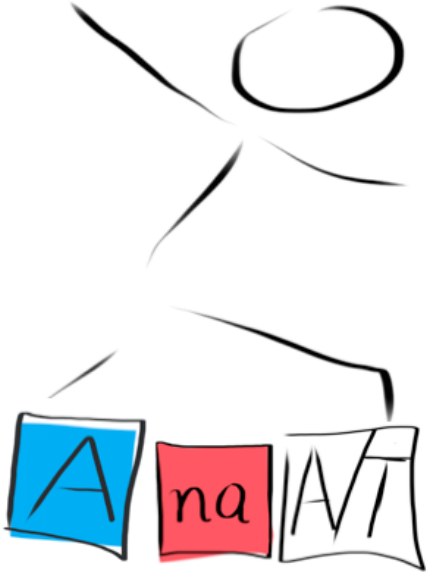
Gap.Jumper v.01

Manual and Tutorial

Pawel Rosikiewicz[1], Frédéric G Masclaux[1,2]

[1] Department of Ecology and Evolution, University of Lausanne, Quartier Sorge, Bâtiment Biophore, Lausanne, 1015, Switzerland; [2] Vital-IT, SiB, Swiss Institute of Bioinformatics, Quartier Sorge, Bâtiment Génopode, Lausanne 1015, Switzerland



June 12, 2016

Contents:

Chapter 1

Introduction

1.1 General Information

Gap.Jumper is a program that uses single nucleotide variant (SNV) calling data from several replicates of a sample to construct a consensus for that sample. Replicates can be independent biological replicates or technical replicates. Each replicate should be sequence and genotyped independently (Figure 1).

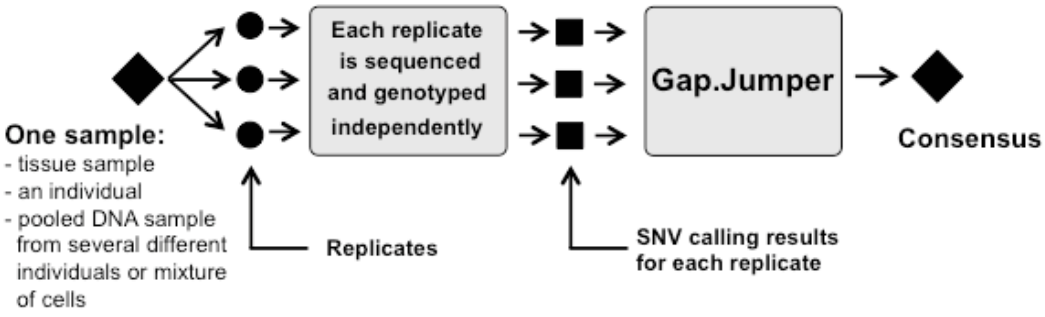


Figure 1. Construction of a consensus for a given sample from SNV calling data that were obtained with many replicates of that sample.

1.2 The purpose of Gap.Jumper

Different replicates of a sample potentially have different information (Figure 2). Firstly, because of missing data at different positions in different replicates ("na"). Secondly, because of amplification and sequencing errors. Finally, because some of replicates may be contaminated with DNA from other samples. For this reason, Gap.Jumper constructs a consensus for each sample using SNV calling data from many replicates of that sample. Additionally, it allows obtaining the information for a larger number of positions than by using only one replicate.

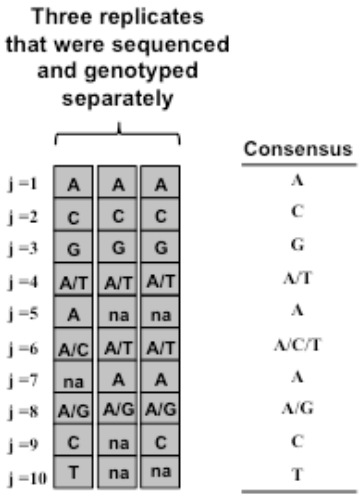


Figure 2. The schematic diagram showing construction of a consensus with SNV calling data at ten different positions (j) in three different replicates. The consensus was constructed using All Recorded Nucleotides method (Chapter 2)

1.3 Construction of a consensus

SNV calling data from many replicates can be use in many different ways to construct a consensus. It depends on type of SNV calling data (coverage, ploidy of the organisms, error rate in sequence data and the amount of missing data in each replicate). It also depends on the type of project (whether, the consensus is build for one individual or a population). For this reasons, Gap.Jumper v.01 allows construction of consensus from many replicates using six different methods that are explain in Chapter 2.

1.4 Gap.Jumper package

Gap.Jumper v.01 is a software package comprising two separate modules. The first module is VCF.Converter. It is composed of two Perl scripts that are called GapJumper_01_Record_all_positions_from_VCF.pl and GapJumper_02_Simplified_VCF_maker.pl. The second module, GapJumper.plus is implemented in R programming language. It is composed of one R script: GapJumper_03_GapJumperPlus.R. It contains all R functions that are used to build the consensus with each of six different methods (Chapter 2). Additionally, The Gap.Jumper comes with three sets of example files that are used in Gap.Jumper tutorial (Chapter 4-7). They are stored in three folders called GapJumper_Example One to Three.

1.5 Downloading Gap.Jumper

Gap.Jumper can be downloaded from GitHub:

1.6 Licence

Gap.Jumper can be distributed, copied and modified without any charge under the terms of GNU General public License; version 3 of that License. For more information see http://www.gnu.org.

WE PROVIDE THE PROGRAM WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.7 Reporting Bugs

All bugs and question about Gap.Jumper v.01 use can be directed to gapjumper.plus@gmail.com.

Please start your email with saying whether you have problem

with VCF.Converter or Gap.Jumper.plus and, which cversion of Gap.jumper you are using.

1.8 Citing Gap.Jumper

Currently submitted

Chapter 2

Six different methods to build a consensus

Gap.Jumper constructs a consensus for each set of replicates with one of six different methods. They are: (a) Probabilistic method (b) Probabilistic beta method (c) Conservative method, (d) Site elimination method, (e) Fixed thresholds method, and (f) All recorded nucleotides method. Methods a and c - e were described in detail in Rosikiewicz et al. (2016). The example of consensus constructed using each of these six methods is given in *GapJumper_Example_Two_Results* folder on GitHub (.multiSNV files). Extension in each .multiSNV file in this folder corresponds to the name of method that was used to construct that consensus.

2.1 Probabilistic method

The probabilistic method constructs a consensus in which each nucleotide at each position obtains a confidence score ranging from zero to one. The probability for each nucleotide is calculated based on procedure described in Rosikiewicz et al (2016).

