# Chapter 2

# OligoTag: A Program for Designing Sets of Tags for Next-Generation Sequencing of Multiplexed Samples

**Eric Coissac**

## Abstract

Next-generation sequencing systems allow high-throughput production of DNA sequence data. But this technology is more adapted for analyzing a small number of samples needing a huge amount of sequences rather than a large number of samples needing a small number of sequences. One solution to this problem is sample multiplexing. To achieve this, one can add a small tag at the extremities of the sequenced DNA molecules. These tags will be identified using bioinformatics tools after the sequencing step to sort sequences among samples. The rules to apply for selecting a good set of tags adapted to each situation are described in this chapter. Depending on the number of samples to tag and on the required quality of assignation, different solutions are possible. The software oligoTag, a part of OBITools that computes these sets of tags, is presented with some example sets of tags.

**Key words:** Next-generation sequencing, Multiplexing, Sample, Tags

## 1. Introduction

High-throughput sequencers allow for easily and quickly generating a huge number of sequences. Currently, two systems are mainly used: the 454 GS FLX from Roche®; and the Solexa system from Illumina®; . In their current versions, the 454 GS FLX produces 1 million sequence reads per run and the Solexa machine produces 1 billion reads. The characteristics of these machines and of the sequences produced make these two technologies more complementary than concurrent. Many molecular ecological studies can take advantage of these new systems allowing the elaboration of large-scale experimental protocols. Many useful techniques for these studies rely on PCR (Polymerase Chain Reaction) amplicon sequencing. Before high-throughput sequencing technologies, these PCR-based techniques required a cloning step for building a

François Pompanon and Aurélie Bonin (eds.), *Data Production and Analysis in Population Genomics: Methods and Protocols*, Methods in Molecular Biology, vol. 888, DOI 10.1007/978-1-61779-870-2_2, © Springer Science+Business Media New York 2012

DNA library where each clone contained one DNA molecule synthesized during the PCR amplification. A subset of clones from the library could then be sequenced to estimate the global diversity included in the PCR amplicon. Such an approach limited the number of sequences that could be produced. Actually, in most of the studies, a few thousands of clones were typically analyzed (e.g., (1–5)). Now the new sequencing techniques allow direct sequencing of individual DNA molecules composing a PCR amplicon without any cloning step. This has many advantages, including protocol simplification, as cloning PCR amplicons is not so easy even with commercialized kits. The cloning was also a potential source of bias for library representativity. But the main advantage is certainly that these new machines allow parallel sequencing of a very large number of individual sequences. It is thus possible to reach a higher sequence depth leading to a better coverage of the sample (e.g., (6)). Depending on the machine used, one can physically divide one run into up to eight or sixteen areas, each of them receiving a sample. Per sequencing area we might obtain 75,000 and 50,000,000 sequence reads for a 454 GS FLS or a Solexa system, respectively. Such high coverages are excessive for specific experiments where a higher sample number and a smaller sequencing depth are required. The strategy used to reach this aim is then to mix several samples in one sequencing area. For allowing individual analysis of each sample, a short sample-specific oligonucleotide (i.e., a tag) is added at the extremity of each molecule of each sample before sequencing (e.g., (7)). This tag will allow sorting sequences corresponding to each sample after the sequencing step using appropriate bioinformatic tools.

## 2. Theoretical Background

### *2.1. The Simplest System*

We can imagine several strategies for adding such tags at the end of the DNA molecules constituting a sample. They can rely on a ligation step adding some adaptors including the sequence tag or the tags can be directly added to the PCR primers during their synthesis. This last strategy requires to order sets of primer pairs differing only by their tags. Whatever the strategy selected, the most important decision to make is to define which sequences should be used as tag. In a wonderful world, one could address this question only by taking into account the number of samples that need to be identified. The number of different DNA words of length $n$ can be calculated from the formula $4^n$. For tagging N samples, you have just to select the smallest n as $4^n > N$. As example, to tag $N = 300$ samples, you will need to use tags of length 5 ($4^4 = 256$ and

$4^5 = 1,024$). Then, by enumerating the 300 first words of length 5 (*AAAAA*, *AAAAC*, *AAAAG*, etc.), you can obtain your tag list.

### 2.2. Dealing with Sequencing Errors

Unfortunately, we are not in a wonderful world, and sequencers produce sequences with errors. Thus, the tag attached to each molecule can be sometimes read correctly and sometimes read erroneously. We are faced with a piece of information transmitted through a noisy channel. The emitter, e.g., the experimentalist, designs or emits a correct tag. The transmitter device is composed of a chained set of complex operations: primer synthesis, PCR amplification, sequencing process. Finally, the receiver is once again the scientist. This metaphor allows for linking our problem to the transmission information theory, a well-known problem in computer science. If we build our tag set (i.e., a code in the information transmission theory) following the simplest model and if an error occurs during the message transmission, we will read a wrong code after reception. Without the possibility to detect this error, we will assign the attached sequence to the wrong sample (see Fig. 1). Such a code is called a "no-error tolerant code".

For detecting reading errors, we must use a subset of all the possible words. Once we choose one word as a tag, i.e., *acggt*, if we preclude all words differing by only one letter (e.g., *aGggt*, *acTgt*, *Gcggt*, etc.), we guarantee that reading a tag with one error gives an erroneous tag corresponding to no sample. Given the Hamming distance ($d_H$) equal to the number of differences between two words, we can define a code as a set of DNA words where for all possible word pairs $(w_i, w_j)$, $d_H(w_i, w_j) \geq 2$. This new code is called a "one-error tolerant code" (see Fig. 2a).

Similarly we can build a "two-error tolerant code" by selecting a subset of words in such a way that for all possible word pairs $(w_i, w_j)$, $d_H(w_i, w_j) \geq 3$. This new code will lead to a sample misassignation only if more than two bases of the tag are erroneously read. This new code has a second property. If we consider that we can produce no more than one error during the transmission process, such a code not only detects the error but also allows correcting it. This is possible because only one of the used tags is present at a Hamming distance of one from the tag read (see Fig. 2b). Thus, a two-error tolerant code can be considered as a "one-error autocorrective code."

