Gap.Jumper v.02

Manual and Tutorial

Pawel Rosikiewicz[1], Frédéric G Masclaux[1,2], Tania Wyss[1], Marco Pagni[2], Ian R. Sanders[1]

[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

October 25, 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. The consensus contains all the information about that sample. The replicates can be independent biological replicates or technical replicates. Each replicate should be sequenced and genotyped separately (Figure 1).

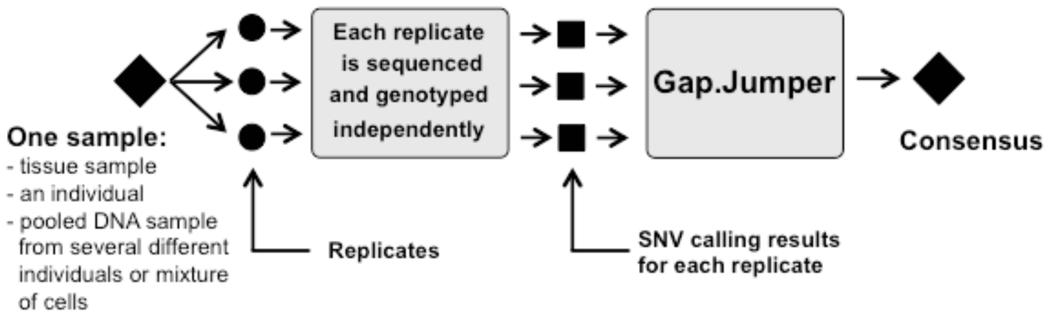

Each replicate is sequenced and genotyped independently
Gap.Jumper
One sample:
- tissue sample
- an individual
- pooled DNA sample from several different individuals or mixture of cells
Replicates
SNV calling results for each replicate
Consensus

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 replicates may be contaminated by 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.

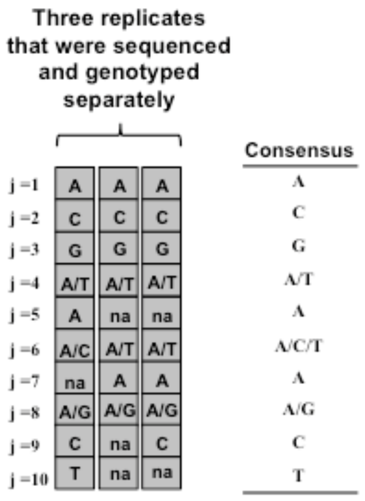


Figure 2. The schematic diagram showing the construction of a consensus from SNV calling data at ten different positions (j) in three different replicates. The consensus was constructed using the All Recorded Nucleotides method (Chapter 2)

1.3 Construction of a consensus

SNV calling data from many replicates can be used in many different ways to construct a consensus. It depends on the 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 (for example, whether the consensus is built for one individual or a population). For this reason, Gap.Jumper v.01 allows the user to construct a consensus from many replicates using six different methods that are explained 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 the 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, Gap.Jumper comes with three sets of example files that are used in the Gap.Jumper tutorial (Chapters 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:

https://github.com/GapJumperPlus/GapJumper_v02

The example files can be downloaded from:

https://www.dropbox.com/sh/gmeoddqn6tjtwxc/AACJ-pnh3gzbhGtqLQ35wVMVa?dl=0

1.6 Licence

Gap.Jumper can be distributed, copied and modified without any charge under the terms of the 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 by saying whether you have problems

with VCF.Converter or Gap.Jumper.plus, and which version 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. (submitted). Schematic diagram with comparison of these methods was depicted on Figure 3. Furthermore, an example of a consensus constructed using each of these six methods is given in *GapJumper_Example_Two_Results* folder on GitHub (.multiSNV files). The extension in each .multiSNV file in this folder corresponds to the name of the method that was used to construct that consensus. In addition to six different methods, Gap.Jumper allows combining the consensus build with different methods (For more information see Chapter 8).

2.1 Probabilistic method

The probabilistic method constructs a consensus in which each nucleotide at each position obtains a quality score ranging from zero to one. The quality scores expresses the probability that a given nucleotide is a true nucleotide (Figure 3). Additionally, each position obtains general quality score, based on quality scores of nucleotides recoded at that position. Qaulity scores are calculated with the procedure described in detail in Rosikiewicz et al. (submitted).

2.2 Probabilistic beta method

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

2.3 Conservative method

The consensus contains only the information at positions that have exactly the same nucleotide composition in all replicates (Figure 3).

2.4 Site Elimination method

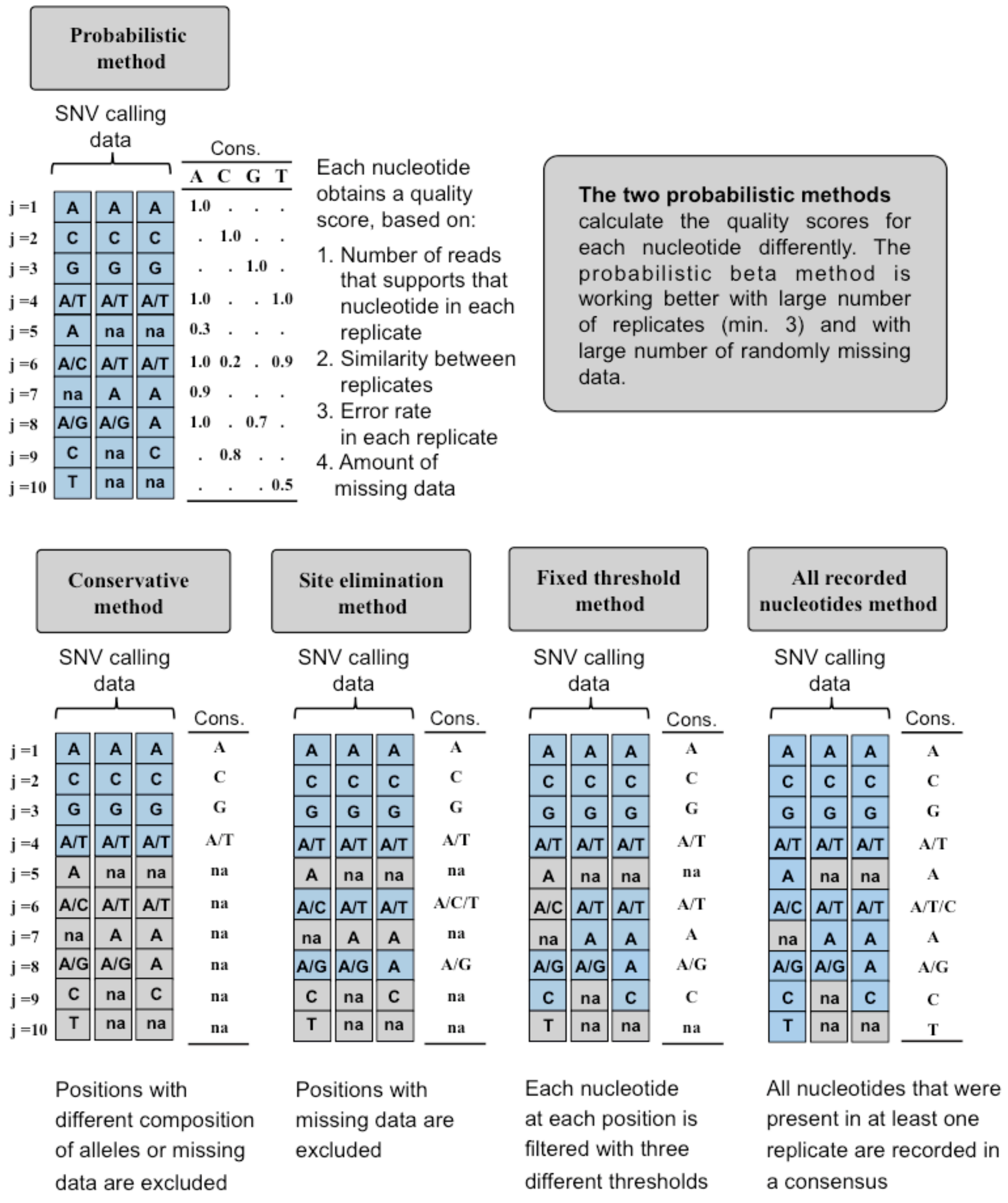
A position is removed if it contains missing data in at least some of the replicates (Figure 3).
For more information see Chapter 5.6.9.

2.5 Fixed thresholds method

Three different thresholds describe the minimal acceptance criteria for each nucleotide and each position (Figure 3). These are: (i) the relative frequency of each nucleotide, (ii) the number of different replicates with the same allele at the same position and (iii) the amount of missing information at that position. For more information see Chapter 5.6.10

2.6 All Recorded Nucleotides method

The All Recorded Nucleotides method constructs 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 (Figure 3). Gap.Jumper always builds a consensus using this method, irrespective which other method was.



