2014

Universitat Autònoma
de Barcelona

Institut de Biotecnologia i
Biomedicina

[PREDICTION OF INVERSIONS WITH GRIAL *–USER'S MANUAL*]

GRIAL is a software for predicting inversions and refining their breakpoints based on the computation of some geometric rules specific to inversions in a PEM assay. If you use GRIAL in your research, please cite: Martínez-Fundichely et al. (in preparation).

Contents

Chapter 1

Introduction to GRIAL

1.1 –What is GRIAL?

GRIAL (Geometric Rules Inversion ALgorithm) is a software for predicting inversions and refining their breakpoints based on the computation of some geometric rules specific to inversions in a paired-end mapping (PEM) assay. GRIAL can also score the predicted inversions based on the number of concordant PEMs in the region of the inversion, and the expected number of discordant PEMs supporting each breakpoint given the inversion size, the repetitive nature of the sequence of the breakpoints, and the sequencing coverage.

The features included in the algorithm are the following:

* **Preprocessig step:** Given a set of discordant PEMs for inversions, GRIAL preprocesses the data to remove duplicated PEMs, PEMs with overlapping ends, or PEMs mapping to repeats. See 4.3 –Filtering of PEMs for further details.

* **Histogram clustering:** GRIAL implements this clustering method to determine clusters of PEMs supporting the same inversion breakpoint. See 4.4 –Clustering for further details.

* **Inversion definition and Breakpoint narrowing:** GRIAL defines predictions of inversions based on one or two clusters of PEMs supporting one or the two inversion breakpoints, respectively, and narrow the breakpoints location based on these clusters. See 4.5 –Prediction of inversions and 4.6 –Definition of the breakpoints for further details.

* **Scoring:** GRIAL scores the predicted inversions based on the number of concordant PEMs in the region of the inversion, and the expected number of discordant PEMs supporting each breakpoint given the inversion size, the repetitive nature of the sequence of the breakpoints, and sequencing coverage. See 4.7 –Scoring for further details.

1.2 –Versions, Licences, Disclaimer and Copyright

1.2.1 –Versions

The first public version of GRIAL (April 2014) is **GRIAL v.3.2**.

1.2.2 –License

1.2.2.1 –GRIAL

GRIAL is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details (http://www.gnu.org/licenses/).

1.2.2.2 –Documentation

This documentation is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

1.2.3 –Copyright

1.2.4 –External scripts

GRIAL uses the **fetchChromSizes** bash script created by the UCSC Genome Bioinformatics Group to grab chromosome sizes from the UCSC. Please refer to the original website of the UCSC Genome Bioinformatics Group at http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/fetchChromSizes for more information on this utility.

1.3 –Getting help & Reporting bugs

We welcome your feedback and suggestions about GRIAL. Please contact us through the GRIAL repository "Issues" system at https://github.com/lpantano/InvFest-code/tree/GRIAL. This ensures that

requests do not get lost and you have the option to automatically receive a notification when a bug has been fixed or an enhancement has been developed.

Alternatively, you can contact us at invfestdb@uab.cat or at http://grupsderecerca.uab.cat/cacereslab/.

Our mailing address is:

Comparative and Functional Genomics group
Institut de Biotecnologia i de Biomedicina
Universitat Autònoma de Barcelona
08193 Bellaterra, Barcelona, Spain

1.4 –Citing GRIAL

If you use GRIAL in your research, please cite: Martínez-Fundichely *et al.* (in preparation).

Chapter 2

Installing GRIAL

2.1 –Where to fetch GRIAL?

GRIAL is available in a GitHub public repository at https://github.com/lpantano/InvFest-code/tree/GRIAL (ZIP download at https://github.com/lpantano/InvFest-code/archive/GRIAL.zip).

2.2 –Requirements

GRIAL runs on Unix OS & Microsoft Windows. However, please note that the external bash script **fetchChromSizes** from the UCSC is for Unix OS only, and thus a couple of options of GRIAL are not available for use under Microsoft Windows. Please refer to 4.1 –Configuration File for further information.

It requires Perl and the following Perl libraries:

- DBI
- DBD::mysql
- Cwd
- List::Util
- Math::CDF
- Number::Interval
- Statistics::Descriptive

2.3 –Installation

1. Install all the requirements listed in 2.2 –Requirements.

2. Download and extract the GRIAL software somewhere in your computer.

2.4 –Usage

Usage: perl ./GRIAL.pl <Configuration File>

Options: All options are set in the self-explanatory Configuration File (<GRIAL.config>). See 4.1 –Configuration File for further information.

Input data: See 4.2 –Input data for further information.

Output data: See 4.8 –Output data for further information.

Chapter 3

Quick-start guide to GRIAL

3.1 –Test GRIAL with the supplied sample data

The sample data supplied with GRIAL will allow you to predict inversions in the human genome based on the discordant and concordant PEMs of fosmids obtained from 9 individuals and mapped to the HG18 reference genome by Kidd *et al.* 2008.

1. Download and extract the sample data supplied with GRIAL in the directory **./testing/input** (relative to the directory where you extracted the GRIAL software)

2. Some other input data for GRIAL will be automatically generated by a script. Execute the following command from the directory where you extracted the GRIAL software: **perl ./generateRepeatTracks.pl**

3. Next you will run GRIAL. Execute the following command from the directory where you extracted the GRIAL software: **perl ./GRIAL.pl**

4. Find your results in the directory **./testing/output**. Please refer to 4.8 –Output data for an explanation of result file formats.

3.2 –Use GRIAL with your own data

By using your own data you can predict inversions in any genome given a set of discordant PEMs (and a set of concordant PEMs if desired) obtained by any paired-end sequencing technology and mapped to any reference genome.

1. Place your input data anywhere in your computer (see 4.2 –Input data for input file formats) and create a directory for the output

2. Create a new Configuration File as needed, editing at least the sections "SECTION 1: PATHS & FILENAMES" and "SECTION 2: LIBRARY STATISTICS & ASSEMBLY" of the default Configuration File provided with GRIAL **<GRIAL.config>**. See 4.1 –Configuration File for further information.

3. If you have chosen 'yes' to any of the discard/tag repeats/segdups parameters in "SECTION 3: FILTERING OF MAPPINGS" of the Configuration File (DISCARD_MAPPINGS_IN_REPEATS, DISCARD_MAPPINGS_IN_SEGDUPS, TAG_MAPPINGS_IN_REPEATS, TAG_MAPPINGS_IN_SEGDUPS), or the mapability parameters in "SECTION 6: SCORING" (MAPABILITY_REPEATS, MAPABILITY_SEGDUPS), you need to provide specific files for these analyses. In this case, those files will be automatically generated by executing the following command from the directory where you extracted the GRIAL software: **perl ./ generateRepeatTracks.pl <Configuration_File_Name>**

4. Next you will run GRIAL. Execute the following command from the directory where you extracted the GRIAL software: **perl ./GRIAL.pl <Configuration_File_Name>**

5. Find your results in the output directory. Please refer to 4.8 –Output data for an explanation of result file formats.