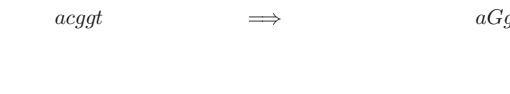


Fig. 1. Reading a tag of five nucleotides with one error can lead to sample missassignation. If all possible DNA tags composed of five nucleotides are assigned to a sample, we are not able to detect reading errors. We produce a no-error tolerant code. The capital letter corresponds to the reading error.

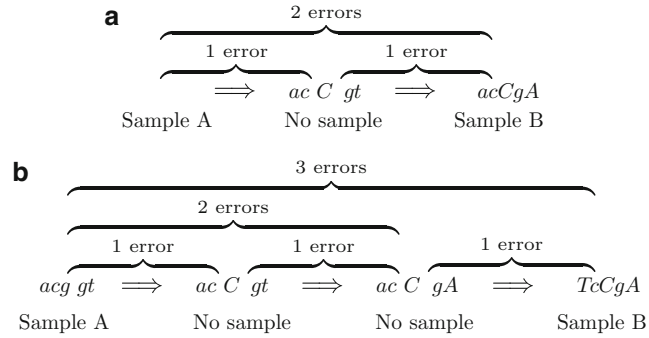


Fig. 2. Reading a tag of five nucleotides (**a**) If tag *acggt* is assigned to sample A and tag *accga* to sample B, when the tag *accgt* is read we can deduce that it is an error and discard the associated sequence. This corresponds to a one-error tolerant code. (**b**) If sample B is associated to tag *tccga* instead of *accga*. When the tag *accga* is read, we conclude to an error. This error can be explained as one reading error of sample B tag or as two reading errors of the sample A tag. This corresponds to a two-error tolerant code. If we assume that no more than one error is possible, only the first hypothesis is acceptable. We can keep the sequence and assign it to the sample B. Thus, we use the code as a"one-error autocorrective code".

### *2.3. Choosing a Set of Tags*

The choice of an adequate tag system depends mainly on two parameters: the number of samples to tag and the number of expected errors that is a combination of the probability to misread a base of the tag and the total number of sequences produced by the sequencer. To objectively decide, we can develop a simple probabilistic model. If we consider a homogeneous misreading probability $P_{\text{mis}}$ and tags of length $l_{\text{tag}}$ then the probability $P_{\text{l,e}}$ to read a tag with $e$ errors can be expressed as a binomial distribution (see Eq. 1).

$$P_{\text{l,e}} = \binom{l_{\text{tag}}}{e} P_{\text{mis}}{}^{e} \left(1 - P_{\text{mis}}\right)^{(l_{\text{tag}} - e)}. \tag{1}$$

The binomial part $\binom{l_{\text{tag}}}{e}$ estimates all the possible ways to position $e$ errors in a tag of length $l_{\text{tag}}$ and is computed using Eq. 2:

$$\binom{l_{\text{tag}}}{e} = \frac{l_{\text{tag}}!}{e!\,(l_{\text{tag}} - e)!}. \tag{2}$$

Using Eq. 1, it is easy to estimate how many tags would be read with 1, 2, 3, or 4 errors, when running a full GS-FLX 454 run or Solexa lane with one or 50 millions of reads, respectively (see Table 1). Even with a low error rate, the large number of reads leads to an expected large number of tags with up to three reading errors. This demonstrates the importance of taking into account errors when designing a set of tags. Table 1 shows that for a

**Table 1**
**Estimated numbers of misread tags: count computation is done for a tag length $l_{tag}=6$, the three used error rates correspond to values usually observed from 454 or Solexa runs. (e) is the number of errors in the tag**

| for a full 454 run $10^6$ reads | | | | for a Solexa lane $50.10^6$ reads | | | |
|---|---|---|---|---|---|---|---|
| errors | Error rate ($P_{mis}$) | | | errors | Error rate ($P_{mis}$) | | |
| ($e$) | 0.20% | 0.25% | 0.30% | ($e$) | 0.20% | 0.25% | 0.30% |
| 1 | 11880 | 14813 | 17731 | 1 | 594023 | 740671 | 886580 |
| 2 | 59 | 92 | 27 | 2 | 2976 | 4640 | 6669 |
| 3 | 0.16 | 0.31 | 0.54 | 3 | 7 | 15 | 26 |
| 4 | $2.4\ 10^{-4}$ | $5.8\ 10^{-4}$ | $1.2\ 10^{-3}$ | 4 | $1.2\ 10^{-2}$ | $2.9\ 10^{-2}$ | $6\ 10^{-2}$ |

GS-FLX 454 run with $P_{\text{mis}}=0.0025$, a tagging system of 6 nucleotides with a Hamming distance greater or equal to three will lead to misassign on average one sequence to a wrong sample every three runs while requiring discarding 14,905 reads (approximatively, 1.5% of the sequences). Considering the same tagging system as a one-error autocorrective code, we correctly reassign 14,813 of these reads to their respective samples but 92 reads with two errors are misassigned. Depending on the effect of such a level of misassignation, an autocorrective code can be considered as a good or a bad solution.

