TASSEL 3.0 Universal Network Enabled Analysis Kit (UNEAK) pipeline documentation

Authors: Fei Lu, Jeff Glaubitz, James Harriman, Terry Casstevens, Rob Elshire

Please note that this is an unfinished work in progress…

April 1st, 2012

Table of Contents

Introduction

The UNEAK is the non-reference Genotyping by Sequencing (GBS) SNP calling pipeline, which is an extension of the Java program of TASSEL. UNEAK commands are run as TASSEL plugins via the command line in the following format (Linux or Mac operating system; for Windows use `run_pipeline.bat`):

```
run_pipeline.pl -fork1 -PluginName --plugin-option -endPlugin -runfork1
```

Each step of the pipeline is specified with a "fork" command and a number, since TASSEL can run several processes at once, and split and recombine their results. The fork option is followed by the name of the plugin, and any plugin-specific options. If no plugin options are provided, the program will print a list of available options. -endPlugin signals the end of plugin-specific options, and -runfork1 then runs the specified plugin. In all of our examples here for the UNEAK pipeline, we run only a single fork at a time.

Please see http://www.maizegenetics.net/tassel/docs/TasselPipelineCLI.pdf for general instructions on how to install the TASSEL 3.0 Standalone Build on your computer. These UNEAK-specific instructions assume that you have unzipped the standalone into the directory (folder)

