CNV detection from genotyping arrays

Pipeline

Lucia Trastulla, PhD student
Laura Jiménez Barrón, PhD student

Max Planck Institute for Psychiatry,
Ziller lab

Input

- Genotyping intensities (.idata files)
- Manifest file provided by Illumina (.bpm or.csv file) depending on the chip
- Cluster file provided by Illumina (.egt file) depending on the chip
- Sample sheet file (Project name and manifest info, sample ID, Sample Plate, Sample Well SentrixBarcode_A, SentrixPosition_A, additional info)

Required program

- GenomeStudio for Genotyping with PLINK Input Report Plug-in v2.1.4
- R with the following packages installed: argparse, doParallel, ggplot2, RColorBrewer, ggrepel, ComplexHeatmap, circlize, grid, gridExtra
- Plink2
- BCFtools
- bedtools
- VCFtools

1 Genome Studio (GS) module

1. Load the samples from the intensities file and the provided Illumina clustering file (.egt), SNPs are automatically clustered so there is no need to use the command ”Cluster all SNPs” in ”Analysis”.
2. In the ”Full Data Table” window, insert the subcolumns B Allele Frequencies (BAF) and Log R Ratio (LRR) (automatically loaded for each sample) and export the table.
3. In the ”Samples Table” window, compute the Call Rate for all samples and export the table.
4. Use PLINK from the GS plug-in and save the report.

5. In the ”Samples Table” window, if samples with Call Rate < 0.98 are present, exclude them and update SNPs statistic.
6. Export ”SNP Table”.

Computational time: around 1 hour for 96 samples
Output:

- Full_Data_Table.txt
- Samples_Table.txt
- SNP_Table.txt (computed only from samples with acceptable quality)
- genomestudio_ouptut.map and genomestudio_output.ped (from PLINK)

2 SNPs and Samples Quality control

Script: QualityControl_GT.sh

```
bash QualityControl_GT.sh <folder with R scripts> <manifest file>
<file with PAR> <output/input folder> <n. cores parallelization>
```

Main steps:

1. Identify SNPs in Pseudoautosomal Regions (PARs).
2. Run the quality control script, exclude SNPs that do not satisfy the criteria, export LRR and BAF and Genotype (GT) from GenomeStudio output.
3. Plot samples call rate before and after quality control (sex inferred from the data).

Note:

- Detailed explanation of the quality control protocol in ”Genotyping_Data_QC.pdf”
- PAR_Coord_GRCh37.txt obtained from here and has the following format (tab separated)

Region GRCh37 Coordinates			
Chr	PAR_type	Start	End
X	PAR1	60001	2699520
X	PAR2	154931044	155260560
Y	PAR1	10001	2649520
Y	PAR2	59034050	59363566

Output:

- PAR_SNPs.txt (indicates whether a SNP is located in a PAR or not)
- info_QC.txt (info on the SNPs filtered)
- Full_Data_Table_filt.txt

- SNP_Table_filt.txt
- Samples_Table_filt.txt
- GT_table.txt (genotype for each sample)
- GT_match.txt (n.samples × n.samples, percentage of matched genotype between samples)
- LRR_table.txt (LRR for each sample)
- BAF_table.txt (BAF for each sample)
- plot_CR_samples.pdf/.png (call rate plots before and after QC)

Computational time: around 1.5 hours for 96 samples

3 Samples Mismatch Control

Write an annotation file for the samples in the following format (save as <project-name>_annotation_file.csv)

ID	Type	Sample_name	SentrixBarcode_A	SentrixPosition_A	Note
SCZ5_fibro	PRE	SCZ5	201715380135	R08C01	
SCZ5_cl1+	POST	SCZ5	201715380135	R12C01	

- **ID**: SampleID in the original Sample sheet file
- **Type**: PRE (for original material) or POST (for reprogrammed)
- **Sample_name**: identifier for material that comes from the same sample
- **SentrixBarcode_A** and **SentrixPosition_A** from the original Sample sheet
- **Note**: additional info, explained later

Then run the following bash script
Script: GT_match.sh

```
bash GT_match.sh <folder with R scripts> <output/input folder>
<project name (e.g. M00940)> <annotation file manually created>
```

Main steps:

1. Compute the percentage of genotype match for each pair of cell lines and plot a heatmap showing the pairing between corresponding cell lines as indicated in the provided annotation file.
2. If the labelling of the lines is not concordant with the GT match, then sample names are changed to ”mismatch” and the heatmap is plotted again with the correct labelling.