Chapter 4

GRIAL reference manual

4.1 –Configuration File

All the necessary information for GRIAL to run an analysis must be specified in the Configuration File. We recommend creating a new Configuration File for each analysis you want to run, while keeping the default Configuration File that comes with the distribution of this software as a reference. All the different parameters for GRIAL are classified into different sections and are properly explained in the same place, making the Configuration File self-explanatory. Parameter names are always uppercase, while values are lowercase. Lines preceded by # in the Configuration File are comments not read by GRIAL. We provide here a formatted copy of the default Configuration File, as used to analyze the provided sample data from Kidd *et al.* 2008, as a reference guide. Please refer to 3.1 –Test GRIAL with the supplied sample data for easy instructions on how to analyze this data.

SECTION 1: PATHS & FILENAMES

# Provide a file with the DISCORDANT MAPPINGS
DISCORDANTS_FILENAME=./testing/input/discordantFosmids.txt

# Prefix for the files with CONCORDANT MAPPINGS (necessary to estimate the library statistics from the concordant mappings -if statistics are not provided in Section 2-, and to score inversions -in Section 5 or using the GRIALscore.pl script-). You should include one file per chromosome, and what you are providing in the parameter is the PATH to these files plus the PREFIX of these files; the name of each file is PREFIX.CHROMOSOME (e.g. ALL_c.chr1, ALL_c.chr2, ALL_c.chr3, etc). If you have all concordant mappings in one single file, you can split them into multiple files using the bash command: *for i in $(seq 1 22) X Y; do grep -oP "^chr$i\t.+" concordantFosmids > concordantFosmids.chr$i; done*
CONCORDANTS_FILENAME_PREFIX=./testing/input/concordantFosmids

# Complete path to output folder. Please, make sure that output folder exists.
OUTPUT_PATH=./testing/output/

You can choose a prefix for all output files. Useful if you run GRIAL multiple times with different parameters
OUTPUT_PREFIX=

# If you choose 'yes' to any of the discard/tag repeats/segdups parameters in Section 3 (DISCARD_MAPPINGS_IN_REPEATS, DISCARD_MAPPINGS_IN_SEGDUPS, TAG_MAPPINGS_IN_REPEATS, TAG_MAPPINGS_IN_SEGDUPS), or the mapability parameters in Section 6 (MAPABILITY_REPEATS, MAPABILITY_SEGDUPS), you need to provide a directory where the specific files for these analyses are found (see manual)
REPEATFILES_PATH=./testing/input/

SECTION 2: LIBRARY STATISTICS & ASSEMBLY

# How are you providing the library statistics?: 0 = Metrics provided below | 1 = GRIAL must calculate them from the concordant mappings provided in CONCORDANTS_FILENAME_PREFIX (Section 1, extra step for GRIAL that takes long time…)
CONCORDANTS=0

# If no concordants file (CONCORDANTS_FILENAME_PREFIX), you need to provide the statistics, at least AVG+XSTD|MIN+MAX && STD|VAR (XSTD is the number of STDs below and above the AVG; will be used to set MIN and MAX)
CONCORDANTS_AVG=39221.9922629086
CONCORDANTS_MIN=25163
CONCORDANTS_MAX=49224
CONCORDANTS_RAN=24061
CONCORDANTS_STD=2690.88150325854
CONCORDANTS_VAR=7240843.26457892
CONCORDANTS_XSTD=3

# Assembly to which mappings where done (UCSC nomenclature). Necessary to repair inconsistencies due to inexistent chromosomes or BPs out of the range of the chromosomes. Default: (blank)
ASSEMBLY=hg18

SECTION 3: FILTERING OF MAPPINGS

# You may specify which libraries you want to analyze from the input file (comma-separated). Otherwise, PEMs from all the libraries will be included in the analyses. Default: all
LIBRARIES=all

# You may specify one or more chromosomes that you want to analyze from the input file (comma-separated). Otherwise, all the chromosomes from the input file will be analyzed. Default: all
CHROMOSOMES=all

# Do you want to analyze PEMs in RANDOM chromosomes? (yes|no). Default: no
ANALYZE_RANDOM_CHROMOSOMES=no

# Remove PEMs that were selected from multiple mappings in the genome and report this as reliability value in the results. Choose 'any' to allow any number of mappings (discard no PEMs). Default: 10
MAX_ALLOWED_MAPPINGS=any

# Weight the support of each PEM based on the # of possible mappings it has in the genome (support = 1 / num_mappings). Otherwise, each PEM adds a support of 1 to the cluster (yes|no). If 'yes', please provide the # of possible mappings for each PEM in the input file!!! Default: no
WEIGHT_PEM_SUPPORT=no

# Remove PEMs with overlapping ends (maximum percentage overlap allowed). Default: 50
OVERLAPPINGENDS_PERC=50

# Weight duplicated PEMs. You may specify different values for different libraries in the form: *DUPLICATEPEMS_BP_LIBNAME=XX*. Default: 5
DUPLICATEPEMS_BP=50
DUPLICATEPEMS_BP_G248=17

Discard/Tag PEMs mapping in repeats masked by RepeatMasker. You may specify which repeats you want to include in the discard/tag-lists (single-character codes one after another, no commas!; e.g. *EIX*). Single-character codes corresponding to long-codes by RepeatMasker:

S=SINE
s=SINE?
L=LINE
l=LINE?
T=LTR
t=LTR?
D=DNA
d=DNA?
R=snRNA/tRNA/rRNA/srpRNA/scRNA/RNA
I=Simple-repeat
X=Low-complexity
E=Satellite
C=RC
U=Unknown
u=Unknown?
O=Other
o=Other_not_in_this_list

If *XXX_REPEATS_LIST=all*, all repeats will be included. Choose 'no' to not discard/tag any PEM mapping in repeats. Defaults: DISCARD_MAPPINGS_IN_REPEATS=no, DISCARD_REPEATS_LIST=all, DISCARD_REPEATS_NUMREADS=any, TAG_MAPPINGS_IN_REPEATS=no, TAG_MAPPINGS_IN_REPEATS_ALL=yes, TAG_REPEATS_LIST=all, TAG_REPEATS_NUMREADS=any

DISCARD_MAPPINGS_IN_REPEATS=no
DISCARD_REPEATS_LIST=all
DISCARD_REPEATS_NUMREADS=any
TAG_MAPPINGS_IN_REPEATS=no
TAG_MAPPINGS_IN_REPEATS_ALL=yes
TAG_REPEATS_LIST=all
TAG_REPEATS_NUMREADS=any

Discard/Tag PEMs mapping in Segmental Duplications. Choose 'no' to not discard/tag repeats. Defaults: DISCARD_MAPPINGS_IN_SEGDUPS=no, DISCARD_SEGDUPS_NUMREADS=any, TAG_MAPPINGS_IN_SEGDUPS=no, TAG_MAPPINGS_IN_SEGDUPS_ALL=yes, TAG_SEGDUPS_NUMREADS=any

DISCARD_MAPPINGS_IN_SEGDUPS=no
DISCARD_SEGDUPS_NUMREADS=any
TAG_MAPPINGS_IN_SEGDUPS=no
TAG_MAPPINGS_IN_SEGDUPS_ALL=yes
TAG_SEGDUPS_NUMREADS=any