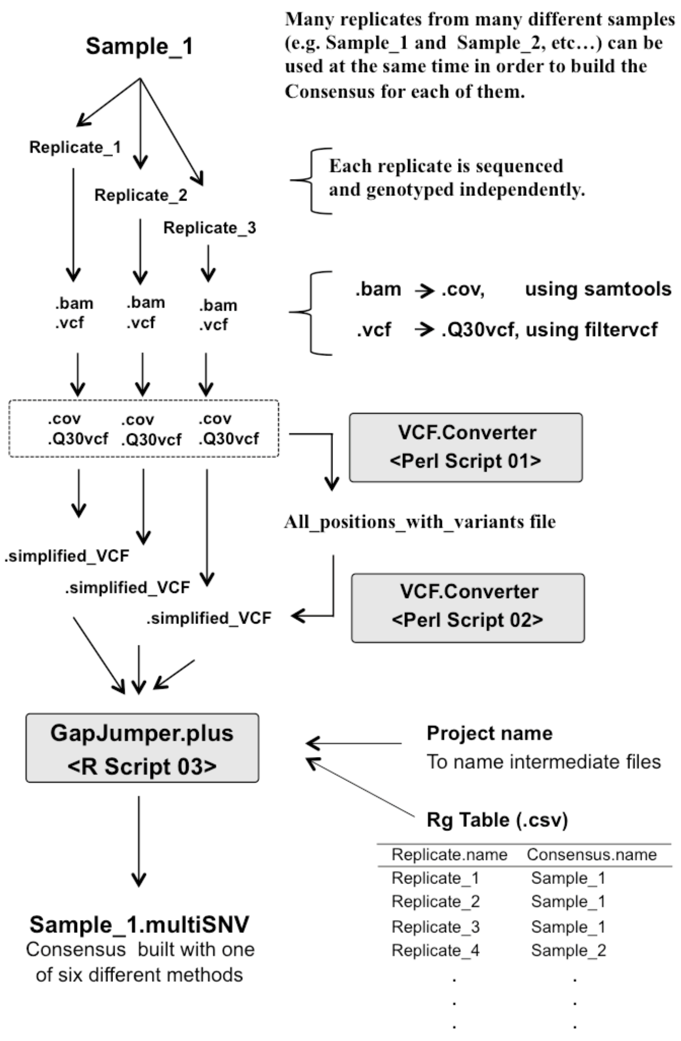
Probabilistic method
SNV calling data
Cons.
Each nucleotide obtains a quality score, based on:
1. Number of reads that supports that nucleotide in each replicate
2. Similarity between replicates
3. Error rate in each replicate
4. Amount of missing data
The two probabilistic methods calculate the quality scores for each nucleotide differently. The probabilistic beta method is working better with large number of replicates (min. 3) and with large number of randomly missing data.
Conservative method
Site elimination method
Fixed threshold method
All recorded nucleotides method
Positions with different composition of alleles or missing data are excluded
Positions with missing data are excluded
Each nucleotide at each position is filtered with three different thresholds
All nucleotides that were present in at least one replicate are recorded in a consensus

Figure 3. Different methods used to build a consensus (cons.) implemented in Gap.Jumper. SNV data represent ten positions (j) in three different replicates that were independently sequenced. na denotes missing data.

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 4). In Chapter 4, we describe how to prepare these files and how to generate simplified_VCF files using VCF.Converter. Subsequently, the Gap.Jumper.plus module generates a .multiSNV file for each set of replicates. The .multiSNV file contains a consensus built with one of six different methods (Figure 4). The R functions that were implemented in the 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 5-9.


Sample_1
Many replicates from many different samples (e.g. Sample_1 and Sample_2, etc…) can be used at the same time in order to build the Consensus for each of them.
Replicate_1
Replicate_2
Replicate_3
Each replicate is sequenced and genotyped independently.
.bam .vcf
.bam → .cov, using samtools
.vcf → .Q30vcf, using filtervcf
.cov .Q30vcf
VCF.Converter <Perl Script 01>
All_positions_with_variants file
.simplified_VCF
VCF.Converter <Perl Script 02>
GapJumper.plus <R Script 03>
Project name
To name intermediate files
Rg Table (.csv)
Sample_1.multiSNV
Consensus built with one of six different methods

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

Figure 4. Schematic diagram showing the pathway to construct a consensus from several replicates of 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 mapped to the same reference genome and treated together with VCF.Converter (i.e. built 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 a .simplified_VCF file for each replicate. A simplified_VCF is an alternatively structured .vcf file, which clearly shows the allelic composition of each genomic position. It is particularly easy to know if the reference allele is present and what is the coverage at each position. Additionally, VCF.Converter allows Gap.Jupmer to use .vcf files generated with several different softwares, such as GATK and Frebayes (see later in the text).

4.2 VCF.Converter

VCF.Converter. is composed of two Perl scripts:

(1) GapJumper_01_Record_all_positions_from_VCF.pl (Perl script 01; Figure 4)
(2) GapJumper_02_Simplified_VCF_maker.pl (Perl script 02; Figure 4)

The first script analyzes all the .vcf files together and generates a list of all positions where either SNV or multi nucleotide variants (MNV) or insertions and deletions were detected in at least one of the .vcf files (Figure 4). The second script converts each .vcf file and .cov file into a .simplified_VCF file where variant information is reported at every position from a list of positions (Figure 4). If a variant was detected at a recorded position, it is simply reported. If no variant was found and the coverage at this position was sufficient, the nucleotide from the reference genome is reported. If the coverage is insufficient, a missing value is reported (na). Additionally, the coverage, type of variant and the number of reads that were found for each nucleotide are reported for each position in each .simplified_VCF file.

All the positions where variants were found in at least one of the replicates are reported in each simplified_VCF file, i.e. all simplified_VCF files generated at the same time have the same set of positions with variant data or missing information (na). An example of .simplified_VCF file is provided in Chapter 7.

4.3 Requirements for VCF converter

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

(1) classic variant calling files (.vcf),
generated by the variant calling program FreeBayes
In case the .vcf file was produced with the other variant calling software
please see Chapters 4.10 and 4.11.

(2) depth of coverage files (.cov),
generated by the 'depth' command in the SAMtools utilities.

Comments:
(a) Both types of files are generated from a binary file containing
sequence alignment to reference genome (.bam file).
(b) 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".

4.4 Before you start

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

(1) Download from Dropbox (also available on GitHub, Chapter 1)
https://www.dropbox.com/sh/gmeoddqn6tjtwxc/AACJ-pnh3gzbhGtqLQ35wVMVa?dl=0

- GapJumper_01_Record_all_positions_from_VCF.pl
- GapJumper_02_Simplified_VCF_maker.pl
- GapJumper_AD_Convert_GATK-vcf_to_Nearly-Freebayes-vcf.pl (see 4.4.10)
- GapJumper_Example_One folder

Comments:
The Gap.Jumper_V01 repository on GitHub as well as the repository on Dropbox contains two Perl scripts (01 and 02) that are required in order to run VCF.Converter. Additionally, on Dropbox, the GapJumper_Example_One folder was provided. It contains a .vcf file, a .Q30vcf file, a .bam file, a .cov file and a .simplified_VCF file for each of four different replicates that are used in this tutorial. These replicates were taken from two different samples (Sample_01 and Sample_02)

4.5 Create VCF_Converter_working_dir folder

Copy to this folder:
(1) GapJumper_01_Record_all_positions_from_VCF.pl
(2) GapJumper_02_Simplified_VCF_maker.pl
(3) .bam files (one for each replicate)
(4) .vcf files (one for each replicate)

Comments:
In case of problems when installing samtools or vcffilter on your computer, .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 only keep high quality positions in the consensus, filter the .vcf files as follows:

```
vcffilter -f 'QUAL > 30' Sample.vcf > Sample_Q30.vcf'
```

Comments:
This step is optional but highly recommended! VCF.Converter can also 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:
(a) The script will read all .vcf files in the current directory.
(b) Output = All_positions_with_variants201XXXXX.combinedVCF
where: 201XXXXX is the date of file creation

(c) An example of an 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)

4.10 Compatibility with different variant calling software's

VCF.Converter uses .vcf files produced with different variant calling software's. However this file format is not standard and different programs can produce vcf. files contaminating different information about each polymorphic position. The file format can also change when the variant calling software is being updated. For these reasons, VCF.Converter requires vcf. files that were produced by Freebayes or any other software that produces the same type of .vcf files as Freebayes. Additionally, we provided perl script that allows changing .VCF produced by the GATK pipeline into the nearly Frebayes –type .vcf files that can be use with VCF.Converter (Chapter 4.11).

WE GIVE NO WARRANTY THAT VCF.Converter WILL WORK WITH ANY OTHER VARIANT CALLING SOWTWARES EXCEPT Freebayes and GATK.

However, please try to use .vcf files that syou obtained and then, if this doenst work try to convert these files into Nerarly-Freebayes .vcf type file. In case it still doesn't work, please report it as a bug (Chapter 1.7) and include small example of your raw .vcf file (no more than 10000 positions) with detailed information how that file was created.

4.11 Using .vcf files generated with GATK

.vcf files produced by the GATK pipeline are different then the v.cf file produced by Freebayes. Thus, in order to usethese files they must be at first convert to a "Nearly-Freebayes .vcf" file using perl scipt that we provided:

GapJumper_AD_Convert_GATK-vcf_to_Nearly-Freebayes-vcf.pl

The script should be copied into the directory that contains all .vcf files generated with the GATK and run. The example of .vcf file produced with GATK was provided in GapJumper_Example_One folder, if needed.

Chapter 5

Generating the consensus with GapJumper.plus

5.1 GapJumper.plus

GapJumper.plus is an 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 used to treat SNV calling data in a consensus built with one of the two probabilistic methods. They are described in Chapters 7-9.

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 Dropbox

https://www.dropbox.com/sh/gmeoddqn6tjtwxc/AACJ-pnh3gzbhGtqLQ35wVMVa?dl=0

(1) *GapJumper_03_GapJumperPlus.R*	(also available at GitHub, Chapter 1)
(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 (choose either the A or B possibilities)

A. Four directories, each to store 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 types of files:

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

and use this folder (B) 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 files that are named differently than what is suggested here, but then the path to the directory must be defined 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 all .simplified_VCF files (9 in total) 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 a new directory only for the purpose of this tutorial. It is possible to use .simplified_VCF files directly in *VCF_Converter_working_dir* and to set the path of the directory in R as follows:
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*

Comments:
In the case you use .simplified_VCF files that you generated following the instructions in Chapter 4 (Possibility 2, 5.3.3), it is important to remove additional replicates and Sample_03 from the Rg_Table and keep only the existing replicates and samples.

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 a 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 the 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 a consensus for each set of related replicates with all six different methods at the same time.