Note:

- 96% of matching genoptypes is required to classify cell lines as coming from the same sample.
- In order to exclude certain cell lines from the analysis, simply do not include them in the annotation file.
- The final annotation file can be manually edited in the column ”Sample_name_new” if the correct sample labelling is known, otherwise the mislabelled samples will be identified as ”MISMATCHED_SAMPLE”.
- A column indicating the Gender is added in the new annotation file.

Output:

- <project name>_hm_GT.pdf/.png (heatmap with original labelling).
- <project-name>_hm_GT_correct.pdf/.png (heatmap with correct labelling).
- <project-name>_annotation_file_GTmatch.csv (updated annotation file with the correct labelling).

4 Prepare sample files

Manually create a .txt file that contains the samples of interest (one sample name in each row). Then run the following bash script:
Script: Create_SampleID.sh

```
bash Create_SampleID.sh <folder with R scripts> <output/input folder>
<updated annotation file> <samples names manually created>
```

Main steps:
For each sample name (e.g. SCZ5) in the samples names file, create a .txt file that contains in the first row the gender of the sample, second row the SentrixBarcode_SentrixPosition id for the PRE reprogramming lines, from the third to the last row the SentrixBarcode_SentrixPosition id for the POST (reprogramming) lines.
Note:

- The script uses updated Sample Name in ”Sample_name_new” column of the new annotation file.
- If the aim is to compare two lines from the same donor both PRE or POST reprogramming, the ”Type” entry can be changed and the info is reported in ”Note”, as in the following example.

ID	Type	Sample_name	SentrixBarcode_A	SentrixPosition_A	Note	Sample_name_new
Ul1_pool1	PRE	UL1	201694380133	R01C01	POST ann as PRE	UL1

- If there are more than one PRE line for the same donor, more than one .txt file will be generated, one for each PRE line. In this way, the comparison of the POST lines with all the PRE lines present is performed.
- If only a line for a sample is present, the single CNV analysis will be performed afterwards (see ”Single analysis” section). If this line is not annotated as ”PRE” in ”Type”, then change to ”PRE”.

Output:
ID_<sample_name>_<n.PRE>.txt where the ”n.PRE” refers to the number of PRE reprogramming lines considered (e.g ID_SCZ5_1.txt if only one ”PRE” is present in the annotation file for the sample SCZ5).

5 Prepare VCF file

5.1 Obtain REF/ALT annotation

Script: GetRefAlt.sh

```
bash GetRefAlt.sh <manifest file .csv> <fasta file>
<number of header lines in the manifest> <number of tail lines in the manifest>
```

Main steps:
Prepare the manifest file and extract the reference allele from the reference genome fasta file.
Note:

- Fasta file can be downloaded from here (version hg19).
- Header and tail lines in the manifest file are the lines at the beginning and end that do not have a SNP record but other info.
- <number of header lines> comprehends both not relevant lines and the header itself

Output:
<manifest file>_reference_alleles file: chromosome | position | reference allele.

5.2 Create VCF file

Script: CreateVcf.sh

```
bash CreateVcf.sh <otuput folder> <path location plink>
<path and common name GenomeStudio ouput .map .ped files>
```

Main steps:
Create the vcf file from the .map and .ped files output of the PLINK plug-in in GenomeStudio.
Output:

- plink.vcf
- plink.log
- plink.nosex

5.3 Correct REF/ALT

Script: CorrectAltRefVCF.sh

```
bash CorrectAltRefVCF.sh <vcf file> <reference alleles file>
```

Main steps:

- Correct REF/ALT in the .vcf file based on the <manifest file>_reference_alleles file.
- Exclude the mitochondrial (MT) and XY SNPs in the .vcf file.

Output:
plink.vcf_RefAltCorrected.vcf (updated vcf file)

5.4 Annotate VCF file

Script: Preproc_cnv.sh

```
bash Preproc_cnv.sh <output/input folder>
```

Main steps:

- Filter updated .vcf file to consider only SNPs that passed the quality control.
- Annotate .vcf file with the BAF and the LRR previously extracted.

Computational time: for 96 samples 10 min circa
Output:

- data_BAF_LRR.vcf file with LRR and BAF in "FORMAT" and correct REF/ALT annotation for the quality control SNPs.
- SNPs_QC.txt file containing SNPs names that pass the quality control.