SECTION 4: CLUSTERING & INVERSION PREDICTION

Minimum *weighted* support for clusters (min PEMs per cluster) and predicted inversions (min total PEMs per inversion). Allow inversions predicted by one single cluster (++ or --)? (yes|no). Defaults: MIN_SUPPORT_CLUSTERS=1, MIN_SUPPORT_INVERSIONS=2, ALLOW_SINGLE_CLUSTER_INVERSIONS=yes

MIN_SUPPORT_CLUSTERS=1
MIN_SUPPORT_INVERSIONS=2
ALLOW_SINGLE_CLUSTER_INVERSIONS=yes

# Use the strict rules to join positive & negative clusters into inversions (see manual) (yes|no). Default: yes
JOINPOSNEGCLUSTERS_STRICT=yes

# Repaire inconsistences due to inexistent chromosomes or BPs out of the range of the chromosomes (yes|no). Requires that ASSEMBLY is specified above. Default: no
REPAIR_BP_INCONSISTENCES=yes

# Tag suspicious inversions due to incongruent narrow BPs (yes|no). Default: yes
TAG_INCONGRUENT_BPS=yes

# Tag the use of the same cluster in >1 inversions (yes|no). Default: yes
TAG_MULTIPLE_USE_OF_CLUSTERS=yes

SECTION 5: OUTPUT

# Do you want PEM indexes to be written in output files? (yes|no). Default: no
OUTPUT_PEM_INDEXES=yes

# You may specify which breakpoint definition you want to get as output (wide|narrow|all). Default: all
BP_DEFINITION=all

SECTION 6: SCORING

# Automatically execute the GRIALscore.pl script after predicting inversions. Needs the file with concordant mappings (CONCORDANTS_FILENAME_PREFIX) (yes|no). Default: no
SCORE_INVERSIONS=yes

# Mapability calculation in the regions in and surrounding the Breakpoints: compute mapability of PEMs according to the presence of repeats and inverted segmental duplications in the regions in and surrounding the breakpoints. It improves the computation of GRIAL scores. Requires that ASSEMBLY and REPEATFILES_PATH are set and available, and MAPABILITY_SEGDUPS=yes requires connection to the UCSC database (yes|no). Default: MAPABILITY_REPEATS=no, MAPABILITY_SEGDUPS=no
MAPABILITY_REPEATS=yes
MAPABILITY_SEGDUPS=yes

# Mapability calculation in the regions in and surrounding the Breakpoints: in PEM reads of MAPABILITY_FRAGMENT_SIZE base pairs, MAPABILITY_NUM_CHANGES sequencing errors are expected (which might result in incorrect mapping). Only required if MAPABILITY_REPEATS=yes OR MAPABILITY_SEGDUPS=yes. Default: 400
MAPABILITY_FRAGMENT_SIZE=400

# Mapability calculation in the regions in and surrounding the Breakpoints: in PEM reads of MAPABILITY_FRAGMENT_SIZE base pairs, MAPABILITY_NUM_CHANGES sequencing errors are expected (which might result in incorrect mapping). Only required if MAPABILITY_REPEATS=yes OR MAPABILITY_SEGDUPS=yes. Default: 2
MAPABILITY_NUM_CHANGES=2

# Mapability calculation in the regions in and surrounding the Breakpoints: reduce mapability for the presence of REPEATS by RepeatMasker. MAPABILITY_REPEATS_IDENTITY indicates the % of identity of repeats, from 0 to 1. Only required if MAPABILITY_REPEATS=yes. Default: 0.99
MAPABILITY_REPEATS_IDENTITY=0.99

# Mapability calculation in the regions in and surrounding the Breakpoints: structure of segmental duplications: % of the segmental duplication that is the central core, from 0 to 1. (1-MAPABILITY_SEGDUPS_CORE) is the % of the segmental duplication that are the extremes (left+right tails). Only required if MAPABILITY_SEGDUPS=yes. Default: 0.8
MAPABILITY_SEGDUPS_CORE=0.8

# Mapability calculation in the regions in and surrounding the Breakpoints: structure of segmental duplications: % of the changes that occur in the central core, from 0 to 1. (1-MAPABILITY_SEGDUPS_CHANGES_CORE) of the changes occur in the extremes (left+right tails). Only required if MAPABILITY_SEGDUPS=yes. Default: 0.5
MAPABILITY_SEGDUPS_CHANGES_CORE=0.5

4.2 –Input data

All the necessary information regarding input data to GRIAL must be specified in the sections "SECTION 1: PATHS & FILENAMES" and "SECTION 2: LIBRARY STATISTICS & ASSEMBLY" of the Configuration File. Please refer to 4.1 –Configuration File for further information.

(i) Input data for GRIAL are paired-end mapping data that is discordant for inversions (not the complete result of a PEM assay).

(ii) General statistics of the PEM assay are necessary for GRIAL to predict inversions. They can be specified in the Configuration File under section "SECTION 2: LIBRARY STATISTICS & ASSEMBLY", or can be rather computed by GRIAL if adequate files with concordant mappings are supplied for each chromosome (see configuration option CONCORDANTS_FILENAME_PREFIX under section "SECTION 1: PATHS & FILENAMES" of the Configuration File). Concordant mappings are also used by GRIAL to score the final inversion predictions. Please refer to 4.1 –Configuration File for further information.

(iii) General file format rules: In each of the following file formats below, lines beginning with "#" are considered comments and ignored. Since mapped start and end are in terms of the reference sequence, Start < End. Each line gives the mapping information for a PEM. Columns are tab-separated in the formats explained below.

4.2.1 – Format for discordant mappings

colLibrary: A string with an individual's library unique code.

colName: A string with a unique PEM code.

colChr: A string identifying the chromosome where the read was mapped. (Format is according to UCSC)

colStartLeft: An integer identifying the coordinate of the start of the left read.

colEndLeft: An integer identifying the coordinate of the end of the left read.

colStartRight: An integer identifying the coordinate of the start of the right read.

colEndRight: An integer identifying the coordinate of the end of the right read.

colStrand: A char identifying the strand where the reads mapped. (For inversion signal both reads map in the same strand. Acceptable formats: + or -)

colSeqOrient: The sequencig orientation of the left read.(On paired-end sequencig, orientation of both reads are complementary Forward<->Reverse. Acceptable formats: R or F)

For example:

#colLibrary	colName	colChr	colStartLeft		colEndLeft	colStartRight		colEndRight
	colStrand	colSeqOrient						
ABC10	ABC10_722068_A17	chr16	1197946	1198800	1230933	1231727	+	R
ABC10	ABC10_722268_G24	chr21_random	823924	824684	1438534	1439260	+	R
ABC10	ABC10_722368_M2	chrX	48586345	48587180	48600806	48601372	+	F
ABC10	ABC10_638658_C11	chr17	2946503	2947231	3098930	3099699	-	R

4.2.2 – Format for concordant mappings

colChr: A string identifying the chromosome where the reads mapped. (Format is according to UCSC)

colStart: An integer identifying the coordinate of the start of the left read.

colEnd: An integer identifying the coordinate of the end of the right read.

colTemplate: A string with the template fragment unique code.

colScore: An integer score.

colStrand: A char identifying the strand where the REVERSE read mapped. (Acceptable formats: + or -)

colReadCount: An integer count of number of reads mapped. (Always 2 for concordants)

colStartLeft: An integer identifying the coordinate of the start of the left read.

colStartRight: An integer identifying the coordinate of the start of the right read.

colLeftReadLenght: An integer identifying the length of the mapping of the left read.

colRightReadLenght: An integer identifying the length of the mapping of the right read.

colLeftReadName: A string with the unique code of the left read.

colLeftReadName: A string with the unique code of the right read.

colLibrary: A string with the individual library unique code.