2.2 Probabilistic beta method

The probability for each nucleotide is calculated in the same way as with the probabilistic method (Rosikiewicz et al 2016). However, the weight that is a part of that calculation was modified for positions with missing data. On each position, the weight, which is calculated for missing data is divided by the number of replicates without missing data at that position. This method can be use with datasets that have large number of replicates (min three replicates are required) and with large amount of randomly missing data.

2.3 Conservative method

The consensus contains only the information at positions, which have exactly the same nucleotide composition in all replicates.

2.4 Site Elimination method

A position is removed if it contains missing data in at least some of the replicates.

2.5 Fixed thresholds method

Three different thresholds describe the minimal acceptance criteria for each nucleotide and each position.

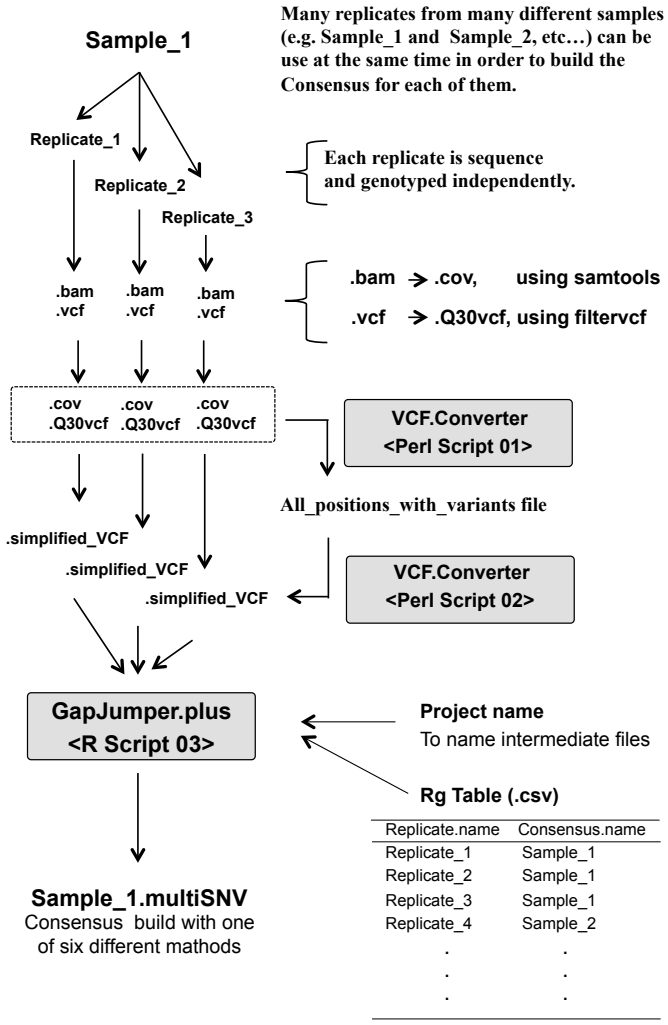
2.6 All recorded nucleotides method

All Recorded Nucleotides method construct a consensus from each set of replicates with all nucleotides at each position that were present in at least one replicate. No threshold is applied in this method. GapJumper always build a consensus for each set of replicates with this method, irrespectively, which other method is used.

Chapter 3

Gap.Jumper Pipeline

Gap.Jumper constructs a consensus in two successive steps. First, it transforms .vcf files (or filtered .vcf files) and .cov files into simplified_VCF files using VCF.Converter (Perl Script 01 and 02, Figure 3). In Chapter 4, we describe how to prepare these files and how to generate simplified_VCF files using VCF.Converter. Subsequently, Gap.Jumper.plus module generates .multiSNV file for each set of replicates that contains a consensus build with one of six different methods (Figure 3). R functions that were implemented in Gap.Jumper.plus module (R Script 03), and additional elements that are required to build a consensus with each method (project name, Rg table) are described in Chapters 5-10.



Replicate.name	Consensus.name
Replicate_1	Sample_1
Replicate_2	Sample_1
Replicate_3	Sample_1
Replicate_4	Sample_2
.	.
.	.
.	.

Figure 3. Schematic diagram showing the pathway of construction of a consensus from several replicates of a Sample_1. It is important to notice that the replicates (1 to 3) can be used together with the replicates from other samples (Sample_2, Sample_3 etc…). In this case, Gap.Jumper constructs a separate consensus for each of these samples using the replicates that are specified in Rg_table. All replicates that are in one Rg_table should be map to the same reference genome and treated together with VCF.Converter (i.e. build using the same All_positions_with_variants.. file).

Chapter 4

Generating simplified_VCF files

4.1 VCF.Converter generates .simplified_VCF files for each replicate

VCF.Converter constructs .simplified_VCF file for each replicate in each set of replicates. A simplified_VCF is a alternatively structured .vcf file, which clearly shows the allelic composition of each genomic site. It is particularly easy to know if the reference allele is present. All the positions where variants were found at least in one of the replicates in each set of replicates are reported in each simplified_VCF file. Thus each simplified_VCF file generated at the same time have the same positions reported. For more information see Chapter 7.

4.2 VCF.Converter

VCF.Converter. is composed of two Perl scripts:

a) GapJumper_01_Record_all_positions_from_VCF.pl (Perl script 01)
b) GapJumper_02_Simplified_VCF_maker.pl. (Perl script 02)

4.3 Requirements for VCF converter

VCF.Converetr requires two types of files for each replicate:

a) classic variant calling files (.vcf), generated by the variant calling program Freebayes
b) depth of coverage files (.cov), generated by 'samtools depth' command.

Comments:
(1) Both types of files are generated from a binary file containing sequence alignment to reference genome (.bam file).
(2) As for many tools, it is important that the reference genome file is formatted before running aligners or variant callers. Spaces, pipes ("|"), tabulations should be avoided in the headers of the .fasta file. The headers should be kept simple like "Scaffold_1" or "Chr1".
(3) GapJumper v.01 is compatible with .vcf files generated by the variant calling program Freebayes. We give no warranty that it will work with .vcf files generated with other variant callers.