5.6 Arguments used by consensus building functions

Each R function 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. A 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 specifies 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 a 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!, NOT THE 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 Include (pos.y)

Gap.Jumper.plus gives the possibility to perform analysis on a subset of positions, e.g. to quickly test a 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 the 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 some of the positions potentially are part of the mitochondrial genome, when only the polymorphisms in nuclear genome need to be analyzed. 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 the 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. It can be use for improving 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 separately. For more information see Rosikiewicz et al. (submitted).

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 (i.e. the minimum coverage per position), 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:
The 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*)

Prior p is used in two probabilistic methods to construct a consensus, which assign quality scores ranging from zero to one to each nucleotide at each position in a consensus. The prior p is required to calculate partial quality scores in each replicate that were called basic probability assignments. For more information see Rosikiewicz et al. (submitted). Prior p value should be adjusted to the type of sequence data that are used. It depends 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 (see further in the text).

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

Prior p value

(1) 0<p<1

(2) Prior p should be equal to, or lower than the least expected relative allele frequency at each position in each replicate. For example, in order calculate quality scores for alleles in sequence data obtained from diploid organism, p value should be lower than, or equal to 0.5 and 0.25 in the case of sequence data obtained from tetraploid organism.

(3) Because of Insufficient coverage, sequence errors and bias in sequencing of different nucleotides at the same position with NGS, the nucleotide frequencies may be different to the expected values (e.g. 1:1.2 in sequence data from diploid organisms instead of expected 1:1). Therefore, it is recommended to lower values of prior p than the expected frequency of rare nucleotides. We recommend to use p=0.9 with sequence data from haploid organisms, p=0.4 to with sequence data from diploid organism and p=0.2 with sequence data from tetraploid organism.

(4) Alternatively, prior p can be calibrated using empirical sequence data. In approach, Gap.Jumper constructs a consensus with different p values. Subsequently, nucleotides in each consensus are classified as true positive or false positive using positions sequenced with an alternative experimental approach (Chapter 7.6.1) or validated with independent set of replicates from the same sample (Chapter 7.6.2). Prior p that allows best discrimination between true positives and false positives in each consensus should be used.

(5) The same procedure can be use to calculate noise threshold. However, it is important to use different set of data sequence data or different positions to calculate p values and noise threshold !.

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

In order to construct a consensus usin Site Elimination method (Run.Site.Elimination.method()), Gap.Jumper requires two arguments that describe the minimum number of replicates with missing data at a position required to exclude this position from a consensus. Gap.Jumper applies the lower value, for each set of replicates.

A. **se.na.nr –** maximum number of replicates with missing data at a given position that is allowed at that position

```
R >    se.na.nr <-as.numeric(1);
```

Exclude all positions with missing data in 2 or more replicates (Figure 5)

B. **B. se.na.rt –** maximum percentage of replicates with missing data at a given position that is allowed at that position (between 0 and 1)

```
R >    se.na.rt <-as.numeric(0.5);
```

Exclude all positions with missing data in at least 51% of replicates (Figure 5)

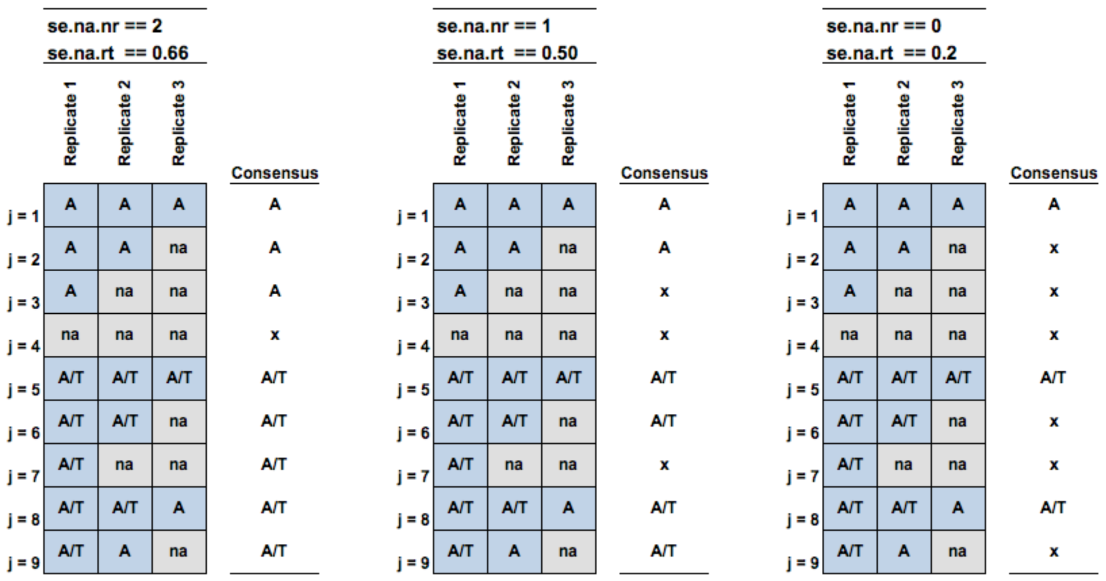
Comments:

(1) **se.na.nr==0,** removes all positions with missing data in at least one replicate

(2) In case you would like to use only **se.na.nr, define se.na.rt <-0.000001**

(3) In case you would like to use only **se.na.rt, define se.na.nr <-1000000**

(4) If both values are given, the lowest value is used



se.na.nr == 2 se.na.rt == 0.66	Replicate 1	Replicate 2	Replicate 3	Consensus
j = 1	A	A	A	A
j = 2	A	A	na	A
j = 3	A	na	na	A
j = 4	na	na	na	x
j = 5	A/T	A/T	A/T	A/T
j = 6	A/T	A/T	na	A/T
j = 7	A/T	na	na	A/T
j = 8	A/T	A/T	A	A/T
j = 9	A/T	A	na	A/T

se.na.nr == 1 se.na.rt == 0.50	Replicate 1	Replicate 2	Replicate 3	Consensus
j = 1	A	A	A	A
j = 2	A	A	na	A
j = 3	A	na	na	x
j = 4	na	na	na	x
j = 5	A/T	A/T	A/T	A/T
j = 6	A/T	A/T	na	A/T
j = 7	A/T	na	na	x
j = 8	A/T	A/T	A	A/T
j = 9	A/T	A	na	A/T

se.na.nr == 0 se.na.rt == 0.2	Replicate 1	Replicate 2	Replicate 3	Consensus
j = 1	A	A	A	A
j = 2	A	A	na	x
j = 3	A	na	na	x
j = 4	na	na	na	x
j = 5	A/T	A/T	A/T	A/T
j = 6	A/T	A/T	na	x
j = 7	A/T	na	na	x
j = 8	A/T	A/T	A	A/T
j = 9	A/T	A	na	x

Figure 5. Differences in consensuses built with Gap.Jumper, using the Site Elimination method and the se.na arguments (se.na.nr and se.na.rt) set for different values. Each consensus depicts the same nine positions (j).

5.6.10 Arguments required for the Fixed Thresholds method (ft…)

In order to construct a consensus using the 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 position in order to include this position in the consensus (Figure 6).
(It is not the same as se.na.nr !)

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

Use only 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 include this nucleotide in the consensus (Figure 7).

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

Use only 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 include that nucleotide in the consensus (Figure 8).

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

Use only nucleotides covered by a minimum 50% of reads (or more) at a given position in a given replicate

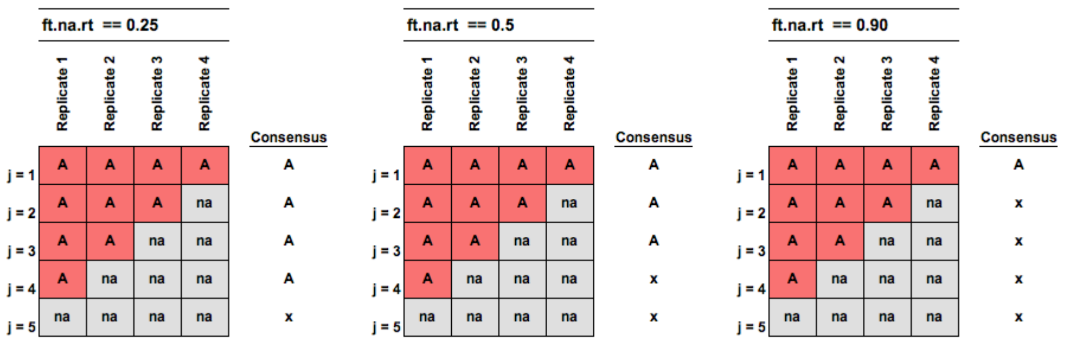


Figure 6. Differences in the consensuses built with Gap.Jumper.plus using the Fixed Threshold method when different values of the ft.na.rt threshold are used.