`/programs`

and then renamed the directory

`/programs/tassel3.0_standalone`

to

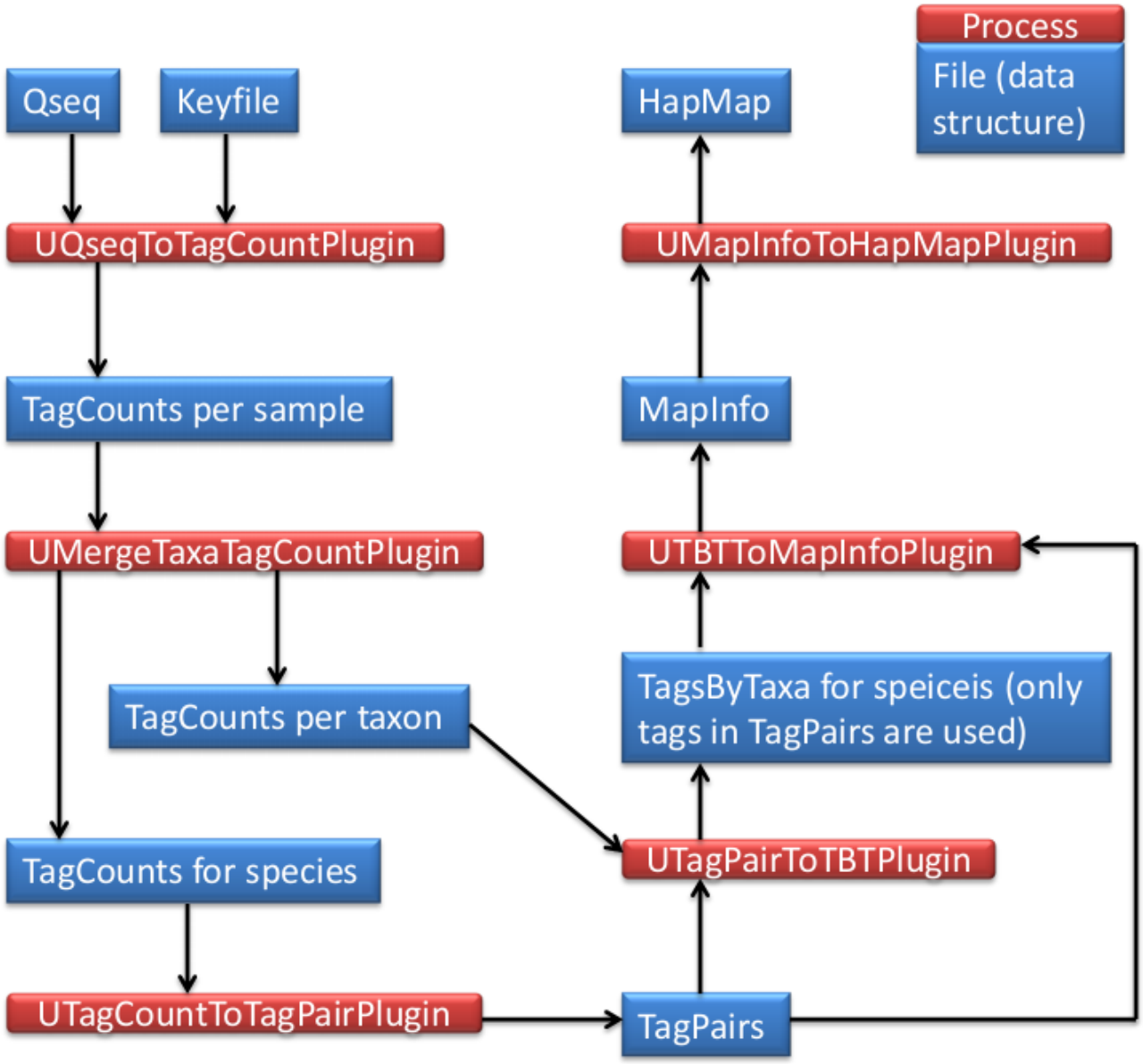
`/programs/tassel`

If not, you will have to edit the example commands appropriately (*e.g.*, replace "`tassel`" with "`tassel3.0_standalone`").

If you have more memory available on your machine than 1.5GB, then you can increase the amount of memory available to TASSEL by opening `run_pipeline.pl` (or `run_pipeline.bat`) in a text editor and modifying "`-Xms512m -Xmx1536m`" to (for example) "`-Xms4g`" (the `-Xms` option controls the amount of

memory allocated to the program on startup).

The flow chart below shows how the steps of the analysis link together. Blue boxes represent files produced at each step of the analysis, and red boxes represent the processes that produced them.



Details and application of UNEAK can be seen at http://www.maizegenetics.net/gbs-bioinformatics.

UCreatWorkingDirPlugin

Summary:
Creat subdirectories in the working directory (folder) for your analysis using this plugin. A dot (.) represents the working directory from your input (e.g., `M:/UNEAK/`). The following subdirectories will be created:

```
./Illumina/          (original raw data files, one file per flowcell lane)
./key/               (Barcode key file of original raw data files)
./tagCounts/         (for output from UQseqToTagCountPlugin OR
                      UFastqToTagCountPlugin OR UMergeTaxaTagCountPlugin)
./mergedTagCounts/   (for output from UMergeTaxaTagCountPlugin)
./tagPair/           (for output from UTagCountToTagPairPlugin)
./tagsByTaxa/        (for output from UTBTToMapInfoPlugin)
./mapInfo/    (for output from UTBTToMapInfoPlugin)
./hapMap/            (for output from TagsToSNPByAlignmentPlugin)
```

After these subdirectories are ready, then you need to move or link the raw sequence data files (Qseq or Fastq) and a barcode key file into the subdirectories ./Illumina/ and ./key/, respectively. Multiple raw data files are allowed in the subdirectory ./Illumina/. But there is only one key file in the subdirectory ./key/.

Input:
- None

Output:
- None

Arguments:

UCreatWorkingDirPlugin	
-w	Working directory to contain subdirectories

Example command:
```
/programs/tassel/run_pipeline.pl -fork1 -UCreatWorkingDirPlugin -w M:/UNEAK/
-endPlugin -runfork1
```

UQseqToTagCountPlugin

Summary:
This plugin derives a tagCount list for each sample in the subdirectory ./tagCounts/. It keeps only good reads having a barcode and a cut site and no N's in the useful part of the sequence. Trims off the barcodes and truncates sequences that (1) have a second cut site, or (2) read into the common adapter. **If your input files are in fastq format (and qseq files are not available), use UFastqToTagCountPlugin instead (same arguments)**.

Input:
- Barcode key file from the subdirectory ./key/ (see example in Appendix 1)
- Qseq files from the subdirectory ./Illunima/

Output:
- tagCount (*.cnt) file for every sample in the subdirectory ./tagCounts/

Arguments:

UQseqToTagCountPlugin	
-w	Working directory to contain subdirectories
-e	Enzyme used to create the GBS library

Example command:

```
/programs/tassel/run_pipeline.pl -fork1 -UQseqToTagCountPlugin -w M:/UNEAK/
-e ApeKI -endPlugin -runfork1
```

Gory Details:

This step reads a user-supplied key file (in subdirectory ./key/) in tab-delimited text format which indicates, for each lane of interest from a flowcell, which barcodes are assigned to which sample (a short example key file is provided in Appendix 1). It then recursively searches the subdirectory ./Illumina/ for qseq files matching one of the flowcell/lane combinations in the key file and with the following acceptable file naming conventions:

FLOWCELL_LANE_qseq.txt (example: **42A87AAXX_2_qseq.txt**)
FLOWCELL_LANE_qseq.txt.gz (example: **42A87AAXX_2_qseq.txt.gz**)
code_FLOWCELL_s_LANE_qseq.txt (example: **10225395_42A87AAXX_s_2_qseq.txt**)
code_FLOWCELL_s_LANE_qseq.txt.gz (example: **10225395_42A87AAXX_s_2_qseq.txt.gz**)

Note that both compressed (*.gz) and uncompressed (*.txt) files can be read. We recommend using compressed files to save disk storage space. The "code" part of the latter two file name examples is a numerical tracking code generated by our sequencing center. UNEAK pipeline doesn't actually use this code, so you can substitute any text or numbers (or use one of the first two conventions). The underscores are essential for correct parsing of the parts of each qseq file name (only FLOWCELL and LANE are actually used by our pipeline).