4.4 Before you start

a) Make sure that have installed:
- samtools (http://www.htslib.org/)
- vcffilter (https://github.com/vcflib/vcflib)
- perl module : Sort::Naturally installed with the command: sudo cpan Sort::Naturally

a) Download from GitHub
- *GapJumper* folder
- *GapJumper_Example_One* folder
- *GapJumper_Example_One_Results* folder

Comments:

GapJumper folder contains two Perl scripts (01 and 02) that are required to run VCF.Converter. The GapJumper_Example_One folder contains example input files for 6 replicates (.vcf files, .Q30vcf files, .bam files and .cov.files). The GapJumper_Example_One_Results folder contains simplified_VCF files and Simplified_VCF_maker.pl All_positions_with_variants… that should be generated using input files from GapJumper_Example_One folder.

4.5 Create *VCF_Converter_working_dir* folder

Copy to this folder:
a) GapJumper_01_Record_all_positions_from_VCF.pl
b) GapJumper_02_Simplified_VCF_maker.pl
c) .bam files (one for each replicate)
d) .vcf files (one for each replicate)

Comments:

In the case of problems with installing samtools or vcfilter on your computer or cluster, .cov files and filtered .Q30vcf files for each example replicate were provided in GapJumper_Example_One folder. In this situation, copy these files into VCF_Converter_working_dir folder and omit the two following steps (4.5 and 4.6)

4.6 Filter .vcf files

In order to keep high quality positions in the consensus filter the .vcf files.
vcffilter -f 'QUAL > 30' Sample.vcf > Sample_Q30.vcf'

Comments:

This step is optional but highly recommended !!!.
VCF.Converter also can be run with data that are not filtered.

4.7 Generate coverage files (.cov)
samtools depth Sample.bam > Sample.coverage

4.8 Record all variable positions from the .vcf files

In *VCF_Converter_working_dir:*
perl GapJumper_01_Record_all_positions_from_VCF.pl

Comments:

(1) The script will read all .vcf files in the current directory.
(2) Output = All_positions_with_variants201XXXXX.combinedVCF
where: 201XXXXX is the date of file creation
(3) An example of the All_positions_with_variants201XXXXX.combinedVCF file is provided in the *GapJumper_Example_One_Results* folder

4.9 Generate .simplified_VCF file for each replicate

In *VCF_Converter_working_dir:*
perl GapJumper_02_Simplified_VCF_maker.pl
All_positions_with_variants201XXXXX.combinedVCF Sample_Q30.vcf Sample.coverage

Comments:

(1) The script will read all .vcf files in the current directory.
(2) The name of the simplified VCF file is based on the name of the coverage file.
(3) The date of creation is included in the filename
(ex: Line_456.coverage -> Line_456_20150101.simplified_VCF)

Chapter 5

Generating consensus with Gap.Jumper.plus

5.1 GapJumper.plus

GapJumper.plus that is a part of GapJumper v.01 is one R script called GapJumper_03_GapJumperPlus.R. GapJumper_03_GapJumperPlus.R contains all functions required to build a consensus with each of six different methods (Chapter 2). Moreover it contains additional functions that are use to treat SNV calling data in consensus build with on of the two probabilistic methods. They are described in Chapter 6.

5.2 Requirements

In order to build a consensus Gap.Jumper.plus requires:

(1) R version 3.1.1 (or newer)
(2) GapJumper_03_GapJumperPlus.R
(1) .simplified_VCF for each replicate (input data)
(2) Rg_Table that specifies, which replicates are used to construct a given consensus and what is the name of that consensus (Chapter 8)

5.3 Before you start

5.3.1 Download from GitHub

(1) GapJumper _v01	(it contains *GapJumper_03_GapJumperPlus.R*)
(2) *GapJumper_Example_Two*	(.simplified_VCF files and Rg_Table)
(3) *GapJumper_Example_Two_Results*	(.multiSNV files)
(4) *GapJumper_Example_Three*	(additional .simplified_VCF and Rg_Table)

5.3.2 Create New directories (A or B)

A. Four directories, each for different files:

(1) *GapJumperPlus_data_dir*	(to store .simplified_VCF files – input data)
(2) *GapJumperPlus_working_dir*	(to store intermediate files)
(3) *GapJumperPlus_results_dir*	(to store .multiSNV files)
(4) *GapJumperPlus_Rg_table_dir*	(to store RG_Table)

B. One directory, for all type of files:

(-) *GapJumperPlus_dir*	(to store .all type of files)

and use it instead of each file that were mentioned in A.

Comments:

We recommend using four different directories in this tutorial. It is also possible to use other names of files, because only path to directory is important and it must be define at each time Gap.Jumper.plus is used (see later in the text).

5.3.3 Acquire input data (.simplified_VCF files)

A. Possibility_1

Tansfer .simplified_VCF (9 of them) files from *GapJumper_Example_Two* directory to *GapJumperPlus_data_dir*

B. Possibility_2

Use .simplified_VCF files that were generated in Chapter 4
Copy .simplified_VCF files from *VCF_Converter_working_dir*
to *GapJumperPlus_data_dir*

Comments:

We recommend to copy .simplified_VCF files to new directory only for the purpose of this tutorial. It is possible to use .simplified_VCF files directly in *VCF_Converter_working_dir,*
`R > vcf.dir <-"~/PATH` *VCF_Converter_working_dir.* `"` (see later in the text)

5.3.4 Copy Rg_Table

From *GapJumper_Example_Two* **to** *GapJumperPlus_Rg_table_dir*
Use *GapJumper_Example_Two_Rg_Table.csv*

5.4 Loading GapJumper.plus

```
R >   setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
R >   source("GapJumper_03_GapJumperPlus.R")
```

GapJumper.plus was build under R version 3.1.1.

5.5 R functions used to construct a consensus

Gap.Jumper.plus constructs a consensus using one of six different methods. Each of these methods is implemented in separate R function, except for the All Recoded Nucleotides method that is used by each function in addition to each of the other five methods (Chapter 2). For this reason, each R function generates two different .multiSNV files for each set of replicates. One with a consensus constructed using All recorded Nucleotides method and the second file with consensus constructed using a given method. These functions are:

(1) Run.Probabilistic.method() (Probabilistic m. + All. Recorded Nucleotides m.)
(2) Run.Probabilistic.beta.method() (Probabilistic beta m. + All. Recorded Nucleotides m.)
(3) Run.Conservative.method() (Conservative m. + All. Recorded Nucleotides m.)
(4) Run.Site.Elimination.method() (Site Elimination m. + All. Recorded Nucleotides m.)
(5) Run.Fixed.Thresholds.method() (Fixed Thresholds m. + All. Recorded Nucleotides m.)

Additionally, Gap.jumper.plus has another function called Run.All.methods(), that allows constructing of a consensus for each set of related replicates with each of the six different methods at the same time.