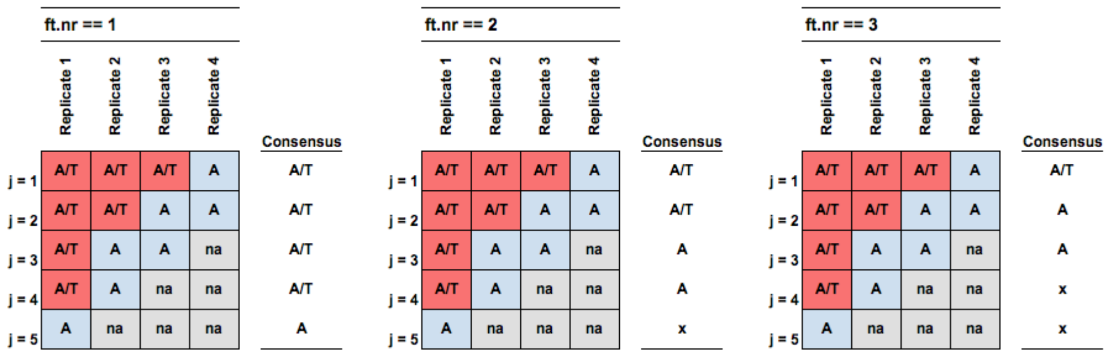
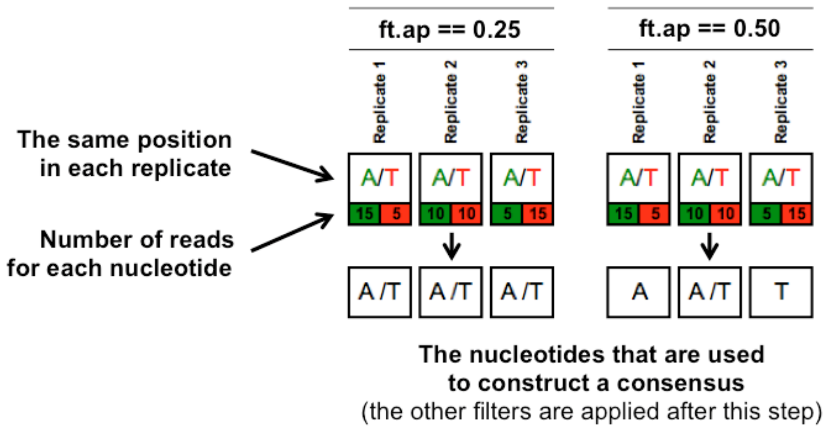


Figure 7. Differences in the consensuses built with Gap.Jumper.plus using the Fixed Threshold method when different values of the ft.nr threshold are used.



ft.ap == 0.25
ft.ap == 0.50
Replicate 1
Replicate 2
Replicate 3
The same position in each replicate
Number of reads for each nucleotide
A/T
15 5
10 10
5 15
A/T
A
T
The nucleotides that are used to construct a consensus
(the other filters are applied after this step)

Figure 8. The results of applying two different ft.ap thresholds 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 the R functions implemented in Gap.Jumper.plus (see section 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

R code examples for each method

6.1 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)
```

6.2 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)
```

6.3 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)
```

6.4 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 the Site Elimination method
  gp.dl      <-as.numeric(10)

# detection limit Arguments used for the 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)
```

6.5 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 the Site Elimination method
  gp.dl     <-as.numeric(10)

# detection limit Arguments used for the 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)
```

6.6 Probabilistic method on a large number of samples

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

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)
```

Chapter 7

Noise Filtration

7.1 Noise filtration

7.1.1 General information

Noise filtration is a process of removing nucleotides that are potentially caused by sequencing errors and sample contamination from a consensus, i.e. the false positives. It is performed only with the consensus build with one of the two probabilistic methods, because it is based on the difference in quality scores calculated for each nucleotide. Furthermore, probabilistic methods implemented in Gap.Jumper were designed to calculate quality scores that are lower in the case of nucleotides that are false positives than the quality scores calculated for nucleotides that were true positives. Therefore, false positives could be removed, without removing true positives. Two different functions were implemented in Gap.Jumper to remove potential false positives from each consensus:

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

(2) Remove.Noise.using.Acceptance.Threshold ()

7.1.1 Noise filtration is optional

Firstly, because each method and each type of sample, may produce SNV calling data with different number of errors. Thus, it may require calibration prior to performing noise filtration on each consensus. Secondly, because each nucleotide identified as potential error in a consensus is also potentially rare nucleotide. Thus, the information about this nucleotide may be used when different samples are compared with each other (see Chapter 9).

7.1.2 Noise filtration produces new consensus

Both functions that perform noise filtration generate new consensus for each consensus that was treated. Therefore, raw data are not overwritten. New file extension is added to each file: "NsFiltered.multiSNV".

7.2 Noise threshold and acceptance threshold

Noise threshold (Ns) and acceptance threshold (Acc) are used to identify nucleotides modeled with low quality scores in a given consensus. Subsequently these nucleotides are removed from that consensus. General quality score for each position is calculated again without quality scores for that nucleotides.

7.2.1 Noise threshold

Noise threshold is a percentage of nucleotides with lowest quality scores in a given consensus that is removed from that consensus.

Denoted as Ns; $0<Ns<1$.

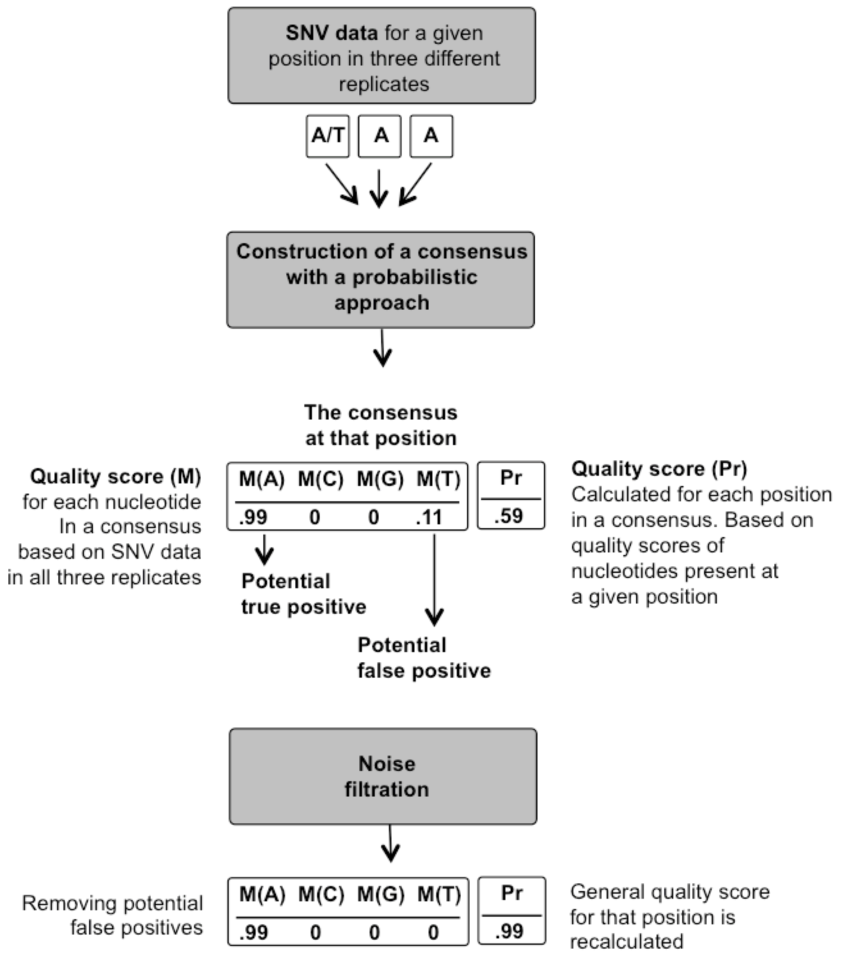
Example: if; Ns=0.02, than 2% of nucleotides with the lowest quality scores in a given consensus will be removed. The quality scores of removed nucleotides are replicated with 0 in a consensus. Furthermoe, the quality score for each position is re-caltucated using quality scores of the remaining nucleotides with quality scores >0 (Figure 10).

7.2.2 Acceptance threshold/ Cutoff value

A quality score that is use as a cutoff in removing nucleotides from the consensus.

Denoted as Acc; 0<Acc<1

Example: if Acc=0.2, than all nucleotides with quality scores lower than, or equal to 0.2 will be removed from a consensus. The percentage of removed nucleotides may be different in each consensus, depending how many nucleotides were modeled with quality score ≤ Acc.



SNV data for a given position in three different replicates
A/T
A
A
Construction of a consensus with a probabilistic approach
The consensus at that position
Quality score (M) for each nucleotide In a consensus based on SNV data in all three replicates
M(A) M(C) M(G) M(T)
.99 0 0 .11
Pr
.59
Quality score (Pr) Calculated for each position in a consensus. Based on quality scores of nucleotides present at a given position
Potential true positive
Potential false positive
Noise filtration
Removing potential false positives
M(A) M(C) M(G) M(T)
.99 0 0 0
Pr
.99
General quality score for that position is recalculated

Figure 9. Schematic diagram showing the noise filtration on one position in a consensus. In this example all nucleotides with quality score equal to or lower than 0.11 were removes as potential false positives.

7.3 Noise filtration using noise threshold

R > Remove.Noise.using.Noise.Threshold()

Arguments:

in.dir the path to directory containing .multiSNV files to be treated

res.dir the path to directory containing where new filtered files are going to be saved

file.name part of a .multiSNV file name that is required to identify files that are going to be filtered. We assume that more than one consensus may be stored in the same directory and that not all of them would be required in each analysis.
Examples:
R > file.name==".multiSNV"
all .multiSNV files in in.dir will be filtered
R > file.name==" Probabilistic_method.multiSNV"
use only files which were constructed with the probabilistic method
R > file.name==" Sample_01__Probabilistic_method.multiSNV"
Only this one file will be used

Ns noise threshold (see section 7.2.1)

per.replicate ("yes" or "no")
If yes; Ns is multiplied by the number of replicates in a given consensus

max.cutoff (0< max.cutoff≤1)
The nucleotides with quality scores higher than the max.cutoff won't be removed from a consensus. If max.cutoff==1, it is not used.