### 2.4. Tagging Both Ends of the PCR Amplicon

When sequencing a short enough PCR amplicon to be fully sequenced in a single read, it is possible to tag it at both ends (i.e., tagging the forward and the reverse primers). This double tagging reduces the sample misassignation probability. By checking the tags at each extremity, the only cause for misassignation is to observe the same errors on both tags. Let's consider that a reading error can change a nucleotide equiprobably into any of the three others, then we deduce Eqs. 3 and 4 from Eq. 1 to estimate the probability $P_{\text{l,e}^2}$ of observing the same $e$ errors on both extremities of a read. This probability function is used for estimating frequencies of sample misassignation with a double-tag system (see Table 2):

$$P_{\text{l,e}^2} = P_{\text{l,e}}\left(\frac{P_{\text{mis}}}{3}\right)^{e}\left(1-\frac{P_{\text{mis}}}{3}\right)^{(l_{\text{tag}}-e)} \tag{3}$$

$$= \binom{l_{\text{tag}}}{e}\left(\frac{4}{3}P_{\text{mis}}\right)^{e}(l_{\text{tag}}-e)^{\left(2-\frac{4}{3}P_{\text{mis}}\right)}. \tag{4}$$

When tags are added to both ends, even with only two differences between tags, almost no misassignation is possible (Table 2). By comparison, with a single-end tagging system, even with four differences between tags we cannot achieve the same level of

**Table 2**
**Estimated numbers of sample misassignations with a double-tag system: computation is done for a double-tag system of length $l_{tag} = 6$ without autocorrection. The three used error rates correspond to values usually observed from 454 or Solexa runs.(e) is the number of errors in the tag**

| | for a full 454 run $10^6$ reads | | | for a Solexa lane $50.10^6$ reads | | |
|---|---|---|---|---|---|---|
| errors | Error rate ($P_{mis}$) | | | errors | Error rate ($P_{mis}$) | | |
| ($e$) | 0.20% | 0.25% | 0.30% | ($e$) | 0.20% | 0.25% | 0.30% |
| 1 | 7 | 12 | 17 | 1 | 394 | 614 | 882 |
| 2 | $2.6\ 10^{-5}$ | $6.4\ 10^{-5}$ | $1.3\ 10^{-4}$ | 2 | $1.3\ 10^{-3}$ | $3.2\ 10^{-3}$ | $6.6\ 10^{-3}$ |
| 3 | $4.7\ 10^{-11}$ | $1.8\ 10^{-10}$ | $5.3\ 10^{-10}$ | 3 | $2.4\ 10^{-9}$ | $9\ 10^{-9}$ | $2.7\ 10^{-8}$ |
| 4 | $4.7\ 10^{-17}$ | $2.8\ 10^{-16}$ | $1.2\ 10^{-15}$ | 4 | $2.4\ 10^{-15}$ | $1.4\ 10^{-14}$ | $6\ 10^{-14}$ |

confidence (Table 1). But as the two copies of the tag can be misread independently, the frequency of discarded sequences is twice higher with a double-tag system than with a single-tag one (see Tables 1 and 2)

Like for a single-tag system, a double-tag system with a Hamming distance greater or equal to three can also be used as a one error autocorrective code. In such a system, a sequence with one or two erroneously read tags is reassigned to the good sample if when corrected, both tags match the same sample.

### 2.5. Lexical Constraints on Tag Design

A tag is a DNA word. To limit misreading, once the tag length is fixed, lexical rules can be set to restrict the usable words. For example, knowing the difficulties to read unambiguously homopolymers with the GS-FLX 454, we should exclude tags with more than $h$ consecutive identical letters. According to the same principle, if a set of tags is designed to be linked to a PCR primer with a nucleotide $X$ (*i.e.*, $A$, $C$, $G$, or $T$) at its 5′ end we should avoid this nucleotide at the 3′ end of the tag. Also, if a tag is linked to the primers during their synthesis, we can use only tags with a precise $G+C$ content to reduce the effect of the heterogeneity of the primer melting temperatures ($Tm$).

### 2.6. Building a Set of Tags

While some approximations were used to build efficiently a set of tags (8), the exact way for defining a set of tags relies on graph theory. In mathematics, a graph $\mathcal{G}(V, R)$ is defined by a set of nodes or vertices $V$ and a relation $R$ describing a set of edges $E$ linking some node pairs. In our particular case, $V$ is the set of all words matching our lexical constraints (length, $G+C$ content, maximum homopolymer length, etc.).

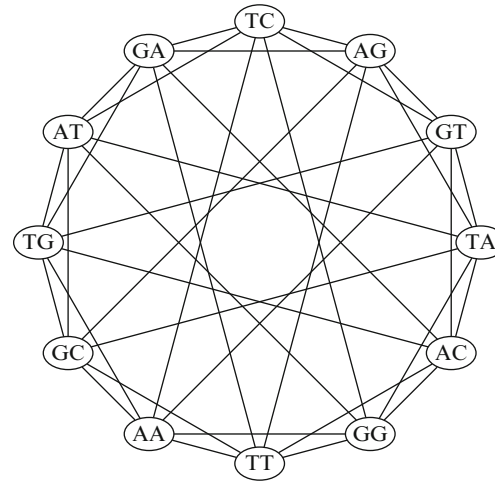


Fig. 3. Hamming graph for DNA words of length 2: This graph is built with all words of size two that do not begin by a *C* so $V=\{AA, AC, AG, AT, GA, GC, GG, GT, TA, TC, TG, TT\}$. The relation *R* is defined for a Hamming distance between two words $d_H \geq 2$.

$$V = \{w | w \text{ is a DNA words of length } l \text{ matching lexical constraints}\}. \quad (5)$$

*R* the set of edges is composed of DNA word pairs with a Hamming distance ($d_H$) greater or equal to *e* (i.e., the minimum number of differences between two tags).

$$R = \{(w_i, w_j) \in V \times V \mid d_H(w_i, w_j) \geq e\}. \quad (6)$$

In such a graph (see Fig. 3), a usable set of tags *T* is defined as a subset of *V* forming a complete subgraph (see Eq. 7). In the graph theory, *T* defines a clique. Identifying the best set of tags is equivalent to looking for the largest clique in the graph $\mathcal{G}$. Unfortunately, this requires a computational time that increases exponentially with the size of the graph. So only approximate solutions are computable.

$$T \subset \mathrm{V} \text{ in such a way that } \forall \{w_i, w_j\} \subset T \Rightarrow (w_i, w_j) \in R. \quad (7)$$

## 3. The *oligoTag* Program

*OligoTag* is part of OBITools (http://www.grenoble.prabi.fr/trac/OBITools), a set of UNIX command line programs dedicated to the analysis of the output from high-throughput sequencers.