5.6 Arguments used by consensus building functions

Each of R functions implemented in Gap.Jumper.plus requires several arguments in order to build a consensus, such as project name, input/output directory etc… Some of these arguments are the same in each function and some of them are specific for a given method (Table 1).

5.6.1 Project name (prj.n)

Project name is used to identify intermediate files that are generated with Gap.Jumper.plus and stored in *GapJumperPlus_working_dir*. Project name should be unique and long enough so it is not a part of name of any other file that can be found in *GapJumperPlus_working_dir*. For more information see examples in Chapter 11. Unique project name allows storing intermediate files generated by Gap.Jumper with many projects in one directory.

```
R >   prj.n <-"GapJumper_Tutorial_project"
```
(Or a name that you wish to put here)

5.6.2 Rg_table name (tab.n)

Full name of Rg_Table, that is used in a given project. This table specify which of different replicates are used to generate each consensus. For more information about Rg_Table see Chapter 8. Rg_Table name should be unique and long enough so it is not a part of name of any other file that can be found in *GapJumperPlus_Rg_table_dir*.

```
R >   tab.n <-" GapJumper_Example_Two_Rg_Table.csv"
```
(FULL NAME !, NO PATH !)

5.6.3 Specify Working Directories

A. In case you use four different directories:

```
R >   vcf.dir <-"~/PATH GapJumperPlus_data_dir"
R >   int.dir <-"~/PATH GapJumperPlus_working_dir"
R >   res.dir <-"~/PATH GapJumperPlus_results_dir"
R >   tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"
```

B. In case you use only one directory

```
R >   vcf.dir <-"~/PATH GapJumperPlus_dir"
R >   int.dir<-vcf.dir; res.dir<-vcf.dir; tab.dir<-vcf.dir # end
```

5.6.4 Positions to Use (pos.y)

Gap.Jumper.plus gives the possibility to perform analysis on the subset of positions, e.g. to quickly test new pipeline or to find the most optimal method.

```
Vector          with 0 or 1
Length          == Number of positions in .simplified_VCF files

If empty:       R >    pos.y <-"n" # no action
If used:        R >    pos.y <-as.numeric(c(rep(1,1000),rep(0,4000)))
```

Where: 1 == Position will be USED in constructing a consensus
to use only first 1000 positions out of 5000 positions
in each .simplified_VCF file

5.6.5 Positions to Exclude (pos.n)

In order to exclude some of the positions in each simplified_VCF, e.g. in case when some of the position potentially are in mitochondrial genome, when analyzing the polymorphisms in nucleolar genome. Furthermore, excluded positions are not used to construct a consensus, even if pos.y was equal to 1 at these positions.

```
Vector        with 0 or 1
Length        == Number of positions in .simplified_VCF files

If empty:     R >    pos.n <-"n" # no action
If used:      R >    pos.n <-as.numeric(c(rep(1,100),rep(0,4900)))
```

Where: 1 == Position will be EXCLUDED
to exclude first 80 positions out of 5000 positions in each .simplified_VCF file

5.6.6 Group of positions – "classifier z" (pos.z)

This argument is optional and it is required be for the two probabilistic methods (Table 1). It is a numerical vector with length equal to the number of positions in each .simplified_VCF file. It allows treating positions as two separate groups in order to estimate the parameters that are required to calculate probability for each nucleotide. For more information see Rosikiewicz et al. (2016)

```
Vector        with 0 or 1
Length        == Number of positions in .simplified_VCF files

If empty:     R >    pos.y <-"n" # no action
If used:      R >    pos.y <-c(rep(1,2500),rep(0,2500))
```

Where: 1 == First group of positions
0 == Second group of positions

5.6.7 Detection Limit (*gp.dl*)

The detection limit is set in order to ensure that sequence data have sufficient coverage at each position to allow the detection of all the possible SNV in each replicate. Typically, the *gp.dl* is equal to 10, although it may depend on the ploidy level of the organism, the number of pooled individuals in each sample and the expected frequency of rare mutations in each replicate.

```
R >      gp.dl <-as.numeric(10)
```

IMPORTANT:
Detection Limit must be equal to, or higher than, the coverage value that was used to filter .vcf files. Lower values may give incorrect results !

5.6.8 Prior p (*gp.p*)

Gap.Jumper uses two probabilistic methods in order to construct a consensus. It calculates the probability for each nucleotide at each position, that this nucleotide is a true nucleotide, using SNV calling from many replicates. Prior p is a parameter that is required to calculate this probability for each nucleotide. In order to accurately calculate the probability for each nucleotide, Prior *p* value should be equal to, or lower than, the least expected relative frequency of a nucleotide at each position in each replicate. For more information see Rosikiewicz et al. (2016).

```
R >      gp.p <-as.numeric(0.2);
```
(This is a value required in our example)

5.6.9 Arguments required for Site Elimination method (se.na…)

In order to construct a consensus using Site Elimination method (Run.Site.Elimination.method()), GapJumper requires two arguments that describe the amount of missing data at each position in different replicates that is sufficient to exclude this position from using in constructing a consensus. Gap.Jumper applies the lower for each set of replicates.

A. se.na.nr – the number of replicates with missing data
that is sufficient to exclude a given position

R > se.na.nr <-as.numeric(2);
Exclude all positions with missing data in 2 or more replicates (Figure 4)

B. se.na.rt – the percentage of replicates with missing data
that is sufficient to exclude a given position (between 0 to 1)

R > se.na.rt <-as.numeric(0.5);
Exclude all positions with missing data in 50% or more replicates (Figure 4)

Comments:

(1) In case you would like to use only **se.na.nr, define se.na.rt <-0.000001**
(2) In case you would like to use only **se.na.rt, define se.na.nr <-1000000**
(3) In case, when bot values are given, the lower value is applied

