SNP finding with Cortex and Bubbleparse

Richard Leggett
richard.leggett@tsl.ac.uk

March 25, 2011

1 Introduction

In this manual, we describe the installation and use of two tools, which together provide a method for SNP prediction and ranking based upon raw sequence reads alone and without the use of a reference.

Cortex Bub is a variant of the Cortex Con genome assembler which assembles reads into a de Bruijn graph structure and then traverses the graph, looking for the characteristic topological features which indicate the presence of polymorphisms. The tool will output a FASTA format file of contigs containing the predicted polymorphisms, as well as a companion text file of coverage information for each contig.

Bubbleparse accepts the output of Cortex Bub and parses the contigs, classifying them according to type and ranking them according to a heuristic which takes into account the coverage and quality of the input data.

This manual concentrates on the SNP finding variant of Cortex and a description of the options relating to this functionality. Some other Cortex options are described in passing, but readers are recommended to refer to the Cortex manual for a full description.

2 Key concepts

2.1 Bubbles

The presence of SNPs in input sequence data causes the formation of structures in the de Bruijn graph known as bubbles (Figure 1). Often, the presence of branches on bubble paths causes more complicated structures to form (Figure 2). A single SNP will result in bubbles with identical path lengths of size k - that is, the number of nodes on each path

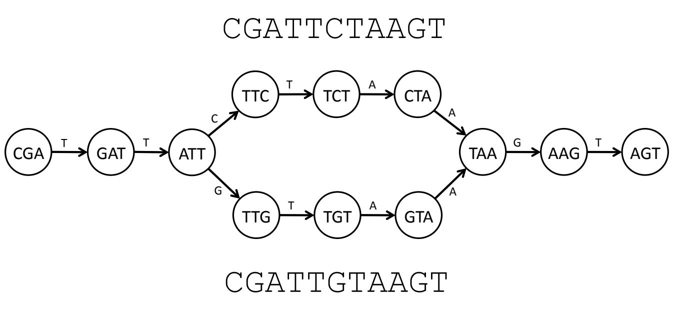

Figure 1: Example of a bubble structure in the de Bruijn graph, resulting from the presence of a SNP. The text above and below the graph shows the sequences obtained by walking the top and bottom paths through the bubble.

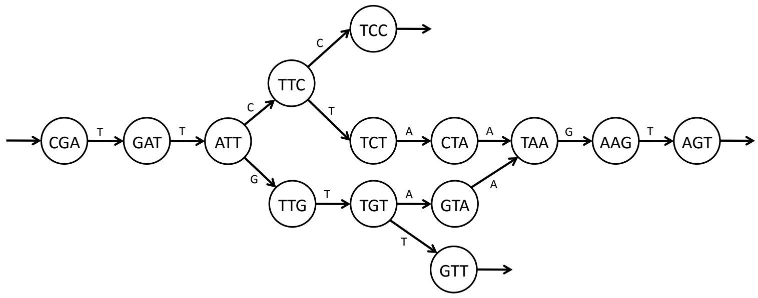

Figure 2: A more complicated SNP, with branches occurring on the paths through the bubble.

through the bubble will equal the size of the kmer. Indels also produce bubble structures, but with paths of varying length.

Cortex will traverse a de Bruijn graph assembly and attempt to locate all the bubbles. However, it is not only SNPs and indels that cause bubbles to form - sequencing errors, read errors, repeats and other features can all result in the production of bubbles in the graph. Thus, the job of bubbleparse is to try to rank the bubbles found by Cortex in an effort to facillitate discovery of the most interesting ones.

2.2 Colours and types

Both pieces of software make use of the concept of colours. Sets of one or more input files are assigned a colour represented by a number. In the plant realm, a common approach would be to assign one colour to reads from a susceptible bulk and one colour to reads from a resistant bulk. When building a de Bruijn graph representation of the reads, Cortex will keep track of the coverage count for each colour and bubbleparse can then use this to classify and rank potential SNPs according to expected proportions of resistant and susceptible alleles.

The colours of the reads that contribute the kmers on each path through the bubble are used in the type classification system adopted by bubbleparse (Figure 3). A type consists of a series of numbers, separated by commas. Each number corresponds to a path through the bubble and provides a count of the number of colours which contain that path. Thus a 2,1 bubble has two paths, one with two colours represented on it, the other with only one colour. The numbers are always given in descending order, which means, for example, that it is possible to have a type 2,1 bubble, but not a type 1,2.