For each qseq file that has a match in the key file, UQseqToTagCountPlugin finds all reads that begin with one of the expected barcodes immediately followed by the expected cut site remnant (CAGC or CTGC for *ApeK*I) and trims them to 64 bases (including the cut site remnant but removing the barcode). Reads containing N within the first 64 bases after the barcode are rejected. If a read contains either a full cut site (from incomplete digestion or chimera formation) or the beginning of the common adapter (from restriction fragments less than 64bp) within the first 64 bases it is truncated appropriately and padded to 64 bases with polyA. The actual length of truncated (or full 64 base) reads is recorded in the output tagCount file.

The output of UQseqToTagCountPlugin is multiple tagCount files for all the samples in qseq files. The output is in the subdirectory ./tagCounts/. The tagCount files are named after their corresponding sample name, qseq file, lane number and well ID in plates by *.cnt, for example, U318_622WNAAXX_1_D3.cnt. The tagCount files are binary, and can only be read by our pipeline. They contain the 64 base sequence of each good, barcoded tag (padded with polyA if truncated), the actual length of the tag (before padding with polyA), and the number of times that tag was observed in the corresponding sample. The tags are sorted by their sequence.

The enzyme used to create the GBS library is indicated via mandatory option **-e**. Currently, our pipeline accepts ApeKI, PstI, PasI, HpaII, PstI-MspI, PstI-TaqI, PstI-EcoT22I and SbfI-MspI.

We recommend using qseq files if you have them because they contain all reads, not just the ones passing Illumina's quality filters. We have found that perfectly good reads (exactly matching a 64 base tag that we have seen many times) can be filtered out by Illumina. **If qseq files are not available, or your raw data are in Illumina's latest FASTQ format (from Casava 1.8), use FastqToTagCountPlugin instead (same arguments as QseqToTagCountPlugin).**

UMergeTaxaTagCountPlugin

Summary:

(1) Merge tagCount files of the same taxon in the subdirectory ./tagCounts/. (2) Merges each tagCount file in the subdirectory ./tagCounts/ into a single "master" tagCount file (./mergedTagCounts/mergedAll.cnt). Only keeps tags with a total count (after merger) greater than or equal to that specified in option **-c** (*minimum number of times a tag must be present to be output*).

Input:

- tagCount (*.cnt) file for every sample in the subdirectory ./tagCounts/

Output:

- Merged tagCount file of the same taxon (./tagCounts/XXXXXX_merged.cnt)
- Merged tagCount file of all taxa (./mergedTagCounts/mergedAll.cnt)

Arguments:

UMergeTaxaTagCountPlugin	
-w	Working directory to contain subdirectories
-c	Minimum count of a tag must be present to be output. Default: 5

Example command:

```
/programs/tassel/run_pipeline.pl -fork1 -UMergeTaxaTagCountPlugin
-w M:/UNEAK/ -c 5 -endPlugin -runfork1
```

Gory Details:

The UMergeTaxaTagCountPlugin step merges multiple tagCount files of the same taxon, for example, U518_ 622WNAAXX_1_D3.cnt, U518_622WNAAXX_1_C11.cnt, etc would be merged into U518_merged.cnt, which would be in the subdirectory ./tagCounts/.
Also, this plugin merges all the tagCount files in the subdirectory ./tagCounts/ into a single "master" tagCount file, which is ./mergedTagCounts/mergedAll.cnt. (For a description of the tagCount file format, see UQseqToTagCountPlugin.)

To remove rare or singleton tags that possibly result from sequencing errors, we use the **-c** option (*minimum number of times a tag must be present to be output*). A **-c** option setting of 5 or 10 is typical, but when deciding on an appropriate cutoff, you should consider the number of individuals in your analysis, the expected coverage (about 0.4-0.5x for maize with *Ape*KI), the expected segregation ratio, minimum minor allele frequency of interest, etc. The merged tagCount output file is used as a master tag list for two subsequent steps: the UTagCountToTagPairPlugin step. The output is in (binary) tagCount format by default, which serves as the input format for the UTagCountToTagPairPlugin step.

UTagCountToTagPairPlugin

Summary:

Identify tag pairs for SNP calling via the network filter.