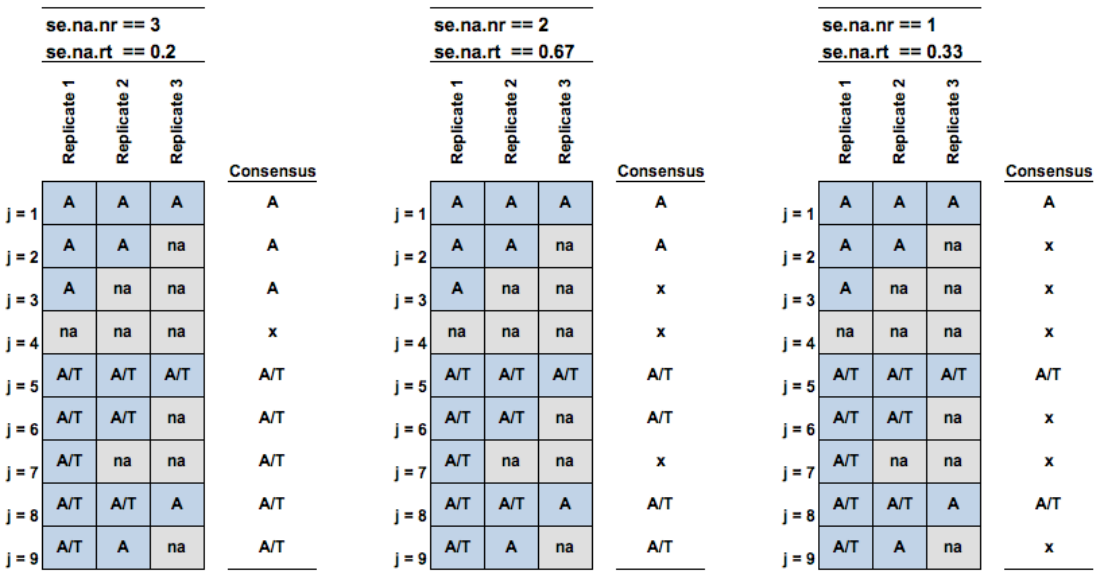


Figure 4. Difference in a consensus build with Gap.Jumper, using Site Elimination method with its arguments (se.na.nr and se.na.rt) set for different values. Each consensus depicts nine positions (j).

5.6.10 Arguments required for Fixed Thresholds method (ft…)

In order to construct a consensus using Fixed Thresholds method it is necessary to define three different thresholds. These are:

A. ft.na.rt – Minimal percentage of replicates without missing data (na) at a given postion In order to use this position in constructing a consensus (Figure 5). (It is not the same as se.na.nr !)

R > ft.na.rt <-as.numeric(0.5); # value: 0< ft.na.rt≤1

Use positions that have coverage ≥ detection limit in at least 50% of replicates

B. ft.nr – Minimal number of replicates with a given nucleotide at a given position in order to use this nucleotide in constructing a consensus (Figure 6).

R > ft.nr <-as.numeric(2); # value: fr.nr≥1

Use nucleotides, which were present in at least 2 replicates at a given position

C. ft.ap – Minimal relative frequency of a nucleotide at a given position in a given replicate in order to use that nucleotide in constructing a consensus (Figure 7).

R > ft.ap <-as.numeric(0.5); # value: 0< ft.ap≤1

Use nucleotide that constitute minimum 50% of reads (or more) at a given position in a given replicate

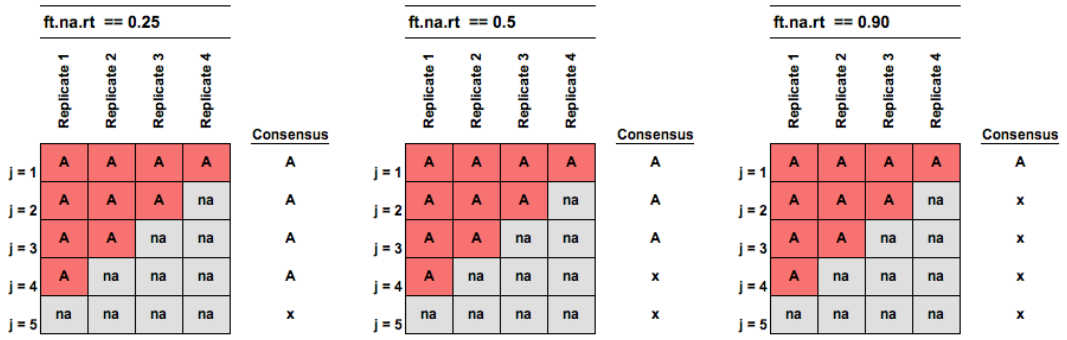


Figure 5. Difference in a consensus build with Gap.Jumper.plus using Fixed Threshold method with applying different values of ft.na.rt threshold.

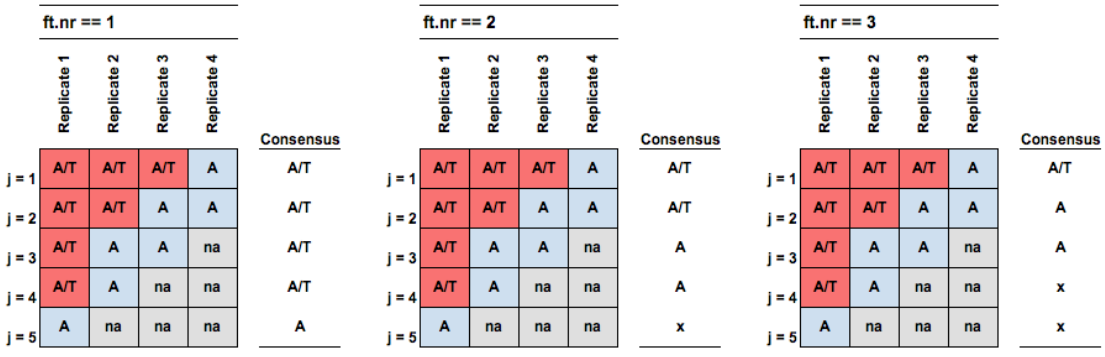


Figure 6. Difference in a consensus build with Gap.Jumper.plus using Fixed Threshold method with applying different values of ft.nr threshold.

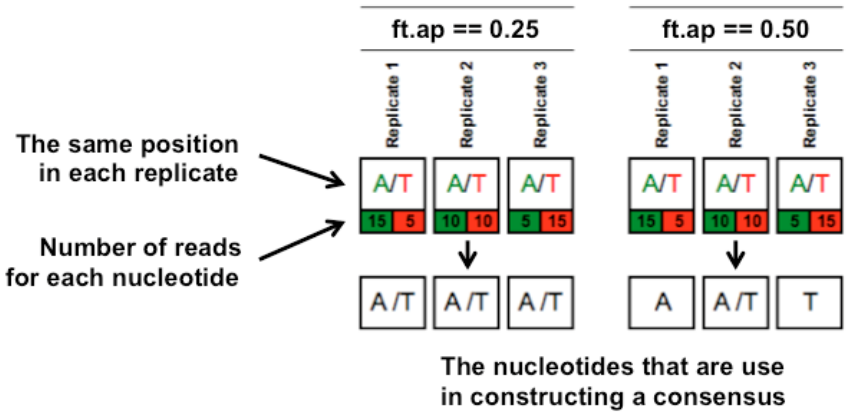


Figure 7. The results of applying ft.ap threshold with different values to SNV calling data at one position with two nucleotides (A and T) in three different replicates.