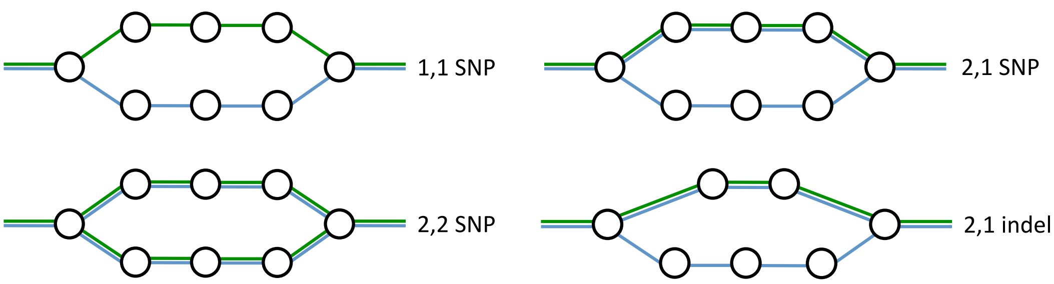

Figure 3: Type classification system for bubbles: Bubble types have one number for each path through the bubble, the number indicating the number of colours represented on the respective path. Thus a 2,1 has two paths, with two colours on the first path and 1 colour on the second.

3 Build and installation

3.1 Cortex

Cortex Con sources can be downloaded from `http://cortexassembler.sourceforge.net/`, where further information on building is also provided. A Makefile is provided which will work on most UNIX, Linux and Mac OS X systems.

As with the original Cortex assembler, a decision needs to be made at compile-time on the largest k-mer size which you wish a given executable to support and this value may be either 31, 63 or 95 nucleotides. Selecting this at compile-time allows for much more efficient use of memory during program execution. Often users will compile three different versions of the code, selecting at run-time the most appropriate one to use.

The maximum kmer size may be specified as a parameter to the make command:

```
make MAXK=31 cortex_bub
```

When the build process completes, the executable may be found in the `bin` directory as `cortex_bub_k` where k is the maximum kmer size chosen.

3.2 Bubbleparse

Bubbleparse sources can be downloaded from `https://github.com/richardmleggett/bubbleparse`. Bubbleparse is built using the Cortex makefile, so the `bubbleparse.c` file needs to be copied into the `src/util` folder of the Cortex build structure. As with Cortex, the largest k-mer size needs to be specified at build-time. Bubbleparse can be built by changing into the Cortex build directory and typing:

```
make MAXK=31 bubbleparse
```

When the build process completes, the executable may be found in the `bin` directory.

4 Using Cortex

For a full description of the Cortex command line options, please refer to the main Cortex manual. In this manual, we concentrate on describing the options that relate to SNP discovery.

Figure 4 illustrates a typical scenario for bubble detection, in which we have reads from resistant and susceptible bulks and we wish to find SNPs or indels between them. In this situation, the most efficient approach is to merge the resistant reads into a single binary file, do the same for the susceptible reads, then use the two merged binary files as the input to the SNP discovery process. The merged files can be created using a command line similar to the following:

```
cortex_bub_31 -k 31 -n 23 -b 65 -i files.txt -t fastq -o output.ctx
```

The options have the following meanings:

- the `-k` option specifies the kmer size to be used for the de Bruijn graph.