6 CNV detection: Pairwise comparison

6.1 Estimates CNVs

Script: Cnv_Analysis.sh

```
bash Cnv_Analysis.sh <input folder> <SampleID file>  <Sample name>
```

Main steps:
Pairwise cnv prediction via bcftools for the specified sample name. Compare the PRE line with each of the POST lines in the SampleID file (e.g ID_SCZ5_1.txt).
Note:

- The analysis is only for the sample specified.
- The input folder is the one containing data_BAF_LRR.vcf file.
- SampleID file is one of the output for the script Create_SampleID.sh

Output: For each of the POST lines the following folder is generated:
outdir_<sample name>_<id_pre>/output_<id_post> which contains

- dat.<id_pre>.tab and dat.<id_post>.tab input LRR and BAF
- cn.<id_pre>.tab and cn.<id_post>.tab probability for each copy number state
- summary.<id_pre>.tab and summary.<id_post>.tab summary of copy number state with quality of prediction, n. of markers and n. of heterozygous sites
- summary.tab combination of single summary files

6.2 Quality control CN and plot different CN detected

Script: Cnv_Diff.sh

```
bash Cnv_Diff.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name> <annotation file>
```

Main steps:

- Filter out detected CN of poor quality. The criteria are based on Kilpinen et al.:

  1. quality score lower than 2,
  2. CN length lower than 200kb,
  3. number of sites lower than 10 for a deletion,
  4. number of heterozygous sites lower than 10 for a duplication.

  To change these quality control parameters, modify input in summary_diffCN_run.R

- Plot number of different CNV prediction in each chromosome for each comparison (PRE vs POST lines for the sample considered).

Output:

- summary_pair_<sample name>_<n.PRE>_<id_post>.tab summary info combined for the comparison, last column indicates whether the region must be filtered or not.
- <sample name>_CNV_diff.txt for each comparison and chromosome, gives the number of different CN detected between the PRE and the POST reprogramming lines.
- <sample name>_CNV_diff.pdf/.png plot of the previous table
- summary_QC.<id_pre>.tab and summary_QC.<id_post>.tab summary of copy number state with filter of specific region due to quality control procedure.

6.3 Plot detected CN on the entire genome

Script: Cnv_Plot.sh

```
bash Cnv_Plot.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name>
```

Main steps:
For a sample, plot the CNVs detected in each line for all the chromosomes.
Note:

- Only the CN different from 2 that pass the quality control are plotted, centromeres are excluded from the analysis.
- The CN plotted are the one detected in each pairwise comparison, for the PRE line the union of the detected CN in each comparison is plotted.

Output:

- cnv_<sample name>_<n.PRE>_QC_summary.txt for a sample and a PRE line associated to it, contains the prediction of the CN status in each of the comparison PRE-POST.
- cnv_<sample name>_<n.PRE>_QC.pdf/png plot of the CN status predicted in each comparison. The first line indicates whether a difference in the CN prediction is detected.

6.4 Plot LRR and BAF

Script: Cnv_Plot_LRR-BAF.sh

```
bash Cnv_Plot_LRR-BAF.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name> <annotation file>
<chromosome to be plotted (optional)>
```

Main steps:
For a sample and for each comparison PRE-POST, plot LRR BAF and detected CN for a specific chromosome.
Note:

- All the CN are plotted, even the one that do not pass the quality control.
- The chromosome to be investigated can be specified, otherwise 24 plots are generated, one for each chromosome.

Output:
plot_<id_pre>_vs_<id_post>_chr<n.chr>.png

7 CNV detection: Single analysis

7.1 Estimates CNVs

Script: Cnv_Analysis_Single.sh

```
bash Cnv_Analysis_Single.sh <input folder> <SampleID file>  <Sample name>
```

Main steps:
Single CNV prediction via bcftools for the specified sample name. Detect CNV for the single line present in the SampleID file.
Note:

- The analysis is only for the sample specified.
- The input folder is the one containing data_BAF_LRR.vcf file.
- SampleID file is one of the output for the script Create_SampleID.sh

Output: The following folder is generated:
outdir_<sample name>_<id_line>/ which contains

- dat.<id_line>.tab input LRR and BAF
- cn.<id_line>.tab probability for each copy number state
- summary.<id_line>.tab summary of copy number state with quality of prediction, n. of markers and n. of heterozygous sites

7.2 Quality control CN and plot number CN detected

Script: Cnv_Det_Single.sh

```
bash Cnv_Det_Single.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name>
```

Main steps:

- Filter out detected CN of poor quality. The criteria are based on Kilpinen et al. (see 6.2). To change quality control parameters, modify input in summary_CNSingleAnalysis_run.R
- Plot number of deletion and duplication detected for the considered line.

Output:

- CNV_detection_<sample name>_<id_line>.txt for each chromosome, gives the number of duplication a deletion detected.
- CNV_detection_<sample name>_<id_line>.pdf/.png plot of the previous table.
- summary_QC.<id_line>.tab summary of copy number state with filter of specific region due to quality control procedure.