### 3.1. Installing OBITools

To install OBITools, you need access to a Unix computer with Python language installed and a C compiler. It can be a PC with Linux or a Macintosh computer or any other Unix system. All the

following commands must be entered in a unix terminal window. Usually, on a linux machine, you can start such a windows from the application/utilities menu. On Macintosh, a similar application is available in the Applications/Utilities folder.

### *3.1.1. Checking Prerequisite*

To check if a C compiler is installed on your system, follow the instructions presented in Listing 1. If a gcc C compiler is installed on your computer, just by running the gcc command without argument you will have an error message "no input file" indicating that you have not specified a c file to compile.

```
unix–shell > gcc
i686–apple–darwin10–gcc –4.2.1: no input files
```

Listing 2 shows the result obtained if no C compiler is installed on your system: if a command is not installed on a system an error message "Command not found" is generated, then you must install a C compiler. If you are a Linux user, you must install the corresponding package from your package manager. If you are a Macintosh user, you need to install the "Developer tools" package available on the system DVD or on the Apple web site.

```
unix–shell > gcc
gcc: Command not found
```

You need Python 2.6 or 2.7, which should be available on all modern Unix system. To check your python version, follow the instruction presented in Listing 3. From a unix shell, you can run the python interpreter in interactive mode by typing the command python. This displays the python version, in our case python version 2.7. To quit python, just press keys Ctrl-D.

```
unix–shell > python
Python 2.7 (r27:82500, Jul 6 2010, 10:43:34)
[GCC 4.2.1 (Apple Inc. build 5659)] on darwin
Type " help", "copyright", "credits" or "license "for more information.
>>>
```

MacOSX users are invited to install a version of python downloadable from the python web site (http://www.python.org) even if a version of python is included by default in the system. Be careful, python 2.x and 3.x versions are almost incompatible, so don't use python 3.x with OBITools.

Finally, the SetupTool python package must be installed. First, you have to check the presence of the easy_install command as you did for gcc. If this command is absent, you can download it from the python package index web site (http://www.pypi.org) and follow the corresponding installation instructions.

*Installing OBITools Package*

OBITools can now be easily installed using the easy_install command (Listing 4). It can be necessary to begin this command line by the word sudo to access the administrator privilege.

```
unix–shell > easyinstall obitools
. . .
unix–shell >
```

### 3.2. OligoTag Options

Several options are available for specifying characteristics of the generated tag set. As in many unix programs, most of them exist in two forms. The short form corresponds to one letter preceded by a dash (e.g., -s). The long one is a full word preceded by a double dash (e.g., --oligo-size). Both forms of the same option are listed together. When an option requires a parameter like --oligo-size or -s, in the short form the parameter must follow directly the option (-s 5), whereas in the long form the option name and the parameter value must be separated by an equal sign (--oligo-size=5).

*Length of the Tags*

The length of the oligonucleotide is strongly related to the maximum size of the potentially identifiable tag set (see Subheading 2). The length must be an integer value greater or equal to 1. Values larger than 8 lead to huge memory usage and very long computation time (see option Subheading 3.2.9).

-s <###>, --oligo-size=<###>

<###> is an integer value corresponding to the generated length of the tags.

*Size of the Set of Tags*

This is the minimum number of tags required in the generated set of tags. These values must be set in relation with the option Subheading 3.2.1. Looking for a too large set with a too small size of tags leads either to no solution or to a very long computation time. To limit this effect, see also option Subheading 3.2.9.

-f <###>, --family-size=<###>

<###> is an integer value corresponding to the size of tag set to generate.

*Minimum Hamming Distance Between Two Tags*

This is the minimum Hamming distance $d_H$ between two tags of the solution set. This distance is associated with the chance of misassigning a sequence to a sample (see the parts Subheadings 2.3 and 2.4). Increasing the distance reduces the probability of assignation errors but reduces the size of the tag set (see options Subheadings 3.2.1 and 3.2.2).

-d <###>, --distance=<###>

<###> is an integer value corresponding to the minimum distance between two tags.

*Maximum Number of G or C Nucleotides per Tag*

This option lexically constraints a tag to be acceptable in a set by limiting the sum of G and C nucleotides. This can be used to limit the nonequivalent impact of GC rich and AT rich tags on the primer melting temperature. This constraint reduces the maximum size of the set of potentially identifiable tags.

-g <###>, --gc-max=<###>

<###> is an integer value corresponding the maximum number of G or C nucleotides acceptable in a tag

*Acceptable Tag Pattern*

Using the IUPAC code (Table 3), you can specify exactly the pattern of the tags to generate. The pattern must have the same length than the oligonucleotide size (see options Subheading 3.2.1) and must be constituted of a series of one of the IUPAC codes. If you set the oligonucleotide size to 6 using the option -s 6 and specify a pattern GNNBNR using the option -a gnnbnr, you will only accept

**Table 3**
**Nucleic IUPAC code used to represent nucleotides**

| Code | Nucleotide |
|---|---|
| A | Adenine |
| C | Cytosine |
| G | Guanine |
| T | Thymine |
| U | Uracil |
| R | Purine (A or G) |
| Y | Pyrimidine (C, T, or U) |
| M | C or A |
| K | T, U, or G |
| W | T, U, or A |
| S | C or G |
| B | C, T, U, or G (not A) |
| D | A, T, U, or G (not C) |
| H | A, T, U, or C (not G) |
| V | A, C, or G (not T, not U) |
| N | Any base (A, C, G, T, or U) |

tags starting on their 5′ end with G, with no A at their fourth position and a purine (A or G) at their 3′ extremity. A too restrictive pattern can drastically reduce the maximum size of the potentially identifiable tag set.

-a <IUPAC pattern>, --accepted=<IUPAC pattern>

<IUPAC pattern> a string describing the IUPAC pattern of acceptable tags.

*Non acceptable Tag Pattern*

Reciprocally to the Subheading 3.2.5 option described above, you can specify a pattern indicating the tag that must not be include in a set using the IUPAC code (Table 3). Using this option can drastically reduce the maximum size of the potentially identifiable tag set.