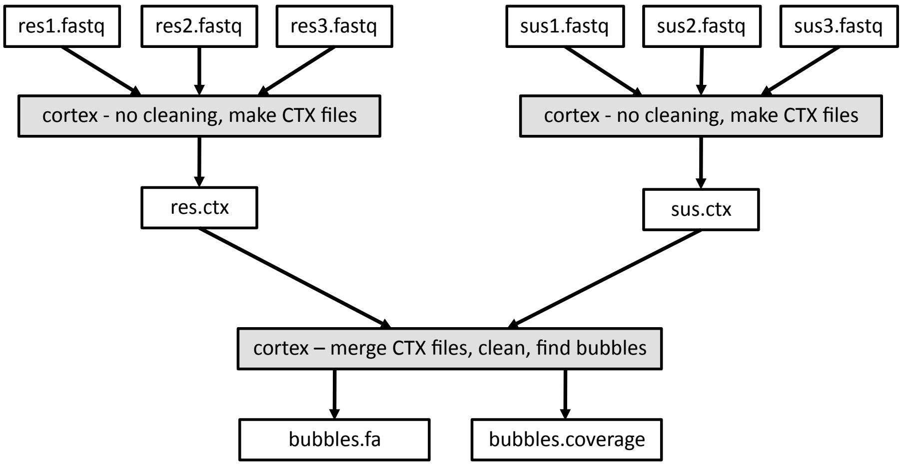

Figure 4: A typical scenario for SNP finding with Cortex: files of reads from resistant and susceptible bulks are merged into two separate binary files. These files then become the input to the cortex bubble detection process, which writes a file of contigs representing the paths through the bubbles.

- the `-n` and `-b` options specify the hash table width and height to be used to store the de Bruijn graph.
- the `-i` option specifies the name of an input file of files. All files listed in this file will be merged to create a single graph. Files must be of the same format, but Cortex will accept FASTA format, FASTQ format, or Cortex's own CTX binary format.
- the `-t` option tells Cortex to expect FASTQ format files.
- the `-o` option specifies the name of the merged binary output file.

Cortex discovers SNPs by identifying nodes where the de Bruijn graph branches and exploring paths from each of these nodes, searching for a point of re-convergence. It adopts a depth-first search algorithm and will explore all available paths up to a user-defined maximum level of bifurcation and a maximum number of nodes in a path.

Using the -w option causes the SNP detection algorithms to be run. This option requires two parameters, separated by commas which specify the maximum depth of search and the maximum number of nodes allowed for the path. Thus, a typical command might look something like the following:

```
cortex_bub_31 -k 31 -n 23 -b 65 -w 2,1000 -i ctxfiles.txt -t binary -f snpout
```

In this example, we have told Cortex to explore all paths with 2 or less bifurcations and with 1000 nodes or less. The other options have the following meanings:

- the `-i` option, as previously, specifies the name of an input file of files. This time, the file of files will contain the names of the CTX files created in the first step.
- the `-t` option tells Cortex that the input files are binary .ctx files.
- the `-f` option specifies the prefix filename of the output files generated by the SNP finding process. In the example above, this would result in a file of SNP contigs called `snpout.fasta` and a coverage file called `snpout.coverage`. See section 5 for details of these files.

As well as the above options, it is likely that we may wish to apply some of Cortex's cleaning options. This should result in a cleaner graph and less likelihood of false positives. Thus, we might also add the following options to the command line given above:

- the `-c` option causes clipping of tips up to a specified length. For example, `-c 100` will clip tips up to 100 nodes long.
- the `-P` option will remove low coverage paths. For example, `-P 1` will remove paths where the coverage of each node is 1 or less.
- the `-z` option will remove low coverage nodes. For example, `-z 1` will remove nodes of coverage 1 or less. Normally, you would not specify `-z` if you were using the `-P` option.

Further information on all options can be found in the Cortex manual.

5 Cortex output files

Once the SNP finding has run, Cortex will generate two files - a FASTA file containing contigs representing the paths through each bubble and a separate text file which contains the coverage of the kmers contributing to the contigs.

5.1 The FASTA file

The FASTA file contains one entry for each path through each bubble. For example:

```
>match_8_path_0 length:69 type:RR pre_length:35 mid_length:31 post_length:
3 c0_average_coverage:25.97 c1_average_coverage:12.48 average_coverage:41.
36 min_coverage:35 max_coverage:89 fst_coverage:89 fst_kmer:ATCAGTGTGTGCGT
TTGGCCCGATTTGAGAA fst_r:G fst_f:CT lst_coverage:83 lst_kmer:AAGCCCCATCAGGA
GGTATGAATGATATAGT lst_r:T lst_f:AT
ATCAGTGTGTGCGTTTGGCCCGATTTGAGAACGATcgaactatatcattcatacctcctgatggggCTT
>match_8_path_1 length:69 type:RR pre_length:35 mid_length:31 post_length:
3 c0_average_coverage: 2.00 c1_average_coverage: 0.00 average_coverage:12.
38 min_coverage:2 max_coverage:89 fst_coverage:89 fst_kmer:ATCAGTGTGTGCGTT
TGGCCCGATTTGAGAA fst_r:G fst_f:CT lst_coverage:83 lst_kmer:AAGCCCCATCAGGAG
GTATGAATGATATAGT lst_r:T lst_f:AT
ATCAGTGTGTGCGTTTGGCCCGATTTGAGAACGATtgaactatatcattcatacctcctgatggggCTT
```