Table 1. Arguments which are required (yes) or not required (x) in order to construct a consensus with each of six different methods using R functions implemented in Gap.Jumper.plus (see 5.5)

Argument		Value	Run.Probabilistic.method()	Run.Probabilistic.beta.method()	Run.Conservative.method()	Run.Site.Elimination.method()	Run.Fixed.Thresholds.method()	Run.All.methods()
prj.n	Project name	"Project_name"	yes	yes	yes	yes	yes	yes
tab.n	Rg Table name	"File_name.csv"	yes	yes	yes	yes	yes	yes
vcf.dir	GapJumperPlus_data_dir	"PATH"	yes	yes	yes	yes	yes	yes
tab.dir	GapJumperPlus_Rg_table_dir	"PATH"	yes	yes	yes	yes	yes	yes
int.dir	GapJumperPlus_working_dir	"PATH"	yes	yes	yes	yes	yes	yes
res.dir	GapJumperPlus_results_dir	"PATH"	yes	yes	yes	yes	yes	yes
pos.y	Positions to use	"n" or vector {0,1}; length == number of positions	yes	yes	yes	yes	yes	yes
pos.n	Positions to exclude	"n" or vector {0,1}; length == number of positions	yes	yes	yes	yes	yes	yes
pos.z	Classifier z	"n" or vector {0,1}; length == number of positions	yes	yes	x	x	x	yes
gp.dl	Detection limit	gp.dl ≥ 1	yes	yes	yes	yes	yes	yes
gp.p	Prior p	0 < gp.p <1	yes	yes	x	x	x	yes
se.na.nr	Number of replicates with missing data	se.na.nr ≥ 1	x	x	x	yes	x	yes
se.na.rt	Percentage of replicates with missing data	0 < se.na.rt <1	x	x	x	yes	x	yes
ft.na.rt	Minimal percentage of replicates without missing data	0 < ft.na.rt <1	x	x	x	x	yes	yes
ft.nr	Minimal number of replicates with a given nucleotide at a given position	ft.nr ≥ 1	x	x	x	x	yes	yes
ft.ap	Minimal relative frequency of a nucleotide at a given position	0 < ft.ap <1	x	x	x	x	yes	yes

Chapter 6

Noise Filtration

6.1 Removing the noise

Each consensus that is generated using probabilistic method or using probabilistic beta method requires noise filtration. The probabilities that were calculated for nucleotides at the postions wre these nucleotides were not present in the sequence data should be removed. Gap.Jumper.plus has implemented two functions that can be use in order to remove these values from the consensus. These are:

(1) Remove.Noise.Global()
(2) Remove.Noise.using.Noise.Threshold()

6.2 Remove.Noise.Global()

This function removes probabilities of all nucleotides that are not present in a corresponding consensus constructed with All Recorded Nucleotides method for the same set of replicates. Additionally, it generates column number 15, with positions that have missing data and a column number 16 with presence of each nucleotide in a consensus in each new filtered consensus (Chapter 10). New consensus is saved in the same directory as the original one. It has new file extension: "NsFiltered.multiSNV". The function filters automatically each consensus that present is a given directory and that has a corresponding consensus constructed using All Recorded Nucleotides in the same directory.

```
R > res.dir <-"~/PATH GapJumperPlus_results_dir"
R > Remove.Noise.Global(res.dir)
```

6.3 Remove.Noise.using.Noise.Threshold()

This function removes probabilities calculated for each nucleotide at each position that are below certain value called the noise threshold. This function was implemented because some of the nucleotides that are present in the All Recorded Positions may be sequence errors and they are should be removed. The function filter automatically each consensus that is present in a given directory.

```
R > res.dir <-"~/PATH GapJumperPlus_results_dir"
R > Ns.Tr    <-as.numeric(0.05)
R > Remove.Noise.using.Noise.Threshold(res.dir,NsTr)
```

It also generates column number 15, with positions that have missing data and a column number 16 with presence of each nucleotide in a consensus in each new filtered consensus (Chapter 10). New consensus is saved in the same directory as the original one. It has new file extension: "NsFiltered.multiSNV".

6.4 Calculating an optimal noise threshold

Because some of the nucleotides potentially are sequencing errors, it is possible to calculate the noise thresholds that can correspond to this uncertainty and to remove these nucleotides from the consensus using **Remove.Noise.using.Noise.Threshold()** fucntion. For this purpose we implemented Calculate.Noise.threshold() function in Gap.Jumper.plus. It works with one consensus at a time and finds the most optimal noise level using the probabilities found for each nucleotide in that consensus and the error rate in sequencing.

```
R > res.dir                          <-"~/PATH GapJumperPlus_results_dir"
R > prob.multiSNV.name               <-"The name of .multiSNV file
                                       constructed using probabilistic method"
R > all.Rec.nucl.multiSNV.name       <-"The name of a corresponding .multiSNV file
                                       constructed using All Recorded Nucletides
                                       method"
R > retain.nucleotides               <-"the percentage of nucletides that are most
                                       probably not error"
Run:
R > Calculate.Noise.threshold(res.dir,prob.multiSNV.name,
                              all.Rec.nucl.multiSNV.name, retain.nucleotides)
```

Comments:

```
retain.nucleotides <- 0.999 # when the estimated error rate is 0.001
```

6.5 Testing noise threshold

In order to test how many nucleotides is going to be removed from the consensus constructed using one of the two probabilistic methods.

```
R > res.dir                          <-"~/PATH GapJumperPlus_results_dir"
R > prob.multiSNV.name               <-"The name of .multiSNV file
                                       constructed using probabilistic method"
R > all.Rec.nucl.multiSNV.name       <-"The name of a corresponding .multiSNV file
                                       constructed using All Recorded Nucletides
                                       method"
R > Noise.Threshold                  <-as.numeric(0.01) # EXAMPLE
Run:
R > est.Noise.threshold(res.dir,prob.multiSNV.name,
                              all.Rec.nucl.multiSNV.name, retain.nucleotides)
```

See Chapter 7

For examples of using R function that were described in this Chapter

Chapter 7

R code examples for each method

7.1 Probabilistic method

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Two"

# Rg table name
  tab.n              <-"GapJumper_Example_Two_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions
  pos.z    <-"n"    # treat all positions in the same way

# detection limit
  gp.dl    <-as.numeric(10)

# prior p
  gp.p     <-as.numeric(0.2)

# generate consensus (.multiSNV)
  Run.Probabilistic.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                            pos.y,pos.n,pos.z,gp.dl,gp.p)