-r <IUPAC pattern>, --rejected=<IUPAC pattern>

<IUPAC pattern> a string describing the IUPAC pattern of unacceptable tags.

*Maximum Homopolymer Length*

Homopolymers may cause many PCR and sequencing errors, especially when using the GS-FLX technology. To limit sample missassignation, it is reasonable to limit the length of homopolymers in tags to two. Only tags with no homopolymer longer than the specified limit will be retained in the tag set.

-p <###>, --homopolymer=<###>

<###> is an integer value corresponding the maximum length of an homopolymer.

*Minimum Homopolymer Length*

This reciprocal option of the previous one is normally less useful. Only tags with at least one homopolymer longer or equal to the specified limit can be retained in a tag set.

-P <###>, --homopolymer-min=<###>

<###> is an integer value corresponding the minimum length of an homopolymer.

*Computation Time Out*

Computation of a tag set is divided into two steps. During the first one, the Hamming distance graph is built according to all the options specified. Then a maximum clique algorithm looks for the cliques larger than the family size limit. If you ask for a too large tag set, this second part can be infinitely long. So it is useful to specify a time out limit (in seconds) for this search. In practice, it is really rare to find an interesting solution in more than ten minutes, so 600 s is a good compromise.

-T <###>, --timeout=<###>

<###> is an integer value expressed in seconds indicating the maximum time that the program can spend to look for a set of the required size. If this time is over, then the largest set of tags found is returned instead.

### 3.3. Running oligoTag

OligoTag is a unix command line program and must be used from a unix terminal windows. From a unix shell, by typing the following command line (Listing 5), we look for a solution corresponding to the graph presented in Fig. 3. The option $-s\,2$ indicates the size of the word (i.e., tag length). The option $-f\,1$ indicates the minimum size of the desired set of tags. The option $-d\,2$ indicates the minimum Hamming distance $d_H$. Finally, the $-r\,CN$ rejects all words matching a given pattern (here $CN$ with $N$ meaning $A, C, G, \{$ or $\}T$).

```
unix–shell > oligoTag –s 2 –f 1 –d 2 –r CN
Build good words graph . . .
Initial graph size : 12 edge count : 36
aa
gc
tg
unix–shell >
```

The proposed solution is reported in Fig. 4. As the exact solution of the problem defined in part Subheading 2.6 cannot be computed in a reasonable time, oligoTag cannot guaranty to find

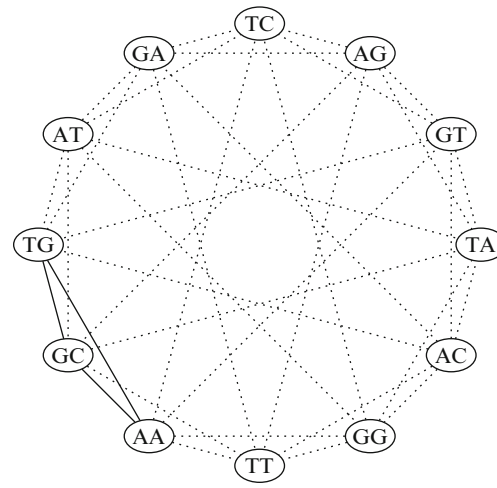


Fig. 4. Solution proposed by oligoTag for DNA word of length 2: Solution proposed by oligoTag is reported on this graph with the filled edges. This clique is maximum, you cannot add another vertex without rejection of the formula 2.7. In this particular case, this solution also corresponds to one of the largest cliques in this graph.

the largest tag set. OligoTag will just try to identify a clique that cannot be extended with a cardinality greater than a threshold defined via the option *–f*.

OligoTag is a standard unix command, so by using the > redirection character you specify to save the output of the oligoTag program to the mytag.txt text file. You are free to choose the output file name. The cat command allows you to read the content of the newly created file (Listing 6).

```
unix–shell > oligoTag –s 2 –f 1 –d 2 –r CN > mytag . txt
Build good words graph . . .
Initial graph size : 12 edge count : 36
unix–shell > cat mytag . txt
aa
gc
tg
unix–shell >
```

## 4. Examples of Precomputed Tag Lists

All these tag lists were computed with oligoTag specifying a minimum Hamming distance of 3 and no homopolymer longer than 2 (see Table 4).

## 5. Analyzing Tagged Sequences

The OBITools are also useful for analyzing tagged raw sequences. They allow identifying and trimming the amplification primers and tags. They allow assigning sequences to a sample according to its tag and distributing these annotated sequences according to their associated sample and experiment in several files. These tasks can be achieved using three OBITools programs: ngsfilter, obiannotate, and obisplit. Several other OBITools exist and are not presented here despite their potential utility. Like oligoTag, all other OBITools programs are unix command line tools. This allows chaining them using simple unix scripts and then automatizing the treatment by defining more or less complex pipelines.

### *5.1. The ngsfilter Program*

Based on the description of the PCR primers used and of the tag associated to each multiplexed sample, ngsfilter looks for the forward and reverse primers, matches the flanked tag, and annotates the sequence with its experiment and sample names. At the same

**Table 4**
**Example of tag lists computed with oligoTag: Each part of the table corresponds to a different tag (4 to 7) length and family size options. The oligoTag unix command and the associated options used to generate each of these sets are indicated**