7.4 Noise filtration using acceptance threshold

R > Remove.Noise.using.Acceptance.Threshold()

Arguments:

in.dir; res.dir; file.name see Chapter 7.3

Acc acceptance threshold (see section 7.2.2)

7.5 Testing noise threshold

In order to test how many nucleotides are 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 the .multiSNV file
                                    constructed using the probabilistic method"
R >  all.Rec.nucl.multiSNV.name <-"The name of a corresponding .multiSNV file
                                    constructed using the All Recorded Nucleotides
                                    method"
R >  Noise.Threshold            <-as.numeric(0.01) # EXAMPLE
```

Run:

```
R > Test.Noise.threshold(res.dir,prob.multiSNV.name,
                         all.Rec.nucl.multiSNV.name,    retain.nucleotides)
```

7.6 Calculating an optimal noise threshold

We propose three different methods to define the thresholds that used for noise filtration.

7.6.1 Empirical approach

Small number of positions that are present in each consensus can be validated with an alternative sequencing approach. The positions may be used to identify the true positives and false positives in a given consensus (Figure 10). With this information, the acceptance threshold or the noise threshold may be calculated. The same approach may be use to calculate optimal prior p value (see Chapter 5.6.8)

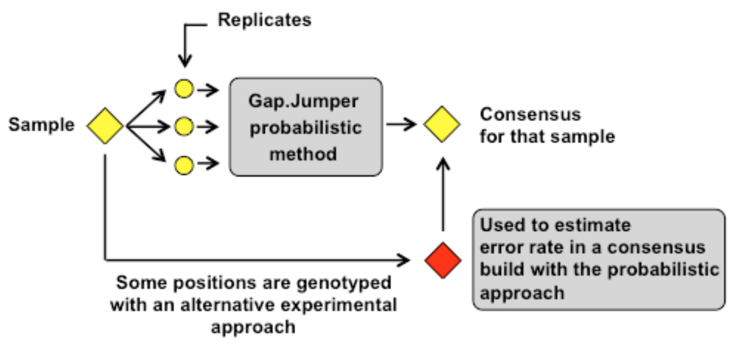

Replicates
Sample
Gap.Jumper probabilistic method
Consensus for that sample
Used to estimate error rate in a consensus build with the probabilistic approach
Some positions are genotyped with an alternative experimental approach

Figure 10. Schematic diagram showing the pathway to estimate error rate in a consensus build from many replicates with the empirical approach.

7.6.2 Using additional replicates

If possible, some samples can be sequence with larger number of replicates. These additional replicates can be use to construct one consensus with the conservative approach. Subsequently, this consensus should be compared with the consensus constructed from the other replicates of the same sample using probabilistic approach (Figure 11). Both consensuses can be compared with each other (see Chapter 9). It is important to remove the positions with missing data. The noise threshold is the percentage of the positions with variable nucleotides. Whereas, the acceptance threshold is the maximum (or mean) quality score calculated for variable nucleotides. For more information see Chapters 9 and 13. The same approach may be use to calculate optimal prior p value (see Chapter 5.6.8).

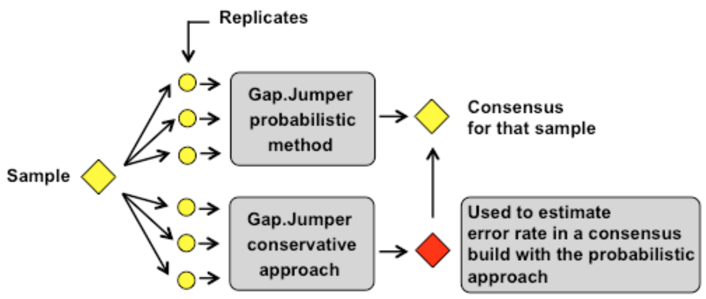

Replicates
Sample
Gap.Jumper probabilistic method
Consensus for that sample
Gap.Jumper conservative approach
Used to estimate error rate in a consensus build with the probabilistic approach

Figure 11. Schematic diagram showing the pathway to estimate error rate in a consensus build from many replicates with additional replicates.

7.6.3 Using predicted error rate in sequence data

In case no additional replicates are available and it is not possible to validate some of the positions with the other experimental approach, the noise threshold can be set as the expected percentage of sequence errors. The noise threshold depends on method that was used to generate the sequence data. For more information see M.Quail et al. (1), Qi et al (2) and Liu et al. (3).

7.7 Calculating acceptance threshold

We implemented this function to calculate the acceptance threshold (Acc) using noise threshold (Ns). The function returns the acceptance threshold that should be used as a cutoff value for that consensus, knowing the percentage of removed positions. In case several samples is used, the function returns the mean value of Acc obtained with each consensus.

R > Calculate.Acc.using.Ns()

Arguments:

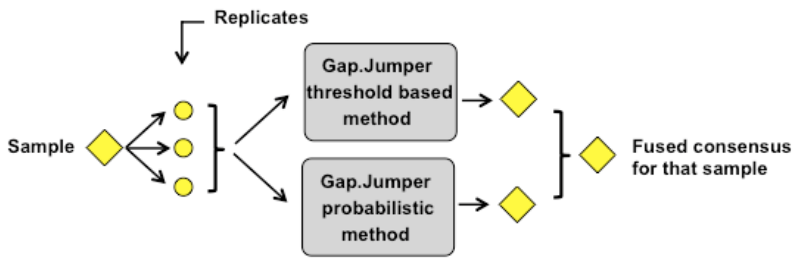
in.dir	the path to directory containing .multiSNV files
file.name	part of a .multiSNV file name that is required to identify files that are going to be filtered. We assume that more than one consensus may be stored in the same directory and that not all of them would be required in each analysis. For examples see section 7.3
Ns	noise threshold (section 7.2.1)
per.replicate	("yes" or "no") If yes; Ns is multiplied by the number of replicates in a given consensus
max.cutoff	($0<$ max.cutoff≤ 1) The nucleotides with quality scores higher than the max.cutoff won't be removed from a consensus. If max.cutoff==1, it is not used.

Chapter 8

Fusing the consensus

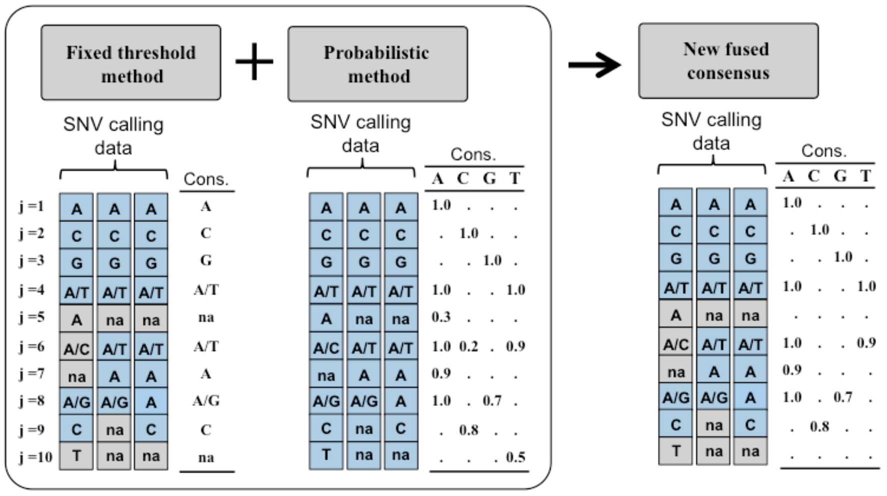
8.1 Combining different methods

Gap.Jumper allows fusing the consensus build with different methods for same set of replicates (Figure 12). It is made in order to produce a consensus that was build with one of the threshold-based methods (the conservative method, the site elimination methods, the fixed threshold method and the all recorded nucleotides method), which contains the quality scores for each nucleotide that were calculated with one of two probabilistic methods (Figure 13).



Replicates
Sample
Gap.Jumper threshold based method
Gap.Jumper probabilistic method
Fused consensus for that sample

Figure 12. Schematic diagram showing construction of a fused consensus.



Fixed threshold method

	SNV calling data			Cons.
j =1	A	A	A	A
j =2	C	C	C	C
j =3	G	G	G	G
j =4	A/T	A/T	A/T	A/T
j =5	A	na	na	na
j =6	A/C	A/T	A/T	A/T
j =7	na	A	A	A
j =8	A/G	A/G	A	A/G
j =9	C	na	C	C
j =10	T	na	na	na

Probabilistic method

SNV calling data			A	C	G	T
A	A	A	1.0	.	.	.
C	C	C	.	1.0	.	.
G	G	G	.	.	1.0	.
A/T	A/T	A/T	1.0	.	.	1.0
A	na	na	0.3	.	.	.
A/C	A/T	A/T	1.0	0.2	.	0.9
na	A	A	0.9	.	.	.
A/G	A/G	A	1.0	.	0.7	.
C	na	C	.	0.8	.	.
T	na	na	.	.	.	0.5

New fused consensus

SNV calling data			A	C	G	T
A	A	A	1.0	.	.	.
C	C	C	.	1.0	.	.
G	G	G	.	.	1.0	.
A/T	A/T	A/T	1.0	.	.	1.0
A	na	na	.	.	.	.
A/C	A/T	A/T	1.0	.	.	0.9
na	A	A	0.9	.	.	.
A/G	A/G	A	1.0	.	0.7	.
C	na	C	.	0.8	.	.
T	na	na	.	.	.	.

Figure 13. A construction of fused consensus with Gap.Jumper that contains quality scores calculated with the probabilistic method at the nucleotides that were selected with the fixed threshold method. Different positions in each consensus were denoted as j.