For each path, there are two lines of output. The line beginning `>` is a comment line and contains an identifier for the sequence (for example, `match_8_path_1`), as well as a range of descriptive data about the sequence. The comment line is followed by a single line of nucleotide sequence which represents the path through the bubble (lower case) and flanking (upper case). The data fields in the comment line have the following meanings:

- **length** - the length of the sequence, in nucleotides.
- **type** - the type of bubble discovered. This consists of two characters, the first representing the start node, the second the end node. An R indicates a reverse branch Y node, an F a forward branch Y node and an X indicates an X node.
- **pre_length** - the length of the flanking before the bubble path sequence.

- **mid_length** - the length of the path through the bubble.
- **post_length** - the length of the flanking after the bubble path sequence.
- **cX_average_coverage** - the mean coverage for colour X of path through bubble (ie. not including flanking).
- **average_coverage** - the mean coverage of the whole sequence (including flanking).
- **min_coverage** - the lowest coverage of any node in the sequence.
- **max_coverage** - the highest coverage of any node in the sequence.
- **fst_f** - valid edges in the de Bruijn graph in the forward orientation for the first kmer of the path that generated the sequence.
- **fst_r** - valid edges in the de Bruijn graph in the reverse orientation for the first kmer of the path that generated the sequence.
- **fst_kmer** - the first kmer in the sequence.
- **lst_f** - valid edges in the de Bruijn graph in the forward orientation for the last kmer of the path that generated the sequence.
- **lst_r** - valid edges in the de Bruijn graph in the reverse orientation for the last kmer of the path that generated the sequence.
- **lst_kmer** - the last kmer in the sequence.

5.2 The coverage file

The coverage file also contains an entry for each path through each bubble. For example:

```
>match_8_path_0
53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53
53 53 53 53 53 53 25 26 26 29 27 28 26 25 25 26 26 25 26 25 24 24 23 24 24
24 25 26 26 27 27 27 29 28 29 26 27 27 27 26 26 28 27 58
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
36 36 36 36 36 36 19 16 16 15 14 14 14 14 15 13 13 12 12 16 16 16 16 15 13
11 10 11 11 11 10 10 10 10 10 11 12 12 12 12 11 11 11 25
>match_8_path_1
53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53
53 53 53 53 53 53 25 26 26 29 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 28 27 58
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
36 36 36 36 36 36 19 16 16 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 11 11 25
```

An entry consists of a comment line, followed by a line of space-separated coverage values for each colour. The comment line begins with a > symbol, followed by an identifier which links the coverage data to an entry in the FASTA file. Each line of coverage data is simply a list of coverage values associated with each nucleotide in the sequence.

6 Using Bubbleparse

Bubbleparse takes as inputs the FASTA file and coverage file generated by Cortex, but also requires access to the original read files (FASTA or FASTQ) used as the input to Cortex. Bubbleparse may be run as follows:

```
bubbleparse_31 -f snpout -i files.txt -t table.txt -c table.csv -k 31
               -o options.txt
```

The options have the following meanings:

- `-f` specifies the prefix filename of the output files generated by Cortex.
- `-i` specifies the name of a file of filenames specifying the original read files.
- `-t` specifies the name of a file to write a text ranking table to.
- `-c` specifies the name of a file to write an output CSV ranking table to.
- `-r` specifies the name of a file to write a ranked contig FASTA file to.
- `-k` specifies the kmer size.
- `-o` specifies an options file which contains values for the expected coverage percent, tolerance and minimum contig size.

The options file consists of a list of keywords and values, with one keyword per line and values given in speech marks. A typical example is give below:

```
EXPECTEDCOVERAGE "0,10,50,50"
EXPECTEDCOVERAGE "1,10,100,0"
MINIMUMCONTIGSIZE "70"
```

A file will contain one `MINIMUMCONTIGSIZE` line and one or more `EXPECTEDCOVERAGE` lines. At the time of writing, these are the only valid options, but more may be added in future releases.

