soft limit of rlimit; 2) let nthreads auto fit nfiles

5.3 Release v1.1.1 (28/11/2020)

• use only single copy of hdr & idx in csp_bam_fs for fetch method
• use only single copy of hdr (and no idx) for pileup method
• merge nthread > 1 and nthread = 1 for fetch method
• merge nthread > 1 and nthread = 1 for pileup method

5.4 Release v1.1.0 (26/11/2020)

• split cellsnp_utils.h & general_utils.h & cellsnp.c into separate modules and scripts (small .h and .c files)
• bgzip temporary files
5.5 Release v1.0.1 (17/11/2020)

• fix a bug of m2 & k2 in csp_infer_allele() which could lead to error AD and error MAF calculation.
• fix a bug of not allocating space for sample_id
• add badges

5.6 Release v1.0.0 (15/10/2020)

• add mode 2
• replace –maxFLAG with –inclFLAG and –exclFLAG
• always filter unmapped reads

5.7 Release v0.3.1 (22/07/2020)

• turn off the PCR duplicate filtering by default (~maxFLAG), as it is not well flagged in CellRanger, hence may result in loss of a substantial fraction of SNPs.

5.8 Release v0.3.0 (05/06/2020)

• fix a bug of using read.qqual and read.query_alignment_sequence in pileup_bases() and fetch_bases(), which could cause error when CIGAR string includes the ‘I’ (Insertion) op
• fix a bug of repetitive UMIs existing in different cells when grouping and counting UMIs
• the pos in the output vcf of mode 2 is switched from 0-based to 1-based

5.9 Release v0.1.8 (06/02/2020)

• Mode 2 now supports pile up chromosomes for a single bulk sample
• Mode 3 now supports multiple bam files in a list file

5.10 Release v0.1.7 (04/10/2019)

• fix a bug when chromosome is not in the bam file
• support barcodes.tsv.gz
• liftOver supports bgzip compress
• add vcf format to cellSNP.base.vcf.gz
5.11 Release v0.1.6 (14/07/2019)

- support saving to sparse matrices: Please use -O for out directory instead of -o for VCF output only. Also, you can use sparseVCF.py to convert existing VCF.gz into sparse matrices
- turn off breaking from warnings
- change P_error with BQ ranges [0.25, 45]
- h5py is not a required dependent package anymore

5.12 Release v0.1.5 (02/07/2019)

- fix a bug in qual_vector when the Quality (Phred) scores is 0, i.e., ASCII Code “!”, and it will give a P_error as 1, hence fail with log transformation. Now, I limited the P_error to 0.9999.

5.13 Release v0.1.4 (24/06/2019)

- use bgzip by default if bgzip is executable, otherwise use gzip
- change GL to PL: Phred-scaled genotype likelihoods and move to after OTH tag
- filter_reads is not in use anymore and the filtering is combined into fetch_bases or pileup_bases
- slightly optimise the memory by not keeping all reads but only positions

5.14 Release v0.1.3 (12/06/2019)

- Fix a minor bug for when loading unzipped vcf file.

5.15 Release v0.1.2 (10/06/2019)

- turn off the defaul HDF5 file output, but keep it optional.

5.16 Release v0.1.1 (09/06/2019)

- support output in hdf5 format for sparse matrix. To convert .vcf.gz to hdf5 file, you can use this script: https://github.com/huanyh09/cellSNP/blob/master/test/VCF_convert.py
5.17 Release v0.1.0 (21/05/2019)

- support the estimate the genotype and genotype likelihood for each cell. The GT is for 0/0, 1/0, 1/1, while the genotype likelihood is for 0/0, 1/0, 1/1, and 0/0+1/0, 1/1+1/0. The genotype estimate is based on the this paper (table 1; same as supp table S3 in Demuxlet paper): [https://doi.org/10.1016/j.ajhg.2012.09.004](https://doi.org/10.1016/j.ajhg.2012.09.004)
- cell tag changed from CR to CB and the lane info is kept
- pileup whole genome uses the same reads filtering as pile up positions
- add test files (note, the bam file is 19G)
- require pysam>=0.15.2 to get the qqual for each base call in the reads

5.18 Release v0.0.8 (21/12/2018)

- update the default setting that UMI tag is not in use in bulk RNA-seq, as UMI is cell specific in pseudo-bulk RNA-seq, hence better turn it UMI off by default
- support output file in the same path of command line
- support cram input file, besides bam/sam
- update readme file, especially for processed common variants from 1000 genome project ([https://sourceforge.net/projects/cellsnp/files/SNPlist/](https://sourceforge.net/projects/cellsnp/files/SNPlist/))

5.19 Release v0.0.7 (04/10/2018)

- change the header of the VCF file to be more suitable for bcftools
- realise the issue of heavy memory consuming, which even kills the jobs in cluster. The memory taken increase linearly to the number of processors used. When using 20 CUPs, >20G memory is recomend for >5K cells. Solution for higher memory efficiency will be proposed in future.

5.20 Release v0.0.6 (29/09/2018)

- fix the bug in pileup a list of positions with `pysam-fetch`: input wrong REF and ALT bases.
- support pileup a list of positions for multiple bulk samples
- check liftOver works fine: the last part of the SNPs have matched REF in fasta file.
- polish the printout log: label the three modes:
  - Mode 1: Pileup a list of positions for single cells (most common)
  - Mode 2: Pileup whole genome for single cells
  - Mode 3: Pileup a list of positions for (multiple) bulk sample(s)
5.21 Release v0.0.5 (24/09/2018)

- pileup a list of positions with `pysam-fetch`, which may return more reads than `pysam-pileup`. This feature requires further check
- change vcf file header to be more compatible with bcftools
- support turning cell-barcode off to return a sample level only

5.22 Release v0.0.4 (25/08/2018)

- pileup the whole genome for 10x single-cell RNA-seq data
- Note, post-filetering is needed as the current filtering doesn’t consider the heterozygous genotype for all donors.
6.1 Q1. No reads captured from bam file

A: If there is no read captured from the bam file, there can be multiple reasons:

- your reads don’t have UMI tag (please --UMI.tag None) or the UMI tag is not UR (please specify)