For example:

```
chr1    17311    58504    ABC10_44536500_F14        3026    -        2        17311    57755    791      749
        173650_ABC10_2_1_000044536500_F14.REVERSE.1        173650_ABC10_2_1_000044536500_F14.FORWARD.1        ABC10

chr1    17311    58532    ABC10_44633400_E8         3223    -        2        17311    57737    835      795
        174552_ABC10_2_1_000044633400_E8.REVERSE.1         174552_ABC10_2_1_000044633400_E8.FORWARD.1         ABC10

chr1    17330    58503    ABC10_44511300_C9         2877    -        2        17330    57804    754      699
        173650_ABC10_2_1_000044511300_C9.REVERSE.1         173650_ABC10_2_1_000044511300_C9.FORWARD.1         ABC10
```

4.3 –Filtering of PEMs

All the options regarding the initial filtering of PEMs are specified in the section "SECTION 3: FILTERING OF MAPPINGS" of the Configuration File. Several filtering options are available, which are explained in the self-explanatory Configuration File. Please refer to 4.1 –Configuration File for further information.

Briefly, input PEMs can be filtered based on the library, the chromosome, and the number of possible mappings to the reference genome. PEMs can also be discarded when the mappings of both ends overlap. In this case a maximum percentage of overlap might be specified. Artificially duplicated PEMs can also be identified and weighted when several PEMs coming from the same library and having the same orientation have virtually the same location (see Tuzun *et al.* 2005). In this case a minimum distance between the mapped ends of the reads might be specified for some or all the libraries, taking into account the sequencing strategy. PEMs can also be either discarded or just tagged when the reads map to repeats or segmental duplications. Different kinds of repeats according to RepeatMasker might be specified.

4.4 –Clustering

The options regarding the clustering of the PEMs are specified in the section "SECTION 4: CLUSTERING & INVERSION PREDICTION" of the Configuration File. Please refer to 4.1 –Configuration File for further information.

Basically, you can specify the minimum number of PEMs required for generating a cluster. The clustering stage harbors the first set of geometrical rules developed and implemented in GRIAL.

4.4.1 – First set of rules

Identification and clustering of discordant PEMs that are consistent with the same inversion breakpoint:

Rule 1.1.- For 2 PEM to belong to the same inversion BP, all should have the same orientation, and the distance between the two ends have to be within the variation of the library size.

Rule 1.2.- Sum rule: The sum of the position of the two ends have to be constant and it cannot vary more than the library size.

Schematically, the following equations must satisfy for two PEMs to be included in the same cluster:

For ++ clusters:

abs(jLS–iLS) < MAX

jLS < iRS

jRS > iLS

abs(iRS–jRS) < MAX

abs((jLS+jRS)–(iLS+iRS)) < RANGE

For -- clusters:

abs(iRE–jRE) < MAX

jRE > iLE

jLE < iRE

abs(jLE–iLE) < MAX

abs((jLS+jRS)–(iLS+iRS)) < RANGE

where i and j are two different PEMs, LS is the mapping start coordinate of the left read, LE is the mapping end coordinate of the left read, RS is the mapping start coordinate of the right read, RE is the mapping end coordinate of the right read, MAX is the maximum fragment size of the library, and RANGE is the range of fragment sizes of the library (maximum fragment size - minimum fragment size).

4.5 –Prediction of inversions

The options regarding the prediction of inversions are specified in the section "SECTION 4: CLUSTERING & INVERSION PREDICTION" of the Configuration File. Please refer to 4.1 –Configuration File for further information.

Basically, you can specify the minimum number of PEMs required to form an inversion, or if inversions predicted by either a ++ cluster or a -- cluster only might be allowed. You can also specify to use or not the most strict rules to join ++ and -- clusters into inversions (see below), to repair inconsistencies due to inexistent chromosomes or location of breakpoint coordinates out of the range of the chromosomes (if a reference ASSEMBLY is specified), to tag suspicious inversions due to incongruent narrow breakpoints (tagged in the output with the symbol !), or to tag the use of the same cluster in >1 predicted inversions (tagged in the output with the symbol #).

The prediction of inversions stage harbors the second set of geometrical rules developed and implemented in GRIAL.

4.5.1 – Second set of rules

Merging of PEM of the two breakpoints of the same inversion:

Rule 2.1.- The maximum distance between the limits of the clusters cannot be more than twice the library length plus the expected variation (and the beginning of PEM + cannot cross with the end of PEM –).

Rule 2.2.- The difference between the sum of the positions of the two ends of PEM + and PEM – clusters should be within twice the library length plus the expected variation.

Schematically, the following equations must satisfy for two clusters to be merged into the same inversion:

Basic rules:

$$posSUM < negSUM$$

$$abs(posSUM–negSUM) > 2 \times MIN$$

$$abs(posSUM–negSUM) < 2 \times MAX$$

$$abs(posLS–negLE) \leq 2 \times MAX$$

$$abs(posRS–negRE) \leq 2 \times MAX$$

$$posRE \leq negRS$$

$$negLS \geq posLE$$

Strict rule:

Definitions:

$$posT = posSUM + (AVG + (MAX–AVG)/sqrt(posDIS))$$

$$posB = posSUM + (AVG – (AVG–MIN)/sqrt(posDIS))$$

$$negT = negSUM – (AVG – (AVG–MIN)/sqrt(negDIS))$$

$$negB = negSUM – (AVG + (MAX–AVG)/sqrt(negDIS))$$

Rule:

$$(((posT > negB) \text{ AND } (posT < negT)) \text{ OR } ((posB > negB) \text{ AND } (posB < negT)) \text{ OR } ((negT > posB) \text{ AND } (negT < posT)) \text{ OR } ((negB > posB) \text{ AND } (negB < posT)))$$

where posSUM and negSUM are the average of iLS+iRS of all the paired-ends supporting the positive and the negative clusters respectively (see previous section), posLS and negLS are the smallest mapping coordinate of all left reads in the positive and negative clusters respectively, posLE and negLE are the highest mapping coordinate of all left reads in the positive and negative clusters respectively, posRS and

negRS are the smallest mapping coordinate of all right reads in the positive and negative clusters respectively, posRE and negRE are the highest mapping coordinate of all right reads in the positive and negative clusters respectively, posDIS and negDIS are the discordant support of the positive and the negative clusters respectively, AVG is the average fragment size of the library, MIN is the minimum fragment size of the library, and MAX is the maximum fragment size of the library.

4.6 –Definition of the breakpoints

In the section "SECTION 5: OUTPUT" of the Configuration File you can specify the refinement of the inversion breakpoints in the option *BP_DEFINITION*. If this option is set to *'wide'*, rule 3.1 below will be applied. Alternatively, if this option is set to *'narrow'*, rule 3.2 below will be applied instead. This option can also be set to *'all'* and you will obtain a set of files with the wide breakpoints and another set of files with the narrow breakpoints. Please refer to 4.1 –Configuration File for further information.