8.2 Fuse consensus function

R > fuse.consensus()

Arguments:

in.dir	the path to directory containing .multiSNV files to be treated
res.dir	the path to directory containing where new filtered files are going to be saved
tr.m.name	the name/names of .multiSNV file constructed with one of the threshold based methods.
pr.m.name	the name/names of .multiSNV file constructed with one of the probabilistic methods.

Comments:

(a) For examples of file names that can be use see section 7.3

(b) It is important to fuse consensens build with two methods, i.e. one file identify with tr.m.name has only one corresponding file that can be identify with pr.m.name

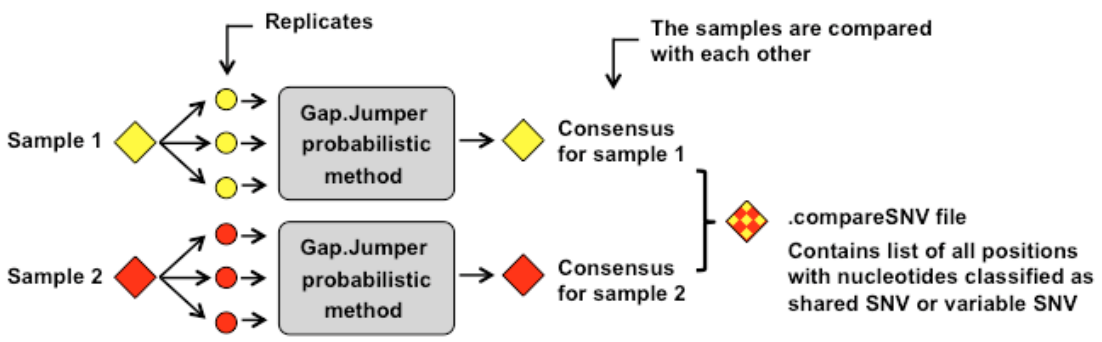
(c) New fused consensus has the name taken from the consensus build with the fixed threshold method and the extension that informs which probabilistic method was used to calculate the quality scores for each nucleotide.

Chapter 9

Sample comparisons

8.1 Comparing polymorphism in different samples

We implemented the function called Compare.samples.using.multiSNV.files() in Gap.Jumper.plus that allows performing comparison between different samples based on information in the consensus build for these samples. The results of comparison between the samples are saved in .compareSNV file format (see Chapter 13, Figure 14).



Replicates
The samples are compared with each other
Sample 1
Gap.Jumper probabilistic method
Consensus for sample 1
.compareSNV file
Contains list of all positions with nucleotides classified as shared SNV or variable SNV
Sample 2
Gap.Jumper probabilistic method
Consensus for sample 2

Figure 14. Schematic diagram showing the pathway to perform a comparison between two different samples using the consensus build from many replicates of each of these two samples.

8.2 Shared SNV and variable SNV

The nucleotides at each position are classified as shared SNV or variable SNV. The shared SNV are the nucleotides that were present in consensus build for both samples. The variable SNV were present in only one consensus, but on the position that had some other nucleotide in the second consensus, either variable SNV or shared SNV. The nucleotides with missing data at the corresponding position in the other sample are not classified (Figure 15).

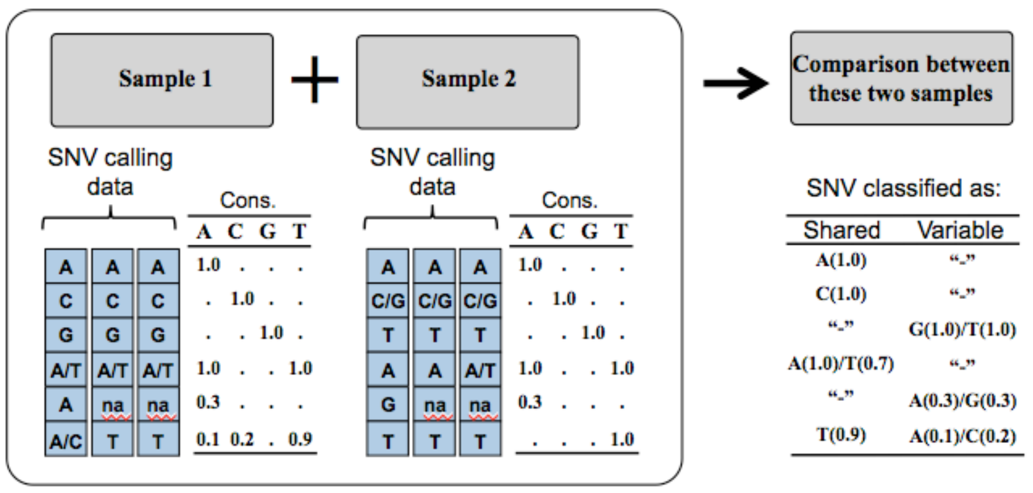


Figure 15. A comparison between consensus build with replicates of sample 1 and replicates of sample 2. Each nucleotide classified as variable SNV or shared SNV obtains new quality score (in brackets).

8.3 Quality scores are calculated for each compared nucleotide

Each nucleotide classified as shared SNV or variable SNV obtains a quality score. New qulity scores are calculated using the proportional conflict redistribution rule no. 5 (PCR5) that allows combining of quality scores estimated for these nucleotides in different samples (4). New quality scores are calculated only for the consensus that were build with one of the probabilistic methods. otherwise these quality scores are equal to one and only presence or absence of each nucleotide is recorded.

8.4 More than two samples can be compared

The same function can be used to compare many samples with each other. In this case each nucleotide is also classified as shared SNV and variable SNV. Variable SNV are not present in at least one sample. New quality score is calculated for each of these nucleotides. Moreover, it is possible to remove all positions with missing data or to perform a comparison using the positions that have data in at least tow different samples. The number of samples was not limited.

Chapter 10

Rg_Table

8.1 General information

Rg_Table defines which 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")

The first column contains the name of each replicate (e.g Replicate_01, Replicate_02, etc…) that is used to construct one consensus (Table 2). GapJumper.plus requires the full name of each replicate to uniquely identify each .simplified_VCF file. It is possible to use the name of each replicate several times. We included this possibility in order to allow creating consensus for the same sample with different replicates, e.g all available replicates and only good quality replicates. Additionally, it gives the possibility to permutate different replicates and build many versions of each consensus.

8.3 Second column ("Consensus_name")

The second column of Rg_Table contains the name of each consensus (i.e. sample name) that should be generated using the replicates listed in the first column. The name of each consensus should be given without file extensions (i.e. no need to add ".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 be generated from different numbers of replicates. However, all replicates that are used together have to be prepared using the same *All_positions_with_variants201XXXXX.combinedVCF* file. Therefore, all replicates used to generate one consensus have the same number of positions in their respective .simplified_VCF file.

(2) Because Gap.Jumper.plus uses only the replicates that are specified in the Rg_Table, all *.simplified_VCF* files can be stored in a same folder and used 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.multiSNV*

Chapter 11

.simplified_VCF file format

Each row in a .simplified_VCF file corresponds to one position in the reference genome that contained a SNV in at least one replicate. Each .simplified_VCF file contains the following columns:

(1) Replicate_name: indicates the names of the sample and the replicate.

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

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

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

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

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

(9) Ref: is the nucleotide found at this position in the reference genome (i.e. the reference allele).

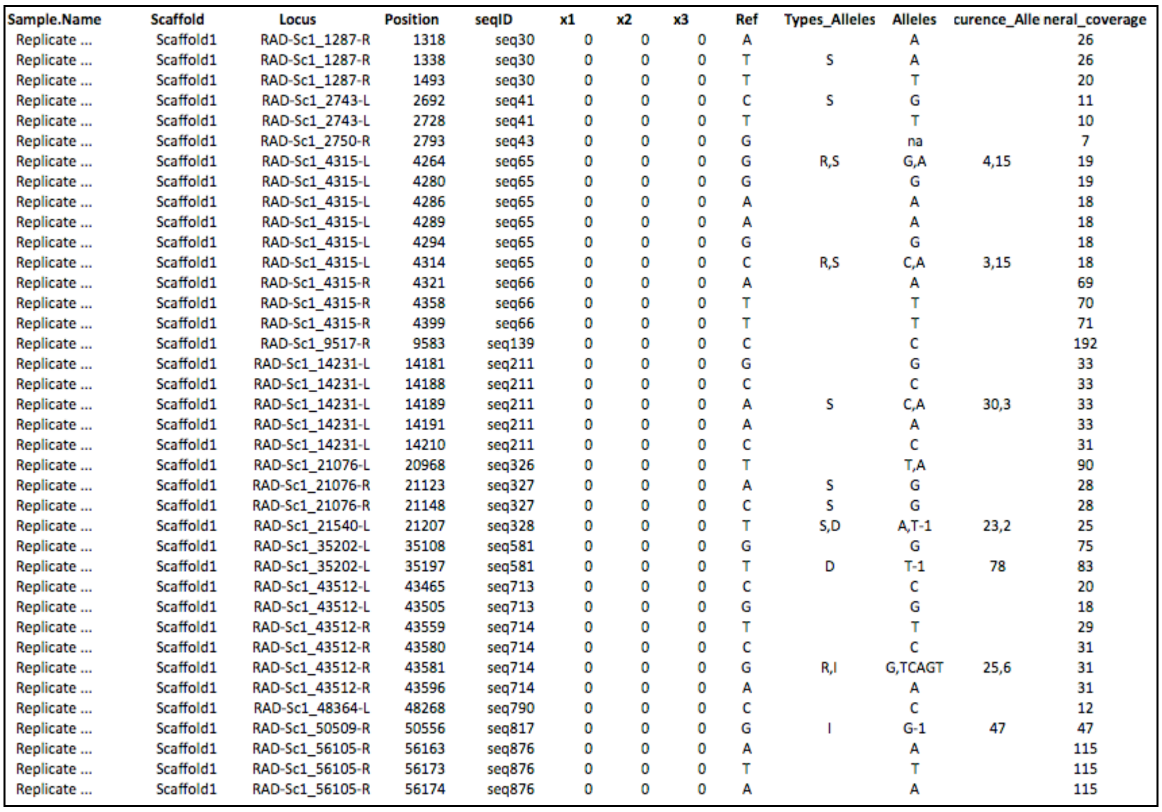