Input:

- Merged tagCount file of all taxa (./mergedTagCounts/mergedAll.cnt)

Output:

- tagPair file (./tagPair/tagPair.tps)

Arguments:

UTagCountToTagPairPlugin	
-w	Working directory to contain subdirectories
-e	Error tolerance rate in the network filter. Default: 0.03

Example command:

```
/programs/tassel/run_pipeline.pl -fork1 -UTagCountToTagPairPlugin
-w M:/UNEAK/ -e 0.03 -endPlugin -runfork1
```

Gory Details:

The UTagCountToTagPairPlugin step implements the pairwise alignment. Tag pairs with 1 bp mismatch are

considered as candidate SNPs. One tag is usually involved in multiple tag pairs. Here, the network filter is used to identify reciprocal tag pairs (For details of the network filter, please see http://www.maizegenetics.net/gbs-bioinformatics). The reciprocal tags pairs are called SNPs.

The –e option, which is the Error tolerance rate (ETR), is an important argument. Higher ETR generates more SNPs (Especially those of high coverage SNPS), also more false SNP calls. When ETR equals to 0, it means only purely reciprocal tags are called and no sequencing error happened to these tags. This is the most stringent criteria, but unrealistic, which would largely reduced the number of SNPs, especially when the coverage is high. The default of ETR is 0.03. Based on the observation on Illumina sequencing error rate, the ETR should not be greater than 0.05.

The tagPair file is binary, and can only be read by our pipeline. It contains the 64 base sequence of tag, the actual length of the tag (before padding with polyA), and the order which makes the tags paired. The tagPair file can be sort by sequence and the order both.

UTagPairToTBTPlugin

Summary:
Generates a TagsByTaxa file for the tags in the tagPair file.

Input:
- tagPair file (./tagPair/tagPair.tps)
- tagCount (*.cnt) file for each taxon in the subdirectory ./tagCounts/

Output:
- tagsByTaxa file (./tagsByTaxa/tbt.bin)

Arguments:

UTagPairToTBTPlugin	
-w	Working directory to contain subdirectories

Example command:
```
/programs/tassel/run_pipeline.pl -fork1 -UTagPairToTBTPlugin
-w M:/UNEAK/ -endPlugin -runfork1
```

Gory Details:
The UTagPairToTBTPlugin step figures out the tag distribution in all of the taxa. Note the tags here are only ones in the tagPair file (./tagPair/tagPair.tps), not all the good tags. The tagPair file is sorted by sequence then searched in tagCount files (./tagCounts/*.cnt) of all taxa.

The tagsByTaxa file is in binary format (only readable by our pipeline), but can be thought of as a grid where the rows are the tags of interest, the columns are taxa names. Because only tagsByTaxaByte is supported by UNEAK for now, cells have a maximum value of 127 per taxon per tag. Storing the number of tags per taxon makes it possible to determine whether reads occur more frequently than expected due to chance. The actual length in bases of each tag (not including the polyA padding) is also recorded.

UTBTToMapInfoPlugin

Summary:
Generates a mapInfo file for HapMap output.

Input:
- tagPair file (./tagPair/tagPair.tps)
- tagsByTaxa file (./tagsByTaxa/tbt.bin)

Output:

- mapInfo file (./mapInfo/mapInfo.bin)

Arguments:

UTBTToMapInfoPlugin	
-w	Working directory to contain subdirectories

Example command:

```
/programs/tassel/run_pipeline.pl -fork1 -UTBTToMapInfoPlugin
-w M:/UNEAK/ -endPlugin -runfork1
```

Gory Details:

The UTBTToMapInfoPlugin sorts the tagsByTaxa file according to the order of tags recorded in the tagPair file. Then it converts each tag pair to a HapMap record and assign genotypes to each taxa.

The mapInfo file is in binary format (only readable by our pipeline), which holds information of tag, tag distribution in each taxa, SNPs and code of heterozygous loci.

UMapInfoToHapMapPlugin

Summary:

Output the HapMap file.

Input:

- mapInfo file (./mapInfo/mapInfo.bin)

Output:

- HapMap file (./hapMap/ HapMap.hmp.txt)
- HapMapCount file (./hapMap/ HapMap.hmc.txt)
- HapMap Fasta file (./hapMap/ HapMap.fas.txt)

Arguments:

UMapInfoToHapMapPlugin	
-w	Working directory to contain subdirectories
-mnMAF	Mimimum minor allele frequency. Default: 0.05
-mxMAF	Maximum minor allele frequency. Default: 0.5
-mnC	Minimum call rate
-mxC	Maximum call rate. Default: 1

Example command:

```
/programs/tassel/run_pipeline.pl -fork1 -UMapInfoToHapMapPlugin
-w M:/UNEAK/ -mnMAF 0.05 -mxMAF 0.5 -mnC 0 -mxC 1 -endPlugin -runfork1
```

Gory Details:

The UMapInfoToHapMapPlugin provide options to output the HapMap file. The –mnMAF and –mxMAF set the cutoff for minimum and maximum allele frequency in the HapMap file. The –mnC and –mxC set the cutoff for call rate in the HapMap file. The call rate denotes a proportion that how many taxa are covered by at least one tag. Note there is no order for these SNPs.