| **Tag size : 4** | | | | **Tag count : 11** | | | | |
|---|---|---|---|---|---|---|---|---|
| **unix command : oligoTag -d 3 -p 2 -s 4 -f 10** | | | | | | | | |
| aaca | acac | ctcc | gctt | tatc | aggt | tccg | tgaa | gtga |
| attg | caag | | | | | | | |
| **Tag size : 5** | | | | **Tag count : 33** | | | | |
| **unix command : oligoTag -d 3 -p 2 -s 5 -f 24** | | | | | | | | |
| aacaa | aagcc | gactt | gtaat | cgagg | aatgg | cgcat | acaac | tctaa |
| gtcca | ggtta | attct | accgt | gtggc | caatc | gccag | acgta | tgatt |
| taacg | gcgct | agaca | catca | tacgc | ctgtt | tgtcc | agctc | aggag |
| ttaga | cttag | tagat | gatac | cctgc | atatg | | | |
| **Tag size : 6** | | | | **Tag count : 108** | | | | |
| **unix command : oligoTag -d 3 -p 2 -s 6 -f 96** | | | | | | | | |
| aacaac | aaccga | ccggaa | agtgtt | ccgctg | aacgcg | ggctac | ttctcg | ttctcg |
| tcactc | gaacta | ccgtcc | aagaca | cgtgcg | ggtaag | ataatt | cgtcac | cgtcac |
| ttgagt | aagcag | ttgcaa | cacgta | taacat | tgcgtg | ggtcga | cactct | cactct |
| cttggt | tccagc | acttca | gcgaga | tggaac | gtacac | aagtgt | tcttgg | tcttgg |
| aaggtc | ggcgca | tcgacg | cctgtc | agaaga | aatagg | ggttct | taatga | taatga |
| gtaaca | aatcct | agaccg | tggcgg | ctataa | aatgaa | cgaatc | agagac | agagac |
| ttcgga | cgacgt | ctcatg | tgtata | acaacc | tcagag | gtagtg | agcact | agcact |
| gcggtt | acacaa | gctccg | tacttc | gttgcc | gtatgt | gtcaat | agcctc | agcctc |
| tcgtta | tgtggc | ctctgc | atggat | acaggt | tccgct | gtccgg | cattag | cattag |
| gaagct | gatatt | agctgg | cgcgat | acattg | ccaagg | accata | aggatg | aggatg |
| gtctta | tatacc | acctat | aggtaa | attcta | gtgatc | gacggc | gtgcct | gtgcct |
| tatctg | cggcca | cctaat | acgcgc | gtgtag | ttcctt | cagagc | tgatcc | tgatcc |
| **Tag size : 7** | | | | **Tag count : 316** | | | | |
| **unix command : oligoTag -d 3 -p 2 -s 7 -f 300** | | | | | | | | |
| aacaaca | aacacac | tccgact | tgcctgc | cgtcgca | ggtcagt | ccgctag | acgcggc | acgcggc |
| gaagctg | aacaggt | gccggcc | ggtcgag | taagcct | aacattg | aaccaag | agtgaag | agtgaag |
| ccggatc | ggtctcc | aaccgcc | ccggcct | ttcggcg | agtgcca | taagtgg | acggcga | acggcga |
| tcctaag | aacctga | ggtgatc | acggtac | aacgagc | tggcacc | taatagt | ccgtaat | ccgtaat |

(continued)

**Table 4**
**(continued)**

| **Tag size : 7** | | | | **Tag count : 316** | | | | |
|---|---|---|---|---|---|---|---|---|
| **unix command : oligoTag -d 3 -p 2 -s 7 -f 300** | | | | | | | | |
| gaatgat | aacgccg | caagtat | acgtacc | ggtccta | aacggaa | tcctgca | ggtgtga | ggtgtga |
| taattaa | ggttaca | tccttgt | aagagta | ccgttca | aactatt | tccgcgc | agttggc | agttggc |
| cctaacg | ggttcgg | gacataa | ataacgc | actaagt | gcgatcc | tcgagcg | tacagtc | tacagtc |
| ggttgtt | gtaacag | ctcacct | ataagtg | cctagga | aagaagg | gcctctg | tgtatag | tgtatag |
| aagacct | gtaagct | tggccgt | tcgcagg | gcgcctt | taccgat | tcgccac | gtaatga | gtaatga |
| gaccttc | cctcata | gtacaat | cctccgc | cacgctt | gacgcga | tcgctct | aagcatc | aagcatc |
| gtacctc | aagccaa | cgtgact | actctaa | taaggta | gacgtct | gtactcg | gcttgaa | gcttgaa |
| cctgcaa | aagctcg | gtagacc | tgtgctg | aaggaat | tactacc | tcggttg | cacttac | cacttac |
| gtagcgt | cctggtg | gtaggaa | tatactt | actggct | tactcta | ctagtta | cctgtgt | cctgtgt |
| atatact | gcgtgtc | gagaatt | gtatagg | gaatata | acttcag | taagaac | gtatcca | gtatcca |
| aagtcgc | caatgtc | gctactc | aagtgag | cgctcgt | cgaacat | ctattgt | agaaccg | agaaccg |
| gtattac | ctcaata | tagatac | gtcaacg | agaagga | aatacga | agaatac | cgaattg | cgaattg |
| tctacca | aatagcg | gtcagac | cagctgt | ggtgcat | aatatat | ttaccga | agacagg | agacagg |
| aatcact | ctccgaa | agacgat | atcctat | tagctta | aatcctg | taggaga | agactca | agactca |
| tctcgcc | gtccgtg | acaagcc | cagtatg | taggctc | agagatt | ctcgcgg | cgaggcc | cgaggcc |
| gtcgatt | gcttagc | tagtccg | gcttcct | gtcgtag | tcttatt | aatgtcc | gagttga | gagttga |
| agtaacc | atctctc | tagtgtt | aattaac | ttatctg | cagaggc | gtccaga | cgatgaa | cgatgaa |
| ttatgcc | agatcta | tataagc | gtctcat | ctctggc | ggaagtc | catattc | ctcttcg | ctcttcg |
| tatagaa | cgcaagg | ttcacaa | catcagg | acaacaa | tacgatg | tgtggac | ctgacac | ctgacac |
| tgaatct | agcactt | catcgac | gtgaagc | agcagag | ttcatcc | gtgacta | acaatgg | acaatgg |
| ttccaac | acacaac | atgcagt | tgacgcg | agccata | gtgatat | acaccgt | cgagaga | cgagaga |
| acacgta | gtgccgg | ctgcgtt | gatggca | gactgcg | acagaca | atggcag | ctggacg | ctggacg |
| cggttgc | cattcat | cgcggat | agcggtc | acagctc | atggtct | acaggag | cgcgtca | cgcgtca |
| ttcgtga | agcgtgg | tcgaata | agctacg | tgatcac | cgctatc | ctgtaga | gtggttc | gtggttc |
| acatatg | agcttaa | atgttgg | cgtctat | cggaaca | ggcagca | acattat | tacacgg | tacacgg |
| tcaagtt | gtgtggt | accaatc | ttgagga | cttagat | ctacatg | aatgata | aggatgt | aggatgt |
| accatct | cttatca | gcactgc | accgcat | cggcctc | gttattg | ttgcgag | cttccag | cttccag |
| aggcgtg | attcttc | accgtta | gttcggc | acctaga | gtgcgca | ccatcga | tcagtcc | tcagtcc |
| aggtcat | ttggcca | cttgagc | attgctt | ccatgcg | aggtgca | acctgac | gttgccg | gttgccg |
| ttgtatc | ctgctcc | tgtaggt | caaccta | tgctgtg | acgactg | caacgct | gccacgt | gccacgt |
| taacaca | acgagat | ttgaact | taaccag | agtatta | taacggc | agtccac | gccgagg | gccgagg |
| caagcgc | | | | | | | | |

```
@HWUSI-EAS1510_0005:1:1:12765:8937#0/1_SUB_CONS
ccagctcagtggggcaagcctcagccgctatccgtgtctttgtaatctcatgggagaa
+HWUSI-EAS1510_0005:1:1:12765:8937#0/1_SUB_CONS
H:555999<9DAD33=\BGcK2EYDS==4^HET?TFOAJQLA4?3T:B:@B?B^V‘?:
@HWUSI-EAS1510_0005:1:1:12866:6509#0/1_SUB_CONS
ccctctgctcagggcaatcctcagcaccaatccttttttttagtattcgaatgatgaac
+HWUSI-EAS1510_0005:1:1:12866:6509#0/1_SUB_CONS
I>;;>DAADD_L@aF:XBGSI?B^@=@27‘UCN=F1T[;B_@B@FDB@B52@4@G441
```