The `EXPECTEDCOVERAGE` parameter defines the expected coverage percentages for a given colour. Typically, we would assign one colour to a resistant bulk and another to a susceptible bulk. Knowledge of the organism being examined and the experimental setup enables us to predict how often the resistant alleles will occur compared to the susceptible ones in each bulk. We express this as a percentage and also give a tolerance which determines the acceptable deviation from the ideal percentages, within which SNPs will still be highly rated. The four numbers separated by commas are, in order, the colour number, the tolerance, the lower percentage, the upper percentage. So, in the example above, we expect a 50/50 ratio for the resistant bulk (colour 0) and a 100/0 ratio for the susceptible bulk (colour 1) and in each case allow a 10% tolerance.

The `MINIMUMCONTIGSIZE` parameter provides a length below which SNP contigs will be considered too short to be useful and will therefore be ranked at the bottom of the list.

7 Bubbleparse output files

The text ranking table is output if the `-t` parameter is specified. It is composed of a number of separate ranked tables, one for each SNP type found in the input. Each new table begins with a header line and the ranked numbering of SNPs starts at 1 for each table. An example extract is shown in Figure 5. The columns have the following meanings:

- **Rank** - gives the rank within this type, with 1 the highest rank.
- **Match** - gives the match number, which can be cross referenced with the Cortex output files.
- **Num Pth** - indicates the number of paths through the bubble.
- **Type** - the type of the bubble, according to the classification given in Section 2.2.
- **Lngst Cntig** - gives the length of the longest contig associated with the bubble - that is, the length of the longest path through the bubble, plus flanking. For SNPs, paths through the bubble will all have the same length, but for indels, path lengths are likely to be different.
- **Path Len** - gives the length of the first two (or three, if present) paths through the bubble. This length does not include flanking.
- **Flags** - shows the status of a number of flags associated with the entry.
- **c0 Coverage** - provides the mean coverage in colour 0 along the first (P0) and second (P1) paths through the bubble.
- **c1 Coverage** - provides the mean coverage in colour 1 along the first (P0) and second (P1) paths through the bubble.
- **c0 Coverage %** - provides the colour 0 percentage coverage along the first (P0) and second (P1) paths through the bubble. The sum of colour 0 percentages along all paths will add up to 100.
- **c1 Coverage %** - provides the colour 1 percentage coverage along the first (P0) and second (P1) paths through the bubble. The sum of colour 0 percentages along all paths will add up to 100.
- **Difference** - provides a measure of how much the coverage percentage differs from the expected coverage ratio.

A CSV ranking table is output if the `-c` parameter is specified. This consists of the same information as the text ranking table, but the white space is removed and fields are separated with commas.