The HapMap genotype files that we generate save disk space and memory by using single letters to represent phase unknown, diploid genotypes. Heterozygotes are represented by IUPAC nucleotide codes:

```
A = A/A
C = C/C
G = G/G
T = T/T
M = A/C
R = A/G
W = A/T
S = C/G
Y = C/T
K = G/T
N = missing data
```

In addition to the HapMap file, there are two other files output in the subdirectory ./hapMap/. The first is HapMapCount file (./hapMap/ HapMap.hmc.txt) which records the tag counts of the SNPs in each taxon. This file can be used for more statistical tests. The other is HapMap Fasta file (./hapMap/ HapMap.fas.txt) which record the sequence of the SNP tags. This file can be used for alignment of these SNPs.

Appendix 1: Key file example

The barcode key file is formatted as tab-delimited text. You can create it from Excel if you save it as tab-delimited text. In the example key below there are two lanes, each at 96 plex. The barcodes correspond to our 96-plex *Ape*KI layout. You can combine lanes from multiple flow cells in a single key file and GBS analysis if you wish. Note that there is a "Blank" in each plate, in different positions (H12 and H11). This facilitates diagnosis of accidental plate swaps. Only the first 7 columns are mandatory. You can add additional columns to the key file as you see fit - these will be ignored by the pipeline. **The sample names must not contain spaces, colons (':') or underscores**. However, it is OK to include dashes, or parentheses.

Flowcell	Lane	Barcode	Sample	PlateName	Row	Column
ABC12AAXX	1	CTCC	MySample001	MyPlate1	A	1
ABC12AAXX	1	TGCA	MySample002	MyPlate1	A	2
ABC12AAXX	1	ACTA	MySample003	MyPlate1	A	3
ABC12AAXX	1	CAGA	MySample004	MyPlate1	A	4
ABC12AAXX	1	AACT	MySample005	MyPlate1	A	5
ABC12AAXX	1	GCGT	MySample006	MyPlate1	A	6
ABC12AAXX	1	TGCGA	MySample007	MyPlate1	A	7
ABC12AAXX	1	CGAT	MySample008	MyPlate1	A	8
ABC12AAXX	1	CGCTT	MySample009	MyPlate1	A	9
ABC12AAXX	1	TCACC	MySample010	MyPlate1	A	10
ABC12AAXX	1	CTAGC	MySample011	MyPlate1	A	11
ABC12AAXX	1	ACAAA	MySample012	MyPlate1	A	12
ABC12AAXX	1	TTCTC	MySample013	MyPlate1	B	1
ABC12AAXX	1	AGCCC	MySample014	MyPlate1	B	2
ABC12AAXX	1	GTATT	MySample015	MyPlate1	B	3
ABC12AAXX	1	CTGTA	MySample016	MyPlate1	B	4
ABC12AAXX	1	ACCGT	MySample017	MyPlate1	B	5
ABC12AAXX	1	GTAA	MySample018	MyPlate1	B	6
ABC12AAXX	1	GGTTGT	MySample019	MyPlate1	B	7
ABC12AAXX	1	CCAGCT	MySample020	MyPlate1	B	8
ABC12AAXX	1	TTCAGA	MySample021	MyPlate1	B	9
ABC12AAXX	1	TAGGAA	MySample022	MyPlate1	B	10
ABC12AAXX	1	GCTCTA	MySample023	MyPlate1	B	11
ABC12AAXX	1	CCACAA	MySample024	MyPlate1	B	12
ABC12AAXX	1	GCTTA	MySample025	MyPlate1	C	1
ABC12AAXX	1	CTTCCA	MySample026	MyPlate1	C	2
ABC12AAXX	1	GAGATA	MySample027	MyPlate1	C	3
ABC12AAXX	1	ATGCCT	MySample028	MyPlate1	C	4
ABC12AAXX	1	TATTTTT	MySample029	MyPlate1	C	5
ABC12AAXX	1	CTTGCTT	MySample030	MyPlate1	C	6
ABC12AAXX	1	ATGAAAC	MySample031	MyPlate1	C	7
ABC12AAXX	1	AAAAGTT	MySample032	MyPlate1	C	8
ABC12AAXX	1	GAATTCA	MySample033	MyPlate1	C	9
ABC12AAXX	1	GAACTTC	MySample034	MyPlate1	C	10
ABC12AAXX	1	GGACCTA	MySample035	MyPlate1	C	11
ABC12AAXX	1	GTCGATT	MySample036	MyPlate1	C	12
ABC12AAXX	1	AACGCCT	MySample037	MyPlate1	D	1
ABC12AAXX	1	AATATGC	MySample038	MyPlate1	D	2
ABC12AAXX	1	ACGACTAC	MySample039	MyPlate1	D	3
ABC12AAXX	1	GGTGT	MySample040	MyPlate1	D	4
ABC12AAXX	1	TAGCATGC	MySample041	MyPlate1	D	5
ABC12AAXX	1	AGTGGA	MySample042	MyPlate1	D	6
ABC12AAXX	1	TAGGCCAT	MySample043	MyPlate1	D	7
ABC12AAXX	1	TGCAAGGA	MySample044	MyPlate1	D	8
ABC12AAXX	1	TGGTACGT	MySample045	MyPlate1	D	9
ABC12AAXX	1	TCTCAGTC	MySample046	MyPlate1	D	10
ABC12AAXX	1	CCGGATAT	MySample047	MyPlate1	D	11
ABC12AAXX	1	CGCCTTAT	MySample048	MyPlate1	D	12
ABC12AAXX	1	AGGC	MySample049	MyPlate1	E	1
ABC12AAXX	1	GATC	MySample050	MyPlate1	E	2
ABC12AAXX	1	TCAC	MySample051	MyPlate1	E	3
ABC12AAXX	1	AGGAT	MySample052	MyPlate1	E	4
ABC12AAXX	1	ATTGA	MySample053	MyPlate1	E	5
ABC12AAXX	1	CATCT	MySample054	MyPlate1	E	6
ABC12AAXX	1	CCTAC	MySample055	MyPlate1	E	7