(10) Type_Alleles: describes the types of recorded alleles that are different from the reference genome. It can be "I" for "insertion", "D" for "deletion", "S" for "single nucleotide substitution (i.e. SNV)", "M" for "multiple nucleotide polymorphism" or "R" for "same as reference allele". If more than one type was found, all types are reported with commas as separators (e.g. a bi-allelic position could be reported as "R,S" if one allele is the same as in the reference genome, and the other allele contains a SNV). When two or more alleles are detected at one position, the order of appearance of the types of alleles in the column Type_Alleles follows the same order as in the columns Alleles and Occurrence_Alleles (see example) described below. If no allele different from the reference genome was detected, the cell remains empty.

(11) Alleles: reports the actual alleles accordingly to the Type_alleles column. Deletions are defined based on the last base remaining before the missing sequence. The last base is completed with a hyphen and a number. The number corresponds to the number of missing bases. E.g.:
ATGCTTGAA
AT-----AA => T-5 (i.e. T is here but the next 5 bases are missing)

(12) Occurrence_Alleles: reports the occurrence of alleles (i.e. number of sequencing reads that cover each allele). In the example below, position j is a bi-allelic position. The column Type_Alleles ("R,S") indicates that the first allele is the same as in the reference genome, and the second allele is a SNV. The column Alleles indicates the actual nucleotides found at this position: the first allele is a T (the reference therefore also has a T at this position) and the second allele is an A. Finally, the occurrence of each allele is indicated in column 12, Occurrence_Alleles: the T is covered by 20 sequencing reads, while the A is covered by 13 sequencing reads.

	Column 10	Column 11	Column 12
	Type_Alleles	Alleles	Occurrence_Alleles
position j	R,S	T,A	20,13

(13) General_coverage: is an indication of the total coverage at this position as calculated with the depth command of the SAMtools utilities'. The general coverage is not as precise as the allele occurrences. Therefore, the coverage in General_coverage sometimes exceeds the sum of the coverage of two or more alleles reported in Occurrence_Alleles, because only the high quality reads are reported for each allele in Occurrence_Alleles.



Sample.Name	Scaffold	Locus	Position	seqID	x1	x2	x3	Ref	Types_Alleles	Alleles	curence_Alle	neral_coverage
Replicate ...	Scaffold1	RAD-Sc1_1287-R	1318	seq30	0	0	0	A		A		26
Replicate ...	Scaffold1	RAD-Sc1_1287-R	1338	seq30	0	0	0	T	S	A		26
Replicate ...	Scaffold1	RAD-Sc1_1287-R	1493	seq30	0	0	0	T		T		20
Replicate ...	Scaffold1	RAD-Sc1_2743-L	2692	seq41	0	0	0	C	S	G		11
Replicate ...	Scaffold1	RAD-Sc1_2743-L	2728	seq41	0	0	0	T		T		10
Replicate ...	Scaffold1	RAD-Sc1_2750-R	2793	seq43	0	0	0	G		na		7
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4264	seq65	0	0	0	G	R,S	G,A	4,15	19
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4280	seq65	0	0	0	G		G		19
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4286	seq65	0	0	0	A		A		18
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4289	seq65	0	0	0	A		A		18
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4294	seq65	0	0	0	G		G		18
Replicate ...	Scaffold1	RAD-Sc1_4315-L	4314	seq65	0	0	0	C	R,S	C,A	3,15	18
Replicate ...	Scaffold1	RAD-Sc1_4315-R	4321	seq66	0	0	0	A		A		69
Replicate ...	Scaffold1	RAD-Sc1_4315-R	4358	seq66	0	0	0	T		T		70
Replicate ...	Scaffold1	RAD-Sc1_4315-R	4399	seq66	0	0	0	T		T		71
Replicate ...	Scaffold1	RAD-Sc1_9517-R	9583	seq139	0	0	0	C		C		192
Replicate ...	Scaffold1	RAD-Sc1_14231-L	14181	seq211	0	0	0	G		G		33
Replicate ...	Scaffold1	RAD-Sc1_14231-L	14188	seq211	0	0	0	C		C		33
Replicate ...	Scaffold1	RAD-Sc1_14231-L	14189	seq211	0	0	0	A	S	C,A	30,3	33
Replicate ...	Scaffold1	RAD-Sc1_14231-L	14191	seq211	0	0	0	A		A		33
Replicate ...	Scaffold1	RAD-Sc1_14231-L	14210	seq211	0	0	0	C		C		31
Replicate ...	Scaffold1	RAD-Sc1_21076-L	20968	seq326	0	0	0	T		T,A		90
Replicate ...	Scaffold1	RAD-Sc1_21076-R	21123	seq327	0	0	0	A	S	G		28
Replicate ...	Scaffold1	RAD-Sc1_21076-R	21148	seq327	0	0	0	C	S	G		28
Replicate ...	Scaffold1	RAD-Sc1_21540-L	21207	seq328	0	0	0	T	S,D	A,T-1	23,2	25
Replicate ...	Scaffold1	RAD-Sc1_35202-L	35108	seq581	0	0	0	G		G		75
Replicate ...	Scaffold1	RAD-Sc1_35202-L	35197	seq581	0	0	0	T	D	T-1	78	83
Replicate ...	Scaffold1	RAD-Sc1_43512-L	43465	seq713	0	0	0	C		C		20
Replicate ...	Scaffold1	RAD-Sc1_43512-L	43505	seq713	0	0	0	G		G		18
Replicate ...	Scaffold1	RAD-Sc1_43512-R	43559	seq714	0	0	0	T		T		29
Replicate ...	Scaffold1	RAD-Sc1_43512-R	43580	seq714	0	0	0	C		C		31
Replicate ...	Scaffold1	RAD-Sc1_43512-R	43581	seq714	0	0	0	G	R,I	G,TCAGT	25,6	31
Replicate ...	Scaffold1	RAD-Sc1_43512-R	43596	seq714	0	0	0	A		A		31
Replicate ...	Scaffold1	RAD-Sc1_48364-L	48268	seq790	0	0	0	C		C		12
Replicate ...	Scaffold1	RAD-Sc1_50509-R	50556	seq817	0	0	0	G	I	G-1	47	47
Replicate ...	Scaffold1	RAD-Sc1_56105-R	56163	seq876	0	0	0	A		A		115
Replicate ...	Scaffold1	RAD-Sc1_56105-R	56173	seq876	0	0	0	T		T		115
Replicate ...	Scaffold1	RAD-Sc1_56105-R	56174	seq876	0	0	0	A		A		115

Figure 16. An example of simplified_VCF file.

Chapter 12

.multiSNV file format

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

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

(2 – 9) **the same columns as in the .simplified_VCF files**, that were used to construct each consensus **(Chapter 11)**

(10-13) **M(A), M(C), M(G), M(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) **Pr**, general quality score calculated at each position using quality scores of nucleotides present at that position with M(snp)>0.

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

(16) **snp.presence.in.each.replicate:** reports nucleotides that were present in column 11 of the .simplified_VCF files used to construct the consensus. 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.

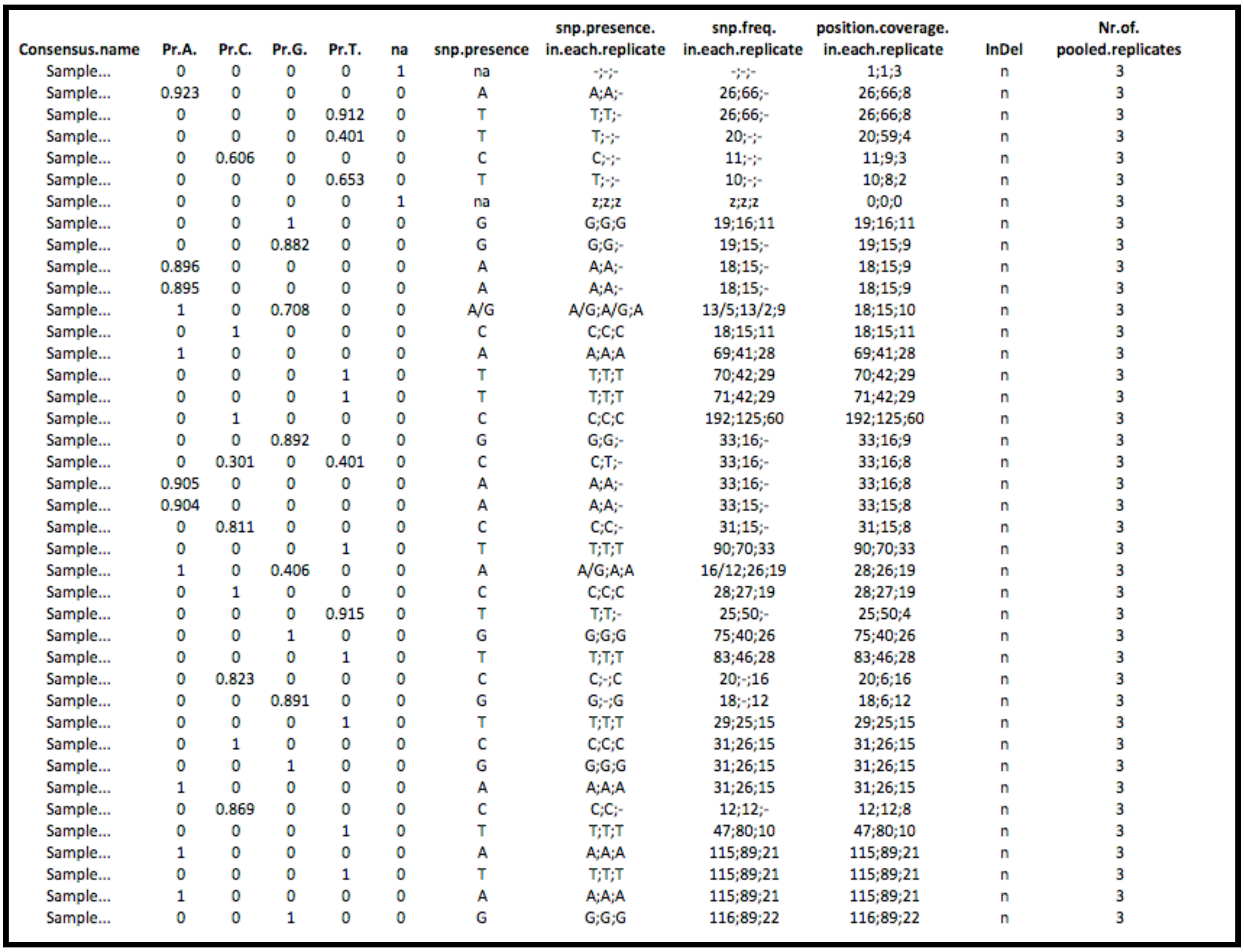
(17) **snp.freq.in.each.replicate:** reports the nucleotide occurrences in each replicate according to the previous column. "-" denotes missing data and "z" denotes no coverage. When more than one nucleotide is present, the order of appearance in each replicate is the same.
e.g. (16) "A/T;A/T;A/T" and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

(18) **position.coverage.in.each.replicate:** reports the values in the column General_coverage in the simplified_VCF files.