8 Problems

Please report any bugs or problems with the bubble finding options in Cortex, or with bubbleparse, to `richard.leggett@tsl.ac.uk`. General problems with Cortex should be referred to the appropriate authors.

```
                 Num             Lngst  Path Len           c0 Coverage     c1 Coverage     c0 Coverage %  c1 Coverage %    Difference
 Rank  Match  Pth     Type     Cntig P0   P1   P2  Flags    P0      P1      P0      P1      P0      P1      P0      P1      c0      c1    QTotal
-------------------------------------------------------------------------------------------------------------------------------------------------
   1    19769   2    S 1,1     217  31   31       --SW--  6.84    7.97    0.06    0.00    46.19  53.81   100.00   0.00    3.81    0.00    663
   2    60706   2    S 1,1     427  31   31       --SW-- 10.48    8.84    0.00    0.13    54.26  45.74     0.00 100.00    4.26    0.00    632
   3       29   2    S 1,1     211  31   31       --SW--  6.68    9.81    0.45    0.00    40.51  59.49   100.00   0.00    9.49    0.00    630
   4    58074   2    S 1,1     426  31   31       --SW--  6.81    6.97    4.32    0.00    49.41  50.59   100.00   0.00    0.59    0.00    600
   5    22418   2    S 1,1     370  31   31       --SW--  8.45    7.97    0.00    0.45    51.47  48.53     0.00 100.00    1.47    0.00    594
   6    35633   2    S 1,1     231  31   31       --SW--  6.74    5.55    0.00    0.87    54.86  45.14     0.00 100.00    4.86    0.00    560
   7    48097   2    S 1,1     228  31   31       --SW--  7.55    7.32    0.00    1.52    50.76  49.24     0.00 100.00    0.76    0.00    551
   8    41942   2    S 1,1     302  31   31       --SW--  8.42   10.94    0.06    2.35    43.50  56.50     2.67  97.33    6.50    2.67    525
   9    59886   2    S 1,1     265  31   31       --SW--  7.00    6.29    0.00    3.81    52.67  47.33     0.00 100.00    2.67    0.00    505
  10    44477   2    S 1,1     235  31   31       --SW--  4.19    6.26    0.00    0.13    40.12  59.88     0.00 100.00    9.88    0.00    495
  11     8147   2    S 1,1     225  31   31       --SW--  6.55    9.61    2.19    0.00    40.52  59.48   100.00   0.00    9.48    0.00    493
  12    30468   2    S 1,1     210  31   31       --SW--  6.13    7.16    0.00    0.00    46.12  53.88     0.00   0.00    3.88    0.00    486
  13    30283   2    S 1,1     278  31   31       --SW--  8.45   11.13    0.00    1.81    43.16  56.84     0.00 100.00    6.84    0.00    480
  14    38514   2    S 1,1     204  31   31       --SW--  3.58    5.10    0.00    3.00    41.26  58.74     0.00 100.00    8.74    0.00    470
  15    51559   2    S 1,1     233  31   31       --SW--  7.26    6.94    0.00    1.94    51.14  48.86     0.00 100.00    1.14    0.00    463
```

```
Rank,Match,NumPth,Type,LngstCntig,LenP0,LenP1,CovC0P0,CovC0P1,CovC1P0,CovC1P1,PcC0P0,PcC0P1,PcC1P0,PcC1P1,C0Dif,C1Dif,QTotal
1,19769,2,S_1_1,217,31,31,6.84,7.97,0.06,0.00,46.19,53.81,100.00,0.00,3.81,0.00,663
2,60706,2,S_1_1,427,31,31,10.48,8.84,0.00,0.13,54.26,45.74,0.00,100.00,4.26,0.00,632
3,29,2,S_1_1,211,31,31,6.68,9.81,0.45,0.00,40.51,59.49,100.00,0.00,9.49,0.00,630
4,58074,2,S_1_1,426,31,31,6.81,6.97,4.32,0.00,49.41,50.59,100.00,0.00,0.59,0.00,600
5,22418,2,S_1_1,370,31,31,8.45,7.97,0.00,0.45,51.47,48.53,0.00,100.00,1.47,0.00,594
6,35633,2,S_1_1,231,31,31,6.74,5.55,0.00,0.87,54.86,45.14,0.00,100.00,4.86,0.00,560
7,48097,2,S_1_1,228,31,31,7.55,7.32,0.00,1.52,50.76,49.24,0.00,100.00,0.76,0.00,551
8,41942,2,S_1_1,302,31,31,8.42,10.94,0.06,2.35,43.50,56.50,2.67,97.33,6.50,2.67,525
9,59886,2,S_1_1,265,31,31,7.00,6.29,0.00,3.81,52.67,47.33,0.00,100.00,2.67,0.00,505
10,44477,2,S_1_1,235,31,31,4.19,6.26,0.00,0.13,40.12,59.88,0.00,100.00,9.88,0.00,495
11,8147,2,S_1_1,225,31,31,6.55,9.61,2.19,0.00,40.52,59.48,100.00,0.00,9.48,0.00,493
12,30468,2,S_1_1,210,31,31,6.13,7.16,0.00,0.00,46.12,53.88,0.00,0.00,3.88,0.00,486
13,30283,2,S_1_1,278,31,31,8.45,11.13,0.00,1.81,43.16,56.84,0.00,100.00,6.84,0.00,480
14,38514,2,S_1_1,204,31,31,3.58,5.10,0.00,3.00,41.26,58.74,0.00,100.00,8.74,0.00,470
15,51559,2,S_1_1,233,31,31,7.26,6.94,0.00,1.94,51.14,48.86,0.00,100.00,1.14,0.00,463
```

Figure 5: Example table output (top) and CSV output (bottom) from Bubbleparse.