Flowcell	Lane	Barcode	Sample	PlateName	Row	Column
ABC12AAXX	1	GAGGA	MySample056	MyPlate1	E	8
ABC12AAXX	1	GGAAC	MySample057	MyPlate1	E	9
ABC12AAXX	1	GTCAA	MySample058	MyPlate1	E	10
ABC12AAXX	1	TAATA	MySample059	MyPlate1	E	11
ABC12AAXX	1	TACAT	MySample060	MyPlate1	E	12
ABC12AAXX	1	TCGTT	MySample061	MyPlate1	F	1
ABC12AAXX	1	ACCTAA	MySample062	MyPlate1	F	2
ABC12AAXX	1	ATATGT	MySample063	MyPlate1	F	3
ABC12AAXX	1	ATCGTA	MySample064	MyPlate1	F	4
ABC12AAXX	1	CATCGT	MySample065	MyPlate1	F	5
ABC12AAXX	1	CGCGGT	MySample066	MyPlate1	F	6
ABC12AAXX	1	CTATTA	MySample067	MyPlate1	F	7
ABC12AAXX	1	GCCAGT	MySample068	MyPlate1	F	8
ABC12AAXX	1	GGAAGA	MySample069	MyPlate1	F	9
ABC12AAXX	1	GTACTT	MySample070	MyPlate1	F	10
ABC12AAXX	1	GTTGAA	MySample071	MyPlate1	F	11
ABC12AAXX	1	TAACGA	MySample072	MyPlate1	F	12
ABC12AAXX	1	TGGCTA	MySample073	MyPlate1	G	1
ABC12AAXX	1	ACGTGTT	MySample074	MyPlate1	G	2
ABC12AAXX	1	ATTAATT	MySample075	MyPlate1	G	3
ABC12AAXX	1	ATTGGAT	MySample076	MyPlate1	G	4
ABC12AAXX	1	CATAAGT	MySample077	MyPlate1	G	5
ABC12AAXX	1	CGCTGAT	MySample078	MyPlate1	G	6
ABC12AAXX	1	CGGTAGA	MySample079	MyPlate1	G	7
ABC12AAXX	1	CTACGGA	MySample080	MyPlate1	G	8
ABC12AAXX	1	GCGGAAT	MySample081	MyPlate1	G	9
ABC12AAXX	1	TAGCGGA	MySample082	MyPlate1	G	10
ABC12AAXX	1	TCGAAGA	MySample083	MyPlate1	G	11
ABC12AAXX	1	TCTGTGA	MySample084	MyPlate1	G	12
ABC12AAXX	1	TGCTGGA	MySample085	MyPlate1	H	1
ABC12AAXX	1	AACCGAGA	MySample086	MyPlate1	H	2
ABC12AAXX	1	ACAGGGAA	MySample087	MyPlate1	H	3
ABC12AAXX	1	ACGTGGTA	MySample088	MyPlate1	H	4
ABC12AAXX	1	CCATGGGT	MySample089	MyPlate1	H	5
ABC12AAXX	1	CGCGGAGA	MySample090	MyPlate1	H	6
ABC12AAXX	1	CGTGTGGT	MySample091	MyPlate1	H	7
ABC12AAXX	1	GCTGTGGA	MySample092	MyPlate1	H	8
ABC12AAXX	1	GGATTGGT	MySample093	MyPlate1	H	9
ABC12AAXX	1	GTGAGGGT	MySample094	MyPlate1	H	10
ABC12AAXX	1	TATCGGGA	MySample095	MyPlate1	H	11
ABC12AAXX	1	TTCCTGGA	Blank	MyPlate1	H	12
ABC12AAXX	2	CTCC	MySample096	MyPlate2	A	1
ABC12AAXX	2	TGCA	MySample097	MyPlate2	A	2
ABC12AAXX	2	ACTA	MySample098	MyPlate2	A	3
ABC12AAXX	2	CAGA	MySample099	MyPlate2	A	4
ABC12AAXX	2	AACT	MySample100	MyPlate2	A	5
ABC12AAXX	2	GCGT	MySample101	MyPlate2	A	6
ABC12AAXX	2	TGCGA	MySample102	MyPlate2	A	7
ABC12AAXX	2	CGAT	MySample103	MyPlate2	A	8
ABC12AAXX	2	CGCTT	MySample104	MyPlate2	A	9
ABC12AAXX	2	TCACC	MySample105	MyPlate2	A	10
ABC12AAXX	2	CTAGC	MySample106	MyPlate2	A	11
ABC12AAXX	2	ACAAA	MySample107	MyPlate2	A	12
ABC12AAXX	2	TTCTC	MySample108	MyPlate2	B	1
ABC12AAXX	2	AGCCC	MySample109	MyPlate2	B	2