# Remove the noise - globally
  Remove.Noise.Global(res.dir)

# calculate optimal noise
  prob.multiSNV.name          <-"Sample_01__Probabilistic_method.multiSNV"
  all.Rec.nucl.multiSNV.name  <-"Sample_01__AllRecorded_Nucleotides.multiSNV"
  retain.nucleotides          <-0.99
  ns                          <-Calculate.Noise.threshold(res.dir,prob.multiSNV.name,
                                all.Rec.nucl.multiSNV.name,retain.nucleotides)


# test it on another consensus
  prob.multiSNV.name          <-"Sample_02__Probabilistic_method.multiSNV"
  all.Rec.nucl.multiSNV.name  <-"Sample_02__AllRecorded_Nucleotides.multiSNV"
  Noise.Threshold             <-ns
  Test.Noise.threshold(res.dir,prob.multiSNV.name,all.Rec.nucl.multiSNV.name,Noise.Threshold)

# use it to remove the noise
# (it will overwrite other filtered files !)
  Remove.Noise.using.Noise.Threshold(res.dir,ns)
```

7.2 Probabilistic beta method

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Two"

# Rg table name
  tab.n              <-"GapJumper_Example_Two_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions
  pos.z    <-"n"    # treat all positions in the same way

# detection limit
  gp.dl    <-as.numeric(10)

# prior p
  gp.p     <-as.numeric(0.2)

# generate consensus (.multiSNV)
  Run.Probabilistic.beta.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                                 pos.y,pos.n,pos.z,gp.dl,gp.p)

# Remove the noise - globally
  Remove.Noise.Global(res.dir)

# calculate optimal noise
  prob.multiSNV.name          <-"Sample_01__Probabilistic_beta_method.multiSNV"
  all.Rec.nucl.multiSNV.name  <-"Sample_01__AllRecorded_Nucleotides.multiSNV"
  retain.nucleotides          <-0.99
  ns                          <-Calculate.Noise.threshold(res.dir,prob.multiSNV.name,
                                all.Rec.nucl.multiSNV.name,retain.nucleotides)


# test it on another consensus
  prob.multiSNV.name          <-"Sample_02__Probabilistic_beta_method.multiSNV"
  all.Rec.nucl.multiSNV.name  <-"Sample_02__AllRecorded_Nucleotides.multiSNV"
  Noise.Threshold             <-ns
  Test.Noise.threshold(res.dir,prob.multiSNV.name,all.Rec.nucl.multiSNV.name,Noise.Threshold)

# use it to remove the noise
# (it will overwrite other filtered files !)
  Remove.Noise.using.Noise.Threshold(res.dir,ns)
```

7.3 Conservative method

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Two"

# Rg table name
  tab.n              <-"GapJumper_Example_Two_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions

# detection limit
  gp.dl    <-as.numeric(10)

# generate consensus (.multiSNV)
  Run.Conservative.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                           pos.y,pos.n,gp.dl)
```

7.4 Site Elimination method

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Two"

# Rg table name
  tab.n              <-"GapJumper_Example_Two_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions

# detection limit Arguments used for Site Elimination method
  gp.dl    <-as.numeric(10)

# detection limit Arguments used for Site Elimination method
  se.na.nr  <-as.numeric(2)
  se.na.rt  <-as.numeric(0.5)

# generate consensus (.multiSNV)
  Run.Conservative.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                           pos.y,pos.n,gp.dl,se.na.nr,se,na.rt)
```

7.5 Fixed Thresholds method

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Two"

# Rg table name
  tab.n              <-"GapJumper_Example_Two_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions

# detection limit Arguments used for Site Elimination method
  gp.dl    <-as.numeric(10)

# detection limit Arguments used for Site Elimination method
  ft.na.rt <-as.numeric(0.5)
  ft.ap    <-as.numeric(0.5)
  ft.nr    <- as.numeric(2)

# generate consensus (.multiSNV)
  Run.Conservative.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                           pos.y,pos.n,gp.dl,ft,na.rt,ft.nr,ft.ap)
```

7.6 Probabilistic method on a large number of samples

Use .simplified_VCF files and Rg_table from GapJumper_Example_Three folder

```
# load Gap.Jumper.plus
  setwd("~/PATH TO DIRECTORY THAT CONTAINS GapJumper_03_GapJumperPlus.R")
  source("GapJumper_03_GapJumperPlus.R")

# project name
  prj.n              <-"GapJumper_Test_Run_On_Example_Three"

# Rg table name
  tab.n              <-"GapJumper_Example_Three_Rg_Table.csv"

# directories
  vcf.dir <-"~/PATH GapJumperPlus_data_dir"
  int.dir <-"~/PATH GapJumperPlus_working_dir"
  res.dir <-"~/PATH GapJumperPlus_results_dir"
  tab.dir <-"~/PATH GapJumperPlus_Rg_table_dir"

# positions
  pos.y    <-"n"    # use all positions
  pos.n    <-"n"    # no excluded positions
  pos.z    <-"n"    # treat all positions in the same way

# detection limit
  gp.dl    <-as.numeric(10)

# prior p
  gp.p     <-as.numeric(0.2)

# generate consensus (.multiSNV)
  Run.Probabilistic.method( prj.n,tab.n,vcf.dir,int.dir,res.dir,tab.dir,
                            pos.y,pos.n,pos.z,gp.dl,gp.p)