4.6.1 – Third set of rules

Refinement of the intervals where the breakpoints are most likely located:

Rule 3.1.- Breakpoints are defined by the position of the most internal mapping outside the inversion and the closest within (either + or –).

Rule 3.2.- Alternatively, the breakpoint limits can be defined by adding to the closest PEM, the difference between the maximum library length minus the distance between the closest and the furthest PEM.

Schematically, breakpoints are refined as follows:

Inversions supported by both a positive and a negative cluster:

Wide breakpoints:

wBP1S = max(posLE, negLE–MAX)

wBP1E = min(posRS, negLS, posLS+MAX)

wBP2S = max(posRE, negLE, negRE–MAX)

wBP2E = min(negRS, posRS+MAX)

Narrow breakpoints:

nBP1S = max(wBP1S, Test all i PEMs [iLEneg–(MAX–(iREneg–wBP2E))])

nBP1E = min(wBP1E, Test all i PEMs [iLEneg–(MIN–(iREneg–iLEneg))])

nBP2S = max(wBP2S, Test all i PEMs [iRSpos+(MIN–(iRSpos–iLSpos))])

nBP2E = min(wBP2E, Test all i PEMs [iRSpos+(MAX–(wBP1S–iLSpos))])

Inversions supported by a positive cluster only:

Wide breakpoints:

wBP1S = posLE

wBP1E = min(posRS, posLS+MAX)

wBP2S = posRE

wBP2E = posRS+MAX

Narrow breakpoints:

nBP1S = wBP1S

nBP1E = wBP1E

nBP2S = max(wBP2S, Test all i PEMs [iRSpos+(MIN–(iRSpos–iLSpos))])

nBP2E = min(wBP2E, Test all i PEMs [iRSpos+(MAX–(wBP1S–iLSpos))])

Inversions supported by a negative cluster only:

Wide breakpoints:

wBP1S = negLE–MAX

wBP1E = negLS

wBP2S = max(negLE, negRE–MAX)

wBP2E = negRS

Narrow breakpoints:

nBP1S = max(wBP1S, Test all i PEMs [iLEneg–(MAX–(iREneg–wBP2E))])

nBP1E = min(wBP1E, Test all i PEMs [iLEneg–(MIN–(iREneg–iLEneg))])

nBP2S = wBP2S

nBP2E = wBP2E

where wBP1S and wBP1E are the start and end positions of the first breakpoint according to rule 3.1 (wide breakpoints), wBP2S and wBP2E are the start and end positions of the second breakpoint according to rule 3.1 (wide breakpoints), nBP1S and nBP1E are the start and end positions of the first breakpoint according to rule 3.2 (narrow breakpoints), nBP2S and nBP2E are the start and end positions of the

second breakpoint according to rule 3.2 (narrow breakpoints), posLS and negLS are the smallest mapping coordinate of all left reads in the positive and negative clusters respectively, posLE and negLE are the highest mapping coordinate of all left reads in the positive and negative clusters respectively, posRS and negRS are the smallest mapping coordinate of all right reads in the positive and negative clusters respectively, posRE and negRE are the highest mapping coordinate of all right reads in the positive and negative clusters respectively, iLSpos is the mapping start coordinate of the left read from a positive PEM, iLEpos is the mapping end coordinate of the left read from a positive PEM, iRSpos is the mapping start coordinate of the right read from a positive PEM, iREpos is the mapping end coordinate of the right read from a positive PEM, iLSneg is the mapping start coordinate of the left read from a negative PEM, iLEneg is the mapping end coordinate of the left read from a negative PEM, iRSneg is the mapping start coordinate of the right read from a negative PEM, iREneg is the mapping end coordinate of the right read from a negative PEM, MAX is the maximum fragment size of the library, and MIN is the minimum fragment size of the library.

4.7 –Scoring

There is a whole section ("SECTION 6: SCORING") in the Configuration File where you can specify several parameters about the scoring. Please refer to 4.1 –Configuration File for further information.

Basically, if *SCORE_INVERSIONS* is set to *'yes'*, GRIAL will compute two different metrics to score the predicted inversions:

4.7.1 – Discordant-Concordant (DC) ratio score

Assuming a diploid genome, the ratio of the number of discordant fragments supporting a predicted breakpoint interval with respect to the total number of mappings (both concordant and discordant) across the same interval is expected to be 1 for homozygous inversions and 0.5 for heterozygous inversions (half the reads mapped at each side of the breakpoint interval should be concordant and the other half should be discordant). Deviations from these expected values can be considered a signal of erroneous inversion predictions. To calculate this score, the observed DC ratio is calculated for all individuals assuming that individuals with discordant PEMs only are homozygous for the inversion, while those having both discordant and concordant PEMs in the region are heterozygous for the inversion (which is a conservative estimate). Observed and expected numbers of PEMs are calculated for each individual separately, and then the DC ratio is calculated across individuals as follows:

Definitions:

$T = total number of PEMs in the region, including discordant and concordant

$O = observed number of discordant PEMs supporting the inversion

$E = expected number of discordant PEMs supporting the inversion, considering that the individual is homozygote if there are no concordant PEMs in the region (then $E=$O), or that the individual is heterozygote if there are concordant PEMs in the region (then $E=$T/2).

Then, after calculating these numbers for each individual and summing up values across individuals:

DC_ratio = $E/$O

Furthermore, an expected *versus* observed ratio test is also performed:

$$DC\_ratio\_test = \frac{(\$O/\$T)-(\$E/\$T)}{\sqrt{(((\$O/\$T)\times(1-(\$O/\$T))/\$T)+((\$E/\$T)\times(1-(\$E/\$T))/\$T))}}$$

or 1 if there are no concordant PEMs in the region of the inversion

We normally consider as unreliable those predictions for which the expected DC ratio is more than double the observed one, and the expected *versus* observed ratio test P-value is less than 0.05.

This score has the advantage that uses concordant mappings in the same region to estimate the expected number of discordant mappings depending on the predicted inversion genotype, and therefore it is not sensitive to specific characteristics of the region that might affect the number of mapped reads. In addition, for valid inversions, the DC ratio for each individual can be also used to predict the inversion genotype. Then, in the output results you will find, for each individual, which is the likelihood of this individual being heterozygous for the inversion, which is calculated as follows:

Definitions:

alpha = $O/$T *or* 0.5 if there are no concordant PEMs in the region of the inversion

discPEMs = observed number of discordant PEMs supporting the inversion = $O

concPEMs = observed number of concordant PEMs in the region of the inversion = $T–$O

Then:

$$Likelihood\_heterozygote = alfa^{discPEMs} \times (1\text{-}alfa)^{concPEMs}$$

4.7.2 – Discordant Support (DS) probability score

The second score calculates the expected support for the inversion considering a homogeneous read-depth for each chromosome, taking into account the predicted inversion size, the size of both breakpoint intervals, the length of the fragments of the sequencing library, and the mapability in the region according to the presence of segmental duplications and other repetitive elements.