(19) **InDel:** reports presence ("y") or absence ("n") of insertions or deletions (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 consensuses saved in .multiSNV format can be found
in *GapJumper_Example_Two_Results* folder.



Consensus.name	Pr.A.	Pr.C.	Pr.G.	Pr.T.	na	snp.presence	snp.presence. in.each.replicate	snp.freq. in.each.replicate	position.coverage. in.each.replicate	InDel	Nr.of. pooled.replicates
Sample...	0	0	0	0	1	na	-;-;-	-;-;-	1;1;3	n	3
Sample...	0.923	0	0	0	0	A	A;A;-	26;66;-	26;66;8	n	3
Sample...	0	0	0	0.912	0	T	T;T;-	26;66;-	26;66;8	n	3
Sample...	0	0	0	0.401	0	T	T;-;-	20;-;-	20;59;4	n	3
Sample...	0	0.606	0	0	0	C	C;-;-	11;-;-	11;9;3	n	3
Sample...	0	0	0	0.653	0	T	T;-;-	10;-;-	10;8;2	n	3
Sample...	0	0	0	0	1	na	z;z;z	z;z;z	0;0;0	n	3
Sample...	0	0	1	0	0	G	G;G;G	19;16;11	19;16;11	n	3
Sample...	0	0	0.882	0	0	G	G;G;-	19;15;-	19;15;9	n	3
Sample...	0.896	0	0	0	0	A	A;A;-	18;15;-	18;15;9	n	3
Sample...	0.895	0	0	0	0	A	A;A;-	18;15;-	18;15;9	n	3
Sample...	1	0	0.708	0	0	A/G	A/G;A/G;A	13/5;13/2;9	18;15;10	n	3
Sample...	0	1	0	0	0	C	C;C;C	18;15;11	18;15;11	n	3
Sample...	1	0	0	0	0	A	A;A;A	69;41;28	69;41;28	n	3
Sample...	0	0	0	1	0	T	T;T;T	70;42;29	70;42;29	n	3
Sample...	0	0	0	1	0	T	T;T;T	71;42;29	71;42;29	n	3
Sample...	0	1	0	0	0	C	C;C;C	192;125;60	192;125;60	n	3
Sample...	0	0	0.892	0	0	G	G;G;-	33;16;-	33;16;9	n	3
Sample...	0	0.301	0	0.401	0	C	C;T;-	33;16;-	33;16;8	n	3
Sample...	0.905	0	0	0	0	A	A;A;-	33;16;-	33;16;8	n	3
Sample...	0.904	0	0	0	0	A	A;A;-	33;15;-	33;15;8	n	3
Sample...	0	0.811	0	0	0	C	C;C;-	31;15;-	31;15;8	n	3
Sample...	0	0	0	1	0	T	T;T;T	90;70;33	90;70;33	n	3
Sample...	1	0	0.406	0	0	A	A/G;A;A	16/12;26;19	28;26;19	n	3
Sample...	0	1	0	0	0	C	C;C;C	28;27;19	28;27;19	n	3
Sample...	0	0	0	0.915	0	T	T;T;-	25;50;-	25;50;4	n	3
Sample...	0	0	1	0	0	G	G;G;G	75;40;26	75;40;26	n	3
Sample...	0	0	0	1	0	T	T;T;T	83;46;28	83;46;28	n	3
Sample...	0	0.823	0	0	0	C	C;-;C	20;-;16	20;6;16	n	3
Sample...	0	0	0.891	0	0	G	G;-;G	18;-;12	18;6;12	n	3
Sample...	0	0	0	1	0	T	T;T;T	29;25;15	29;25;15	n	3
Sample...	0	1	0	0	0	C	C;C;C	31;26;15	31;26;15	n	3
Sample...	0	0	1	0	0	G	G;G;G	31;26;15	31;26;15	n	3
Sample...	1	0	0	0	0	A	A;A;A	31;26;15	31;26;15	n	3
Sample...	0	0.869	0	0	0	C	C;C;-	12;12;-	12;12;8	n	3
Sample...	0	0	0	1	0	T	T;T;T	47;80;10	47;80;10	n	3
Sample...	1	0	0	0	0	A	A;A;A	115;89;21	115;89;21	n	3
Sample...	0	0	0	1	0	T	T;T;T	115;89;21	115;89;21	n	3
Sample...	1	0	0	0	0	A	A;A;A	115;89;21	115;89;21	n	3
Sample...	0	0	1	0	0	G	G;G;G	116;89;22	116;89;22	n	3

Figure 17. An example of .multiSNV file. Columns two to nine were remover from the example because they contain the same information as the corresponding columns in the .simplified_VCF. File (Chapter 11)

Chapter 13

.compareSNV file format

GapJumper genereates one file calles .compareSNV for each set of samples that were compared with each other.

(1) **File.Name**: given by the user

(2 – 9) **the same columns as in the .simplified_VCF files**, that were used to construct each consensus **(Chapter 11)**

(10) **Pr(all.elements)**
The quality score for a given position. It is calculated using quality scores of all nucleotides recorded at that positions, ether shared or variable which were present in all compared samples at that position.

(11) **Shared.SNV**
The identity of nucleotide/s classified as shared between the samples

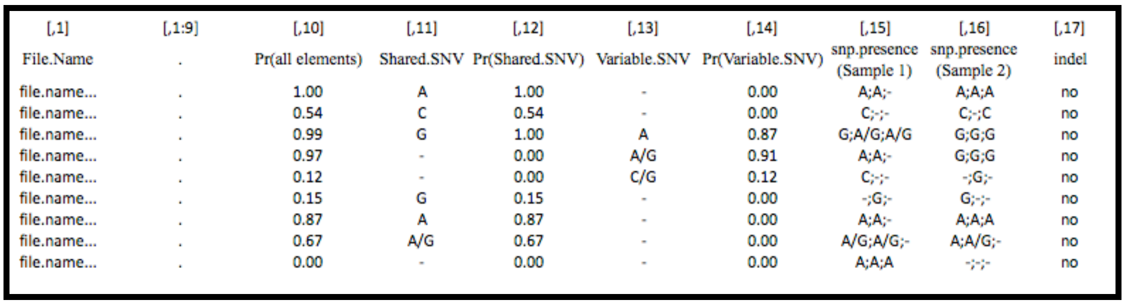
(12) **Pr(Shared.SNV)**
The quality calculated for nucleotide/s classified as shared between the samples

(13) **Variable.SNV**
The identity of nucleotide/s classified as variable between the samples

(14) **Pr(Shared.SNV)**
The quality calculated for nucleotide/s classified as variable between the samples

(15-16) **snp.presence:** reports the nucleotides that are present in each replicates in each sample. The nucleotides from different replicates of the same sample are dividied by ";". Each column represents one sample for which the consensus was build. The number of columns depends on the number of samples that were compared. "-" denotes missing data (cov<dl); "z" denotes missing data in a replicates on position with no coverage (cov.=0)

(last column) shows the positions with insertions or deletions that were detected in at least one of the replicates



[,1]	[,1:9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]	[,17]
File.Name	.	Pr(all elements)	Shared.SNV	Pr(Shared.SNV)	Variable.SNV	Pr(Variable.SNV)	snp.presence (Sample 1)	snp.presence (Sample 2)	indel
file.name...	.	1.00	A	1.00	-	0.00	A;A;-	A;A;A	no
file.name...	.	0.54	C	0.54	-	0.00	C;-;-	C;-;C	no
file.name...	.	0.99	G	1.00	A	0.87	G;A/G;A/G	G;G;G	no
file.name...	.	0.97	-	0.00	A/G	0.91	A;A;-	G;G;G	no
file.name...	.	0.12	-	0.00	C/G	0.12	C;-;-	-;G;-	no
file.name...	.	0.15	G	0.15	-	0.00	-;G;-	G;-;-	no
file.name...	.	0.87	A	0.87	-	0.00	A;A;-	A;A;A	no
file.name...	.	0.67	A/G	0.67	-	0.00	A/G;A/G;-	A;A/G;-	no
file.name...	.	0.00	-	0.00	-	0.00	A;A;A	-;-;-	no

Figure 18. An example of .compareSNV file. Columns two to nine were removed from the example because they contain the same information as the corresponding columns in the .simplified_VCF. File (Chapter 11). Each row represents one position.

Chapter 14

When naming files:

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

14.2 It is allowed to use “-“, “_” ,
but not more than once per file name

14.3 The name of each file cannot 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..

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

References

1. Quail,M.A., Smith,M., Coupland,P., Otto,T.D., Harris,S.R., Connor,T.R., Bertoni,A., Swerdlow,H.P. and Gu,Y. (2012) A tale of three next generation sequencing platforms : comparison of Ion Torrent , Pacific Biosciences and Illumina MiSeq sequencers.
2. Qi,Y., Liu,X., Liu,C., Wang,B. and Hess,K.R. (2015) Reproducibility of Variant Calls in Replicate Next Generation Sequencing Experiments. 10.1371/journal.pone.0119230.
3. Liu,Q., Guo,Y., Li,J., Long,J., Zhang,B. and Shyr,Y. (2012) Steps to ensure accuracy in genotype and SNP calling from Illumina sequencing data. *BMC Genomics*, **13 Suppl 8**, S8.
4. Smarandache,F. and Dezert,J. (2006) Advances and Applications of DSmT for Information Fusion (Collected works) American Research Press Rehoboth.