Fig. 5. The first two sequences of a Solexa run in fastq format: each sequence is described by four lines. The first line starting with the @ character is the equivalent of a fasta title line. The next line gives the sequence. The third line starting with a + character repeats the title line, and the fourth line corresponds to the encoded quality line.

time, the primers and tag copies are trimmed out and only the amplified part of the sequence is kept in the output.

The input file can be a raw sequence file in fastq or fasta format. It can also be a pair of fasta and quality files as provided by GS-FLX sequencers. Depending on the input format, the output can be formatted in fasta or fastq. Fastq is an extension of fasta format including *per base quality values* (see Fig. 5) (9).

*The Main ngsfilter Options*

For simplicity reasons, only the most useful options of ngsfilter will be listed below.

*Specifying the Input Format*

Like all OBITools, ngsfilter usually automatically recognizes the input read file format. However, you can specify the encoding quality schema of the fastq files:

- --sanger: input file is in sanger fastq nucleic format (standard and default fastq format)
- --solexa: input file is in fastq nucleic format produced by a Solexa sequencer (Illumina 1.3+)
- --illumina: input file is in fastq nucleic format produced by an old Solexa sequencer (Solexa/Illumina 1.0 format)

For the fasta format, the --fna and the --qual=<filename> options allows specifying that the fasta file was produced by a GS-FLX sequencer and what the associated quality file (file with the .qual extension) is.

*Specifying the Output Format*

If the input file contains quality data, then the default output format is fastq, otherwise fasta format will be used. Fastq outputs are always encoded following the Sanger rules. You can force the output to fasta or fastq format using the following options, respectively : --fastq-output or --fasta-output. If the output if forced to fastq without providing quality data in the input, a default quality of 40 will be associated with each base.

```
Sol3-gh_R   EH1087    acacgctct   GGGCAATCCTGAGCCAA      CCATTGAGTCTCTGCACCTATC   F
Sol3-gh_R   EH0380B   acacgtcgt   GGGCAATCCTGAGCCAA      CCATTGAGTCTCTGCACCTATC   F
Sol3-gh_R   EH0379B   acactctgc   GGGCAATCCTGAGCCAA      CCATTGAGTCTCTGCACCTATC   F
Sol3-ITSA   EH1087    cagctgatg   GATATCCGTTGCCGAGAGTC   GCACGGCATGTGCCAAGG       F
Sol3-ITSA   EH0380B   cagctgtga   GATATCCGTTGCCGAGAGTC   GCACGGCATGTGCCAAGG       F
```

Fig. 6. Sample description file: The full set of multiplexed samples must be described in a tabular text file containing six columns. The first two describe, respectively, the experiment, and the sample, the third one the tag associated with this sample. The fourth and fifth columns indicate the forward and reverse PCR primers. The sixth column indicates if partial sequences (i.e., incomplete amplicons) are expected with T (True) or F (False).

*Specifiying the Sample Description File*

The file describing all the samples multiplexed in one sequencer lane must follow the format presented in Fig. 6. Its name is specified to ngsfilter using either the short -t <filename> option or its long version --tag-list=<filename>.

*Running ngsfilter*

In this example, ngsfilter is used to analyze a fastq file named myrun.fastq produced by a Solexa sequencer. By using the > redirection character, the output is saved to the good.fastq text file. During the execution of ngsfilter, a progress bar with an estimation of the remaining computation time is displayed

```
unix–shell > ngsfilter –t samples . tag —solexa myrun . fastq > good . fastq
myrun . fastq 100.0 % | ############################–] remain : 00:00:00
unix–shell >
```

### *5.2. The obiannotate Program*

ngsfilter like other OBITools annotates the sequences in fasta or fastq format using a tag/value system. These tag/value pairs are included in the title line of each sequence (see Fig. 7). obiannotate is used to change the annotation associated to each sequence.