Given a haplotype carrying an inversion, only a region of the length of the library or of the size of the inversion, whichever is smaller, will contain PEMs discordant for one breakpoint, and the same applies to the other breakpoint. Then, if we assume that the read-depth is homogeneous along the chromosome,

we can calculate which is the expected number of discordant PEMs laying in the region and thus supporting the inversion breakpoint in the case that the individual is heterozygote (if the individual is homozygous it will be twice this number, and thus our choice is conservative). Finally, for each inversion breakpoint separately, we calculate which is the probability that this breakpoint, for which we expect lambda discordant PEMs supporting it, it is supported by $k \leq k_{obs}$ discordant PEMs, according to the Poisson distribution:

$$P(k \leq k_{obs}; lambda) = \sum_{k=0}^{k \leq k_{obs}} \frac{e^{-lambda} \times lambda^{k}}{k!}$$

4.7.2.1 – Mapability correction

However, some of the sites in these regions might not be effectively mapable (or less mapable) if they contain repetitive sequences, so the presence of repeats and segmental duplications is considered here to reduce the size of the region (mapability correction). If *DISCARD_MAPPINGS_IN_REPEATS=yes* in the Configuration File, then those sites included in *DISCARD_REPEATS_LIST* are effectively not mapable (PEMs mapping there were excluded before inversion prediction) and are also excluded when computing the mapability correction. The same is true when *DISCARD_MAPPINGS_IN_SEGDUPS=yes*. Furthermore, even if those parameters above were set to *'no'*, GRIAL can take into account the presence of repeats by RepeatMaker (*MAPABILITY_REPEATS=yes*) or segmental duplications (*MAPABILITY_SEGDUPS=yes*) (or both) to reduce the mapability in the region of the breakpoints for the computation of the DS score. Please refer to 4.1 –Configuration File for further information.

Repeats:

If *MAPABILITY_REPEATS=yes*, you can choose a value for the percentage of identity of repetitive sites (*MAPABILITY_REPEATS_IDENTITY*). Then, the mapability on these sites is calculated using the Binomial distribution as follows:

Definitions:

n = length of the sequencing reads in base pairs (*MAPABILITY_FRAGMENT_SIZE*)

x = number of sequencing errors expected in the sequencing reads (*MAPABILITY_NUM_CHANGES*)

id = % of identity of repetitive sites (*MAPABILITY_REPEATS_IDENTITY*)

Then:

mapability_repeats = 1–pbinom(x, n, 1–id)

Segmental Duplications:

In the case of segmental duplications, the calculation of the mapability is a bit trickier. Mismatches among different copies of a segmental duplication are normally not randomly distributed across the segmental duplication, so that the central core of the segmental duplication tends to be really identical, while most mismatches occur at the ends. Then, if *MAPABILITY_SEGDUPS=yes*, you can set a couple of parameters to determine the structure of the segmental duplications in your genome:

Definitions:

MAPABILITY_SEGDUPS_CORE = % of the segmental duplication that is the central core, from 0 to 1

MAPABILITY_SEGDUPS_CHANGES_CORE = % of the changes that occur in the central core, from 0 to 1

Then, taking into account the identity of each specific segmental duplication from the UCSC database, the region is subdivided into different sites with different identities, depending on the position within the segmental duplication (core or extremes), and for each region, the corresponding mapability is calculated as follows:

$$\text{mapability\_segdups}_i = 1 - \text{pbinom}(x, n, 1 - \text{id}_i)$$

Then, the general mapability of segmental duplications is ponderated by taking into account the length of each region with a different identity:

$$\text{mapability\_segdups} = \frac{\sum_i^{\#\text{regions}} (\text{mapability\_segdups}_i \times \text{length\_segdups}_i)}{\sum_i^{\#\text{regions}} (\text{length\_segdups}_i)}$$

Combined mapability:

When *DISCARD_MAPPINGS_IN_REPEATS=yes* or *DISCARD_MAPPINGS_IN_SEGDUPS=yes* or *MAPABILITY_REPEATS=yes* or *MAPABILITY_SEGDUPS=yes*, a value for mapability_total is calculated for each breakpoint as follows:

$$\text{mapability\_total} = \frac{\text{length\_norep} + (\text{mapability\_repeats} \times \text{length\_repeats}) + (\text{mapability\_segdups} \times \text{length\_segdups})}{\text{total\_BP\_length}}$$

where length_norep is the number of sites which are not repeats nor segmental duplications (note that the mapability of these sites is 1), and total_BP_length is the total length of the breakpoint region also including those sites that have been totally discarded by the options *DISCARD_MAPPINGS_IN_REPEATS* or *DISCARD_MAPPINGS_IN_SEGDUPS* (the mapability of these sites is 0).

This value ranges from 0 to 1, where 0 means that the breakpoint region is no mapable (we don't expect any discordant PEM supporting the breakpoint) and 1 means completely mapable (we expect a number of discordant PEMs supporting the breakpoint that depends exclusively on the size of the fragments of the library and the size of the inversion). Then, this mapability_total is used to calibrate lambda as follows:

$$lambda\_mapability = lambda \times mapability\_total$$

and this lambda corrected by the mapability is the one used to calculate which is the probability that this breakpoint, for which we expect lambda_mapability discordant PEMs supporting it, it is supported by $k \leq k_{obs}$ discordant PEMs, according to the Poisson distribution:

$$P(k \leq k_{obs}; lambda\_mapability) = \sum_{k=0}^{k \leq k_{obs}} \frac{e^{-lambda\_mapability} \times lambda\_mapability^{k}}{k!}$$

We normally consider as unreliable all the predictions for which the Poisson distribution P-value of the observed discordant support is less than 0.001.

4.8 –Output data

From less processed to more processed results:

4.8.1 – PEM Indexes

One file per chromosome and strand with the processed input PEMs (optional; see *OUTPUT_PEM_INDEXES* in the Configuration File). Please refer to 4.1 –Configuration File for further information.

Filename convention: <chr><strand>.PEMids

Each line corresponds to one input PEM. Columns are tab-separated in the following format:

colChr_strand: A string with the chromosome and strand of the PEM.

colPEMID: An integer with the unique identifier of the PEM (assigned by GRIAL).

colLibraryID: A string with the library of the PEM.

colPEMoriginalID: A string with the original PEM identifier.

colPEMnumMaps: An integer with the number of possible mappings of this PEM against the reference genome.

colClusterID: An integer with the cluster ID to which the PEM was assigned.

colDuplicatePEMs: A string with other PEM IDs which are duplicates of the current PEM.

colTags: If requested in the Configuration File (see options TAG_MAPPINGS_IN_REPEATS and TAG_MAPPINGS_IN_SEGDUPS in the section "SECTION 3: FILTERING OF MAPPINGS"), this column will include a set of symbols telling if the PEM maps to repeats or segdups in the reference genome (^ = repeats, and : = segdups).

For example:

```
chr1-	128	ABC8	ABC8_42542700_M4	1	494	129..
chr1-	129	ABC8	ABC8_5692549_G9	1	494	128..
chr1-	305	ABC9	ABC9_45485400_J12	1	495	306..307..
```

4.8.2 – Clusters

One file per chromosome and strand with the PEM clusters.