Flowcell	Lane	Barcode	Sample	PlateName	Row	Column
ABC12AAXX	2	GTATT	MySample110	MyPlate2	B	3
ABC12AAXX	2	CTGTA	MySample111	MyPlate2	B	4
ABC12AAXX	2	ACCGT	MySample112	MyPlate2	B	5
ABC12AAXX	2	GTAA	MySample113	MyPlate2	B	6
ABC12AAXX	2	GGTTGT	MySample114	MyPlate2	B	7
ABC12AAXX	2	CCAGCT	MySample115	MyPlate2	B	8
ABC12AAXX	2	TTCAGA	MySample116	MyPlate2	B	9
ABC12AAXX	2	TAGGAA	MySample117	MyPlate2	B	10
ABC12AAXX	2	GCTCTA	MySample118	MyPlate2	B	11
ABC12AAXX	2	CCACAA	MySample119	MyPlate2	B	12
ABC12AAXX	2	GCTTA	MySample120	MyPlate2	C	1
ABC12AAXX	2	CTTCCA	MySample121	MyPlate2	C	2
ABC12AAXX	2	GAGATA	MySample122	MyPlate2	C	3
ABC12AAXX	2	ATGCCT	MySample123	MyPlate2	C	4
ABC12AAXX	2	TATTTTT	MySample124	MyPlate2	C	5
ABC12AAXX	2	CTTGCTT	MySample125	MyPlate2	C	6
ABC12AAXX	2	ATGAAAC	MySample126	MyPlate2	C	7
ABC12AAXX	2	AAAAGTT	MySample127	MyPlate2	C	8
ABC12AAXX	2	GAATTCA	MySample128	MyPlate2	C	9
ABC12AAXX	2	GAACTTC	MySample129	MyPlate2	C	10
ABC12AAXX	2	GGACCTA	MySample130	MyPlate2	C	11
ABC12AAXX	2	GTCGATT	MySample131	MyPlate2	C	12
ABC12AAXX	2	AACGCCT	MySample132	MyPlate2	D	1
ABC12AAXX	2	AATATGC	MySample133	MyPlate2	D	2
ABC12AAXX	2	ACGACTAC	MySample134	MyPlate2	D	3
ABC12AAXX	2	GGTGT	MySample135	MyPlate2	D	4
ABC12AAXX	2	TAGCATGC	MySample136	MyPlate2	D	5
ABC12AAXX	2	AGTGGA	MySample137	MyPlate2	D	6
ABC12AAXX	2	TAGGCCAT	MySample138	MyPlate2	D	7
ABC12AAXX	2	TGCAAGGA	MySample139	MyPlate2	D	8
ABC12AAXX	2	TGGTACGT	MySample140	MyPlate2	D	9
ABC12AAXX	2	TCTCAGTC	MySample141	MyPlate2	D	10
ABC12AAXX	2	CCGGATAT	MySample142	MyPlate2	D	11
ABC12AAXX	2	CGCCTTAT	MySample143	MyPlate2	D	12
ABC12AAXX	2	AGGC	MySample144	MyPlate2	E	1
ABC12AAXX	2	GATC	MySample145	MyPlate2	E	2
ABC12AAXX	2	TCAC	MySample146	MyPlate2	E	3
ABC12AAXX	2	AGGAT	MySample147	MyPlate2	E	4
ABC12AAXX	2	ATTGA	MySample148	MyPlate2	E	5
ABC12AAXX	2	CATCT	MySample149	MyPlate2	E	6
ABC12AAXX	2	CCTAC	MySample150	MyPlate2	E	7
ABC12AAXX	2	GAGGA	MySample151	MyPlate2	E	8
ABC12AAXX	2	GGAAC	MySample152	MyPlate2	E	9
ABC12AAXX	2	GTCAA	MySample153	MyPlate2	E	10
ABC12AAXX	2	TAATA	MySample154	MyPlate2	E	11
ABC12AAXX	2	TACAT	MySample155	MyPlate2	E	12
ABC12AAXX	2	TCGTT	MySample156	MyPlate2	F	1
ABC12AAXX	2	ACCTAA	MySample157	MyPlate2	F	2
ABC12AAXX	2	ATATGT	MySample158	MyPlate2	F	3
ABC12AAXX	2	ATCGTA	MySample159	MyPlate2	F	4
ABC12AAXX	2	CATCGT	MySample160	MyPlate2	F	5
ABC12AAXX	2	CGCGGT	MySample161	MyPlate2	F	6
ABC12AAXX	2	CTATTA	MySample162	MyPlate2	F	7
ABC12AAXX	2	GCCAGT	MySample163	MyPlate2	F	8
ABC12AAXX	2	GGAAGA	MySample164	MyPlate2	F	9