# Remove the noise - globally
  Remove.Noise.Global(res.dir)
```

This example shows that each function can generate consensus for many different samples. Each sample has different number of replicates (1 to 6). Moreover some of replicates were used to build two different consensuses.

Chapter 8

Rg_Table

8.1 General information

Rg_Table defines which of replicates are used by Gap.Jumper.plus in order to generate each consensus. It has two columns and the header. It is saved in .csv format.

8.2 First column ("Replicate_name")

First column contains the name of each replicate in each set of replicates (e.g Replicate_01, Replicate_02 etc…) that are used to construct a consensus (Table 2). GapJumper.plus requires, full names of each replicate or it sufficient part that can uniquely identify each .simplified_VCF file. It is possible to use name of each replicate many times.

8.3 Second Column ("Consensus_name")

The second column of Rg_Table contains the name of each consensus that should be generated using replicates in the first column. Name of each consensus should be given without extensions (i.e .multiSNV) and it must be unique.

Table 2. An example of Rg_Table

Replicate_name	Consensus_name
Sample01_Replicate01.simplifiedVCF	Sample_01
Sample01_Replicate02.simplifiedVCF	Sample_01
Sample01_Replicate03.simplifiedVCF	Sample_01
Sample02_Replicate01.simplifiedVCF	Sample_02
Sample02_Replicate02.simplifiedVCF	Sample_02
Sample02_Replicate03.simplifiedVCF	Sample_02

Comments:

(1) It is possible to generate several consensuses at the same time and each of them can have different number of replicates. However, each of replicates that are used together needs to be prepared using the same *All_positions_with_variants201XXXXX.combinedVCF* file. It also means that all these replicates have the same number of positions in each .simplified_VCF file.

(2) Because, Gap.Jumper.plus uses only replicates that are specified in Rg_Table, all *.simplified_VCF* files can be stored together and use in different projects and in different combinations.

(3) Gap.Jumper.plus constructs a consensus with one of six different methods.
Each of them has its own extension that is added to the consensus name
e.g *Sample_01__Conservative_method.multiVCF*

Chapter 9

.simplified_VCF file format

Each row in .simplified VCF file corresponds to one position in the reference genome that was different than the reference in at least one replicate. Each .simplified_VCF file contains the following columns:

(1) Replicate_name: corresponds to the name of the sample and the replicate.

(2) Scaffold: is the scaffold, contig or chromosome in the reference genome.

(3) Locus: contains additional positional information like a particular region, a gene or a locus identified previously. It contains "na" if nothing was specified.

(4) Position: reports the position in the scaffold, contig or chromosome.

(5) seqID: is another optional information. It contains "na" if nothing was specified.

(6-8) x1 – x3: can be filled with information about coding or repeated status of the position. It contains "na" if nothing was specified.

(9) Ref: is the base found at this position in the reference genome

(10) Type_Alleles: describes the types of the recorded alleles. It can be "I" for "insertion", "D" for "deletion", "S" for "substitution", "M" for "multiple nucleotide polymorphism" or "R" for "reference allele". If more than one type was found, they are all reported with comma as separator. The sorting is done in order to match the content in the next columns. If no alleles different from the reference genome were detected, the cell remains empty.

(11) Alleles: reports the alleles accordingly to the Type_alleles column. Deletion are defined based on the last base remaining before the missing sequence. The last base is completed with a tiret and a number. The number corresponds to the number of missing bases. Ex:

```
ATGCTTGAA
AT-----AA     =>     T-5        (i.e. T is here but the next 5 bases are missing)
```

(12) Occurrence_Alleles: reports the allele occurrences accordingly to the 2 previous columns.

(13) General_coverage: is an indication of the general coverage as measured with 'samtools depth'. But the general coverage is not as precise as the measured allele occurrence.

Chapter 10

.multiSNV file format

Each row in .multiVCF file corresponds to one position in the reference genome that was different than the reference in at least one replicate. Each .multiSNV file contains the following columns:

(1) **Consensus_name**: Taken form Rg_Table

(2 – 9) **taken from .simplified_VCF files**, that were used to construct each consensus **(Chapter 9)**

(10-13) **Pr(A), Pr(C), Pr(G), Pr(T)**
The presence of each nucleotide at each position shown as 0 or 1, respectively or as probability [0,1] that is calculated using one of the two probabilistic methods.

(14) **"na":** binary vector showing positions with missing data (1) in the consensus

(15) **snp.presence:** reports nucleotides that are present in the consensus {A,C,G,T} or missing data {na} at each position

(16) **snp.presence.in.each.replicate:** report nucleotides that were present in .simplified_VCF used to construct a given nucleotides in the column 11 (Alleles). Data from different replicates are separated with ";". Positions in replicates with missing data are shown as "-", whereas positions in replicates with no coverage as "z". e.g: "A;A;z;-" and "C/T;C/T;C" Nucleotides are ordered alphabetically in each replicate.

(17) **snp.freq.in.each.replicate:** reports the nucleotide occurrences in each replicate according to the previous column. "-" denotes missing data and "z" denotes no coverage. In case more than one nucleotide is present, the order of appearance in each replicate is the same.
e.g. (16)"A/T;A/T;A/T" and (17)"5/10;5/10;5/10" where each replicates has 5 reads for A and 10 reads for T at that position.

(18) **position.coverage.in.each.replicate:** reports General_coverage from each simplified_VCF file.

(19) **InDel:** reports presence ("y") or absence ("n") of InDels at each position in any of the replicates that were used to construct a given consensus

(20) **Nr.of.pooled.replicates:** Number of replicates that were used to construct a given consensus

Examples of consensus saved in .multiVCF format can be found in *GapJumper_Example_Two_Results* folder.

Chapter 11

When making names for files:

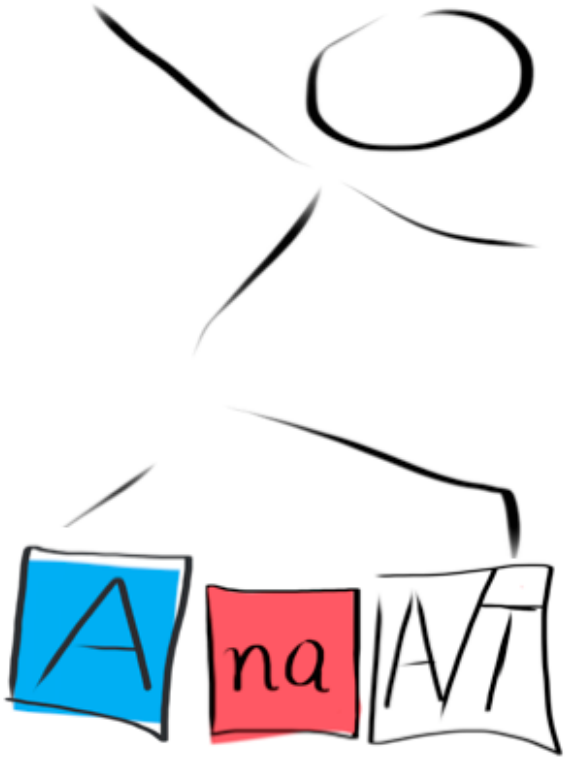
11.1 Do not use “__”, “/”, ”|”, tab, ”.xxx.xxx.”, etc..

11.2 It is allowed to use “-“, “_” ,
but no more than one in a row

11.3 The name of each file can not be a part
of the name of another file that is being loaded from the same directory:

sample_01	and	sample_01_bis	etc…
repl_1	and	repl_11, repl_12	etc..

11.4 Do not start the name of a file with a number.



THE END