Filename convention: <chr><strand>.clusters

Each line corresponds to one cluster. Columns are tab-separated in the following format:

colChr_strand: A string with the chromosome and strand of the cluster. If *TAG_MULTIPLE_USE_OF_CLUSTERS=yes* in the Configuration File, lines starting with the # symbol mean that the cluster has been used in multiple inversion predictions. Please refer to 4.1 –Configuration File for further information.

colClusterID: An integer with the unique identifier of the cluster (assigned by GRIAL)

colPEM_list: A string with the list of PEMs included in the cluster (comma-separated)

colSupport: An integer with the total number of PEMs supporting the cluster (unfiltered PEMs)

colFiltSupport: An integer with the total number of PEMs supporting the cluster (filtered PEMs)

colSumCoef: A float with the sum coefficient of the cluster

colVarSumCoef: A float with the variance of the sum coefficient of the cluster

colLeftStartMin: An integer with the minimum coordinate of left reads (start coordinate for positive clusters / end coordinate for negative clusters)

colLeftEndMax: An integer with the maximum coordinate of left reads (start coordinate for positive clusters / end coordinate for negative clusters)

colRightStartMin: An integer with the minimum coordinate of right reads (start coordinate for positive clusters / end coordinate for negative clusters)

colRightEndMax: An integer with the maximum coordinate of right reads (start coordinate for positive clusters / end coordinate for negative clusters)

colOneMapPEMs: An integer with the number of PEMs having only one possible mapping to the reference genome, that are supporting the cluster

colTags: If requested in the Configuration File (see options *TAG_MAPPINGS_IN_REPEATS* and *TAG_MAPPINGS_IN_SEGDUPS* in the section "SECTION 3: FILTERING OF MAPPINGS", and options

TAG_INCONGRUENT_BPS and *TAG_MULTIPLE_USE_OF_CLUSTERS* in the section "SECTION 4: CLUSTERING & INVERSION PREDICTION"), this column will include a set of symbols telling if the PEM maps to repeats or segdups in the reference genome (^ = repeats, : = segdups, and # = use of the same cluster for multiple inversion predictions). Please refer to 4.1 –Configuration File for further information.

colCoords_list: A string with the start and end coordinates of each of the reads in colLeftStartMin, colLeftEndMax, colRightStartMin and colRightEndMax above.

For example:

```
chr1+   66    251       1    1    224934035       0          112070201 112070201    112863834   112863834
        1               112070201,112070837,112070201,112070837,112863834,112864611,112863834,112864611

chr1+   67    252       1    1    234813248       0          117385848 117385848    117427400   117427400
        1               117385848,117386600,117385848,117386600,117427400,117427544,117427400,117427544

chr1+   68    259       1    1    283775150       0          141749949 141749949    142025201   142025201
        1               141749949,141750254,141749949,141750254,142025201,142025931,142025201,142025931
```

4.8.3 – Predicted inversions with wide breakpoints

One file per chromosome with the predicted inversions with wider breakpoint locations.

Filename convention: <chr>.inversions.wide

Each line corresponds to one predicted inversion. Columns are tab-separated in the following format:

colCluster+ID: An integer with the unique identifier for the positive cluster (PEMs mapping in the positive strand)

colCluster-ID: An integer with the unique identifier for the negative cluster (PEMs mapping in the negative strand)

colChr: A string with the chromosome where the inversion was predicted. (Format is according to UCSC)

colSupport: An integer with the total number of PEMs supporting the inversion

colBP1Start: An integer identifying the left-most coordinate of breakpoint 1

colBP1End: An integer identifying the right-most coordinate of breakpoint 1

colBP2Start: An integer identifying the left-most coordinate of breakpoint 2

colBP2End: An integer identifying the right-most coordinate of breakpoint 2

colBPsupport:
<BP1_filtered_support>(*<BP1_unfiltered_support>*)/*<BP2_filtered_support>*(*<BP1_unfiltered_support>*)

colTags: If requested in the Configuration File (see options *TAG_MAPPINGS_IN_REPEATS* and *TAG_MAPPINGS_IN_SEGDUPS* in the section "SECTION 3: FILTERING OF MAPPINGS", and options *TAG_INCONGRUENT_BPS* and *TAG_MULTIPLE_USE_OF_CLUSTERS* in the section "SECTION 4:

CLUSTERING & INVERSION PREDICTION"), this column will include a set of symbols telling if the PEM maps to repeats or segdups in the reference genome (^ = repeats, : = segdups, # = use of the same cluster for multiple inversion predictions, and ! = incongruent narrow BPs had to be corrected because startBP2 < endBP1). Please refer to 4.1 –Configuration File for further information.

For example:

```
26    34    chr1  4    245357627    245358524    245398436    245400877    2(2)/2(2)
29    NA    chr1  2    246729104    246735033    246754267    246776397    2(2)/0(0)
3     NA    chr1  18   246715485    246719487    246796677    246790926    18(18)/0(0)    !
```

4.8.4 – Predicted inversions with narrow breakpoints

One file per chromosome with the predicted inversions with narrower breakpoint locations.

Filename convention: <chr>.inversions.narrow

Format is equivalent to the previous file type. See **4.8.3 – Predicted inversions with wide breakpoints** for further details.

4.8.5 – Summary of PEMs, clusters and predicted inversions

One file with a summary of PEMs, clusters and predicted inversions in each chromosome (excluding scoring).

Filename convention: summaryresults.out

Each line corresponds to one analyzed chromosome. Columns are tab-separated in the following format:

colChr: A string with the chromosome. (Format is according to UCSC)

colPEMs: *<Number of PEMs in the positive strand>/ <Number of PEMs in the negative strand>*

colClusters: *<Number of clusters in the positive strand>/ <Number of clusters in the negative strand>*

colCorrClusters: *<Number of clusters that needed correction (tagged as !) in the positive strand>/ <Number of clusters that needed correction (tagged as !) in the negative strand>*

colInversions: *<Number of inversions predicted by both positive and negative clusters>/ <Number of inversions predicted by a positive cluster only>/ <Number of inversions predicted by a negative cluster only>*

colInconsistences: Number of inversions with discrepancies in the breakpoints location: *<A1>, <A2>, <A3>/<B1>, <B2>, <B3>/<C1>, <C2>, <C3>/<D1>, <D2>, <D3>/<E1>, <E2>, <E3>/<F1>, <F2>, <F3>*, where: A = Chromosome name does not exist in the UCSC for this species; B = Breakpoint 1 is outside of the range of the chromosome; C = Breakpoint 2 is outside of the range of the chromosome; D

= BP1start > BP1end; E = BP2start > BP2end; F = BP1end > BP2start; 1 = Definition 1 (basic) of the breakpoints (not reported in the GRIAL output files); 2 = Definition 2 (wide) of the breakpoints; 3 = Definition 3 (narrow) of the breakpoints. Please refer to 4.6 –Definition of the breakpoints for further information.

colTimestamp: Finished analyses on this chromosome on this timestamp (excluding scoring)

For example:

```
chr14   79/68   38/37   0/0     9/5/5   0,0,0/0,0,0/0,0,0/0,0,0/0,0,1/0,2,0     Thu Jan 16 10:57:37 2014
chr18   34/18   21/14   0/0     1/3/1   0,0,0/0,0,0/0,0,0/0,0,0/0,0,0/0,2,1     Thu Jan 16 10:57:37 2014
chr20   46/54   23/31   0/0     2/3/1   0,0,0/0,0,0/0,0,0/0,0,0/0,0,0/0,2,0     Thu Jan 16 10:57:37 2014
```

4.8.6 – Scored inversions

One file per chromosome with the scored inversions.