Flowcell	Lane	Barcode	Sample	PlateName	Row	Column
ABC12AAXX	2	GTACTT	MySample165	MyPlate2	F	10
ABC12AAXX	2	GTTGAA	MySample166	MyPlate2	F	11
ABC12AAXX	2	TAACGA	MySample167	MyPlate2	F	12
ABC12AAXX	2	TGGCTA	MySample168	MyPlate2	G	1
ABC12AAXX	2	ACGTGTT	MySample169	MyPlate2	G	2
ABC12AAXX	2	ATTAATT	MySample170	MyPlate2	G	3
ABC12AAXX	2	ATTGGAT	MySample171	MyPlate2	G	4
ABC12AAXX	2	CATAAGT	MySample172	MyPlate2	G	5
ABC12AAXX	2	CGCTGAT	MySample173	MyPlate2	G	6
ABC12AAXX	2	CGGTAGA	MySample174	MyPlate2	G	7
ABC12AAXX	2	CTACGGA	MySample175	MyPlate2	G	8
ABC12AAXX	2	GCGGAAT	MySample176	MyPlate2	G	9
ABC12AAXX	2	TAGCGGA	MySample177	MyPlate2	G	10
ABC12AAXX	2	TCGAAGA	MySample178	MyPlate2	G	11
ABC12AAXX	2	TCTGTGA	MySample179	MyPlate2	G	12
ABC12AAXX	2	TGCTGGA	MySample180	MyPlate2	H	1
ABC12AAXX	2	AACCGAGA	MySample181	MyPlate2	H	2
ABC12AAXX	2	ACAGGGAA	MySample182	MyPlate2	H	3
ABC12AAXX	2	ACGTGGTA	MySample183	MyPlate2	H	4
ABC12AAXX	2	CCATGGGT	MySample184	MyPlate2	H	5
ABC12AAXX	2	CGCGGAGA	MySample185	MyPlate2	H	6
ABC12AAXX	2	CGTGTGGT	MySample186	MyPlate2	H	7
ABC12AAXX	2	GCTGTGGA	MySample187	MyPlate2	H	8
ABC12AAXX	2	GGATTGGT	MySample188	MyPlate2	H	9
ABC12AAXX	2	GTGAGGGT	MySample189	MyPlate2	H	10
ABC12AAXX	2	TATCGGGA	Blank	MyPlate2	H	11
ABC12AAXX	2	TTCCTGGA	MySample190	MyPlate2	H	12