*Running obiannotate*

After ngsfilter, you can clean the file for keeping only the experiment and sample annotations. The option -k allows specifying which annotation must be kept (see Listing 8). Then sequences will look like those presented in Fig. 8. Cleaning the annotated sequences to keep only the experiment and sample annotations can be done following Listing 8.

```
unix–shell > obiannotate –k exper iment –k sample good. fastq > clean . fastq
good. fastq 100.0% | ############################–] remain : 00:00:00
unix–shell >
```

### *5.3. The obisplit Program*

Obisplit allows distributing sequences from the file annotated and cleaned by ngsfilter and obiannotate into different files according to

```
@HWUSI-EAS1510_0005:1:2:13183:12244#0/1_CONS_SUB reverse_score=88.0; tag_length=9;
tail_quality=28.2; reverse_match=ccattgagtctctgcacctatc; direct_tag=acacgctct;
sample=EH1087; reverse_tag=acacgctct; reverse_primer=ccattgagtctctgcacctatc;
direct_score=68.0; cut=[28,42,1]; direct_match=gggcaatcctgagccaa;
direct_primer=gggcaatcctgagccaa; experiment=Sol3-gh_R; mid_quality=32.3703703704;
head_quality=40.4; avg_quality=32.8918918919;
tggctcagctgtgg
+
LLG4BGC@>BB5:C
@HWUSI-EAS1510_0005:1:2:14001:3874#0/1_CONS_SUB_CMP reverse_score=88.0; tag_length=9;
complemented=True; tail_quality=24.8; reverse_match=ccattgagtctctgcacctatc;
direct_tag=acactctgc; sample=EH0379B; reverse_tag=acactctgc;
reverse_primer=ccattgagtctctgcacctatc; direct_score=68.0; cut=[33,50,1];
direct_match=gggcaatcctgagccaa; direct_primer=gggcaatcctgagccaa; experiment=Sol3-gh_R;
mid_quality=40.3103448276; head_quality=43.7; avg_quality=38.7564102564;
atcttattctaaaatga
+
H0E71A?>QN5_\QUO]
```

Fig. 7. Output ngsfilter file formatted in fastq format: The first two sequences analyzed are provided. In contrary to the classical fastq format, the long title line is not repeated twice. Several fields were added to the title line. complemented=True in the second sequence indicates that the reverse complement of the sequence has been used to find the primers. head_quality is the average quality of the first 10 bases of the sequence. mid_quality is the average quality of the central part of the sequence. tail_quality is the average quality of the last 10 bases of the sequence. avg_quality is the average quality of the whole sequence. direct_primer is the true sequence of the identified forward primer. direct_score is the alignment score of sequence with the forward primer. direct_match is the real sequence of the forward primer identified on the sequence. reverse_primer is the true sequence of the identified reverse primer. reverse_score is the alignment score with the reverse primer. reverse_match is the real sequence of the reverse primer identified on the sequence. cut indicates where the original sequence was truncated. tag_length is the length of the tag found. direct_tag is the sequence of the tag found on the 5′ side of the forward primer. reverse_tag is the sequence of the tag found on the 5′ side of the reverse primer. sample is the sample id associated with the sequence. experiment is the experiment id associated with the sequence.

```
@HWUSI-EAS1510_0005:1:2:13183:12244#0/1_CONS_SUB sample=EH1087; experiment=Sol3-gh_R;
tggctcagctgtgg
+
LLG4BGC@>BB5:C
@HWUSI-EAS1510_0005:1:2:14001:3874#0/1_CONS_SUB_CMP sample=EH0379B; experiment=Sol3-gh_R;
atcttattctaaaatga
+
H0E71A?>QN5_\QUO]
```

Fig. 8. Two sequences cleaned by obiannotate.

the value of one of their annotations. This can be seen as a demultiplexing of the sequences. A first level of demultiplexing must be done on the base of the experiment: in Listing 9, the clean.fastq file is splitted in as many files as necessary, each file storing the subset of sequences with the same experiment value. All file names will begin by the prefix myrun_ and finish with the experiment value.

unix–shell > obisplit –t exper iment –p myrun clean . fastq

```
good . fastq 100.0% | ##########################–] remain : 00:00:00
unix–shell >
```

A second split of each of the created file can then be done according to the sample tag.

## References


1. O'Brien HE, Parrent JL, Jackson JA et al (2005) Fungal community analysis by large-scale sequencing of environmental samples. Appl Environ Microbiol 7:5544–5550
2. Briée C, Moreira D, López-García P (2007) Archaeal and bacterial community composition of sediment and plankton from a suboxic freshwater pond. Res Microbiol 158:213–227
3. Fierer N, Morse JL, Berthrong ST et al (2007) Environmental controls on the landscape-scale biogeography of stream bacterial communities. Ecology 88:2162–2173
4. Nicol GW, Leininger S, Schleper C, Prosser JI (2008) The influence of soil ph on the diversity, abundance and transcriptional activity of ammonia oxidizing archaea and bacteria. Environ Microbiol 10:2966–2978
5. Zinger L, Coissac E, Choler P, Geremia RA (2009) Assessment of microbial communities by graph partitioning in a study of soil fungi in two alpine meadows. Appl Environ Microbiol 75:5863–5870
6. Teixeira LCRS, Peixoto RS, Cury JC et al (2010) Bacterial diversity in rhizosphere soil from antarctic vascular plants of admiralty bay, maritime antarctica. ISME J 4:989–1001
7. Cronn R, Liston A, Parks M et al (2008) Multiplex sequencing of plant chloroplast genomes using solexa sequencing-by-synthesis technology. Nucleic Acids Res 36:e122
8. Hamady M, Walker JJ, Harris JK et al (2008) Error-correcting barcoded primers for pyrosequencing hundreds of samples in multiplex. Nat Methods 5:235–237
9. Cock PJA, Fields CJ, Goto N et al (2009) The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. Nucleic Acids Res 38:1767–1771

http://www.springer.com/978-1-61779-869-6

Data Production and Analysis in Population Genomics
Methods and Protocols
Pompanon, F.; Bonin, A. (Eds.)
2012, XI, 337 p. 67 illus., 16 illus. in color., Hardcover
ISBN: 978-1-61779-869-6
A product of Humana Press