Filename convention: <chr>.inversions.<narrow|wide>.genotyped

Each section, delimited by a line of dash symbols, corresponds to one predicted inversion. For each inversion, the first line contains basic statistics in tab-separated columns in the following format:

colCluster+ID: An integer with the unique identifier for the positive cluster (PEMs mapping in the positive strand)

colCluster-ID: An integer with the unique identifier for the negative cluster (PEMs mapping in the negative strand)

colChr: A string with the chromosome where the inversion was predicted. (Format is according to UCSC)

colSupport: An integer with the total number of PEMs supporting the inversion.

colBP1Start: An integer identifying the left-most coordinate of breakpoint 1

colBP1End: An integer identifying the right-most coordinate of breakpoint 1

colBP2Start: An integer identifying the left-most coordinate of breakpoint 2

colBP2End: An integer identifying the right-most coordinate of breakpoint 2

colBPsupport:
<BP1_filtered_support>(*<BP1_unfiltered_support>*)/*<BP2_filtered_support>*(*<BP1_unfiltered_support>*)

colConcordants: An integer with the number of concordant PEMs bridging the breakpoints (BP1+BP2). Please refer to 4.7 –Scoring for further information.

colDCRatio: A float with the DC ratio Expected/Observed for the Concordant + Discordant PEMs. Please refer to 4.7 –Scoring for further information.

colDCTest: A float with the DC test P-value for the Concordant + Discordant PEMs. Please refer to 4.7 –Scoring for further information.

colDSLambda: A float with the lambda for the DS score (expected number of discordant PEMs supporting one breakpoint of the inversion). Please refer to 4.7 –Scoring for further information.

colDSMapabilityBP1: A float with the BP1 mapability for the DS score (% of the region of BP1 mapable by PEMs). Please refer to 4.7 –Scoring for further information.

colDSMapabilityBP2: A float with the BP2 mapability for the DS score (% of the region of BP2 mapable by PEMs). Please refer to 4.7 –Scoring for further information.

colDSLambdaBP1: A float with the BP1 lambda corrected by the BP1 mapability for the DS score (expected number of discordant PEMs supporting BP1 of the inversion). Please refer to 4.7 –Scoring for further information.

colDSLambdaBP2: A float with the BP2 lambda corrected by the BP2 mapability for the DS score (expected number of discordant PEMs supporting BP2 of the inversion). Please refer to 4.7 –Scoring for further information.

colDSScoreBP1: A float with the DS BP1 score (probability that the inversion is supported by the observed number of discordant PEMs or less in BP1). Please refer to 4.7 –Scoring for further information.

colDSScoreBP1: A float with the DS BP2 score (probability that the inversion is supported by the observed number of discordant PEMs or less in BP2). Please refer to 4.7 –Scoring for further information.

colDSScoreTotal: A float with the DS Total score (probability that the inversion is supported by the observed number of discordant PEMs or less). Please refer to 4.7 –Scoring for further information.

After this, a number of lines starting with > follow. For these lines, columns are tab-separated in the following format:

col>Symbol: > symbol identifying this type of lines

colLibraryID: A string with a library ID having discordant PEMs supporting this inversion

colLibraryConc: An integer with the number of concordant PEMs for the current library bridging the breakpoints of the inversion

colLibraryDisc: An integer with the number of discordant PEMs for the current library supporting the inversion

colLibraryLikely: A float with the likelihood of this individual being heterozygous for the inversion. Please refer to 4.7 –Scoring for further information.

For example:

```
10      37      chr1    8       25515013    25532113    25571051    25577607    6(6)/2(2)    33      2.5625    -3.05981055433589
        8.82637665606407        0.891762696372936    0.897874871032482    7.87103344601483    7.92498180174763
        0.329378477473795       0.0145823343072942   0.0245609882395128

>       ABC11   19      1       0.00238455736529416

>       ABC7    14      7       6.63884801721228e-07
```

Chapter 5

Frequently Asked Questions (FAQs)

5.1 –What does GRIAL do?

GRIAL is an algorithm for predicting inversions from paired-end mapping (PEM) data. It is based on some geometrical rules that are specific to inversions. GRIAL also refines the breakpoints location to the shortest possible region given the uncertainty inherent to the data. For more information on GRIAL please refer to 1.1 –What is GRIAL? or the GRIAL publication: Martínez-Fundichely et al. (in preparation).

5.2 –What do we mean by "breakpoints" in the results of GRIAL?

GRIAL clusters together all PEMs that can be supporting the same inversion breakpoint according to the set of geometrical rules described in 4.4 –Clustering and the GRIAL publication. According to the mapping position of these PEMs in the reference genome, GRIAL predicts and refines the location of the inversion breakpoints. Please refer to 4.5 –Prediction of inversions and 4.6 –Definition of the breakpoints for further information.

Sometimes several discordant PEMs might be compatible with multiple possible inversions. In this case, GRIAL clusters the PEMs such that they predict the minimum number of inversions with the highest PEM support. Please refer to 4.4 –Clustering for further information.

Also in some cases a predicted inversion is supported by discordant PEMs in only one breakpoint. In this case GRIAL infers the location of the other breakpoint by applying some geometrical rules compatible with the inversion prediction, but in this case the location of this inferred breakpoint is not refined. Please refer to 4.6 –Definition of the breakpoints for further information.

For more information and illustration of this, please refer to the corresponding sections of this manual or the GRIAL publication.

5.3 –Can GRIAL be used to predict other types of structural variants?

NO. The geometrical rules implemented in GRIAL are specific to the prediction of inversions from PEM data. However, the theoretical framework developed and implemented in GRIAL might be extended to other structural variants. See Escaramís *et al.* 2013.

5.4 –Why running GRIAL on my PEM data is taking so much time?

The running time of GRIAL not only depends on the total number of PEMs being analyzed, but also on the density of PEMs in specific regions of the genome. Quite often, highly complex regions of the genome (especially centromeres and satellite regions) contain errors or gaps in the reference assembly. These regions tend to have a high density of discordant PEMs that are mostly unreliable. To reduce the running time of GRIAL we suggest avoiding these artifactual regions by filtering PEMs mapping into satellite regions, low-complex regions or simple-repeats (*DISCARD_MAPPINGS_IN_REPEATS=yes*, *DISCARD_REPEATS_LIST=EIX*). Please refer to 4.1 –Configuration File for further information.

The use of concordant PEMs, either to compute the statistics of the library prior to the inversion prediction or to score the predicted inversions, also increments the running time of GRIAL several-fold. If your PEM data is huge, we suggest avoiding the use of concordant PEM data by providing the library statistics (*CONCORDANTS=0* and metrics provided in the section "SECTION 2: LIBRARY STATISTICS & ASSEMBLY" of the Configuration File) and not scoring the predicted inversions (*SCORE_INVERSIONS=no*). Please refer to 4.1 –Configuration File for further information.