7.3 Plot detected CN on the entire genome

Script: Cnv_Plot_Single.sh

```
bash Cnv_Plot_Single.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name>
```

Main steps:
For a sample, plot the CNVs detected in the line for all the chromosomes.
Note:
Only the CN different from 2 that pass the quality control are plotted, centromeres are excluded from the analysis.
Output:
cnv_<sample name>.pdf

7.4 Plot LRR and BAF

Script: Cnv_Plot_LRR-BAF_Single.sh

```
bash Cnv_Plot_LRR-BAF_Single.sh  <folder with R scripts> <input/output folder>
<SampleID file> <Sample Name> <annotation file>
<chromosome to be plotted (optional)>
```

Main steps:
For a sample, plot LRR, BAF and detected CN for a specific chromosome.
Note:

- All the CN are plotted, even the one that do not pass the quality control.
- The chromosome to be investigated can be specified, otherwise 24 plots are generated, one for each chromosome.

Output:
plot_<id_line>_chr<n.chr>.png

8 Merge two projects

The aim is to merge two complete vcf files (with LRR and BAF annotation) in order to compare lines from the same sample that are present in two different projects.

1. **Script:** Merge_vcf.sh

   ```
   bash Merge_vcf.sh  <output folder> <folder project 1> <folder project 2>
   ```

   Main steps:

   - Find common set of SNPs between the two projects after the quality control procedure.
   - Filter each .vcf file considering only the common set of SNPs.
   - Merge the two filtered .vcf files.

   Note:
   This step can be performed only if steps 1,2 and 5 have been executed for both projects.
   Output:

   - comm_SNPs.txt set of common SNPs.
   - data_BAF_LRR.vcf merged .vcf file.

   Computational Time: For 2 projects of 96 samples each, 10 min circa

2. Manually create a .txt file that contains the names of the samples of interest (one in each row). Then run the following bash script:
   Script: GT_match_merge.sh

```
bash GT_match_merge.sh  <folder with R scripts> <output folder>
<folder project 1> <folder project 2> <annotation file project 1>
<annotation file project 2> <samples names manually created> <n.of cores>
```

Main steps:

- Compute GT match for the samples of interest using the GT tables from the two projects.
- Plot heatmap based on the match.
- Create an annotation file containing the info for the sample considered and correct the name if there is any mismatch.

Note:
The sample name column in the annotation files used as input is the original one and not the updated.
Output:

- GT_comp.txt
- Merged_hm_GT.pdf/.png (heatmap with original labelling)
- Merged_hm_GT_samplename.pdf/.png (heatmap with original labelling, labels are ID names)
- Merged_annotation_file_GTmatch.csv (updated annotation file with correct labelling, another column is added to indicate different projects)

3. Steps 4, 6 and 7 can be now performed using merged annotation file and merged vcf file.

9 Merge multiple projects

The aim is to merge multiple vcf files (with LRR and BAF annotation) in order to compare lines from the same donor and check if donor annotation is correct.

1. **Script:** Merge_multiple_vcf.sh

   ```
   bash Merge_multiple_vcf.sh  <output folder> <file with input folders>
   ```

   Main steps:

   - Find common set of SNPs between all projects after the quality control procedure.
   - Filter each .vcf file considering only the common set of SNPs.
   - Merge all filtered .vcf files.

   Note:
   This step can be performed only if steps 1,2 and 5 have been executed for all projects.
   Output:

   - comm_SNPs.txt set of common SNPs.
   - data_BAF_LRR.vcf merged .vcf file.

   Computational Time: depends on number of projects and samples

2. Manually create a .txt file that contains the names of the samples of interest (one in each row). Then run the following bash script:
   Script: GT_match_merge_multiple.sh

   ```
   bash GT_match_merge_multiple.sh  <folder with R scripts> <output folder>
   <file with folders name> <file with annotation files name> <samples names ma
   ```

   Main steps:

   - Compute GT match for the samples of interest using the GT tables from the all projects.
   - Plot heatmap based on the match.
   - Create an annotation file containing the info for the sample considered and correct the name if there is any mismatch.

   Note:
   The sample name column in the annotation files used as input is the original one and not the updated.
   Output:

   - Merged_GT_comp.txt
   - Merged_hm_GT.pdf/.png (heatmap with original labelling)
   - Merged_hm_GT_samplename.pdf/.png (heatmap with original labelling, labels are ID names)
   - Merged_annotation_file_GTmatch.csv (updated annotation file with correct labelling, another column is added to indicate different projects)

3. Steps 4, 6 and 7 can be now performed using merged annotation file and merged vcf file.

10 Example Complete pipeline

To see an example of the complete pipeline, check example_pipeline_June2020.txt and example_pipeline_Oct19_June20.txt (merged 2 projects) and example_pipeline_merged_multiple.txt (merge multiple projects)