GeMo

Release 1.0.0

Summo

Feb 03, 2023

CONTENTS:

GeMo is a WebApp to represent Genome Mosaics with current focus on plants. However, GeMo is developed in a generic way it can be also applied to other organisms.

CHAPTER

ONE

INTRODUCTION

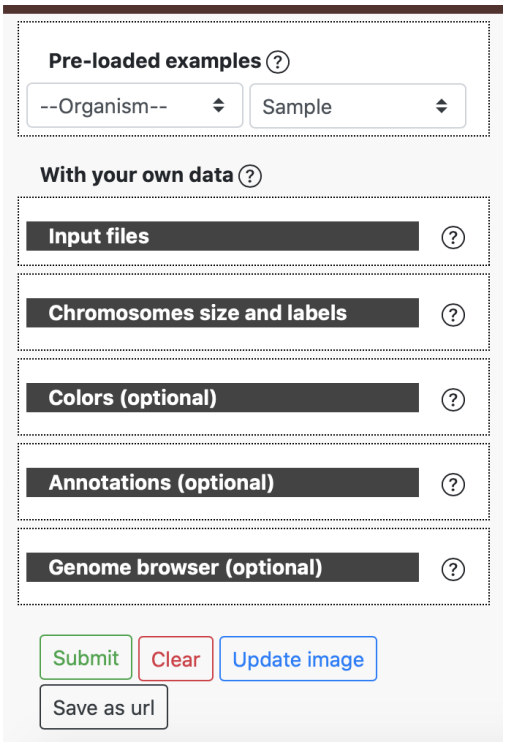
GeMo is a WebApp to represent Genome Mosaics with current focus on plants. However, GeMo is developed in a generic way it can be also applied to other organisms.

1.1 Main features

- Dynamic chromosome painting visualisation
- Online Data curation of mosaic prediction
- Markers or Genes Plots on mosaic karyotypes
- Data and high quality image export

1.2 Input formats

GeMo requires two types of datasets to generate the ideogram visualization



The position of the mosaic blocks along the chromosomes. It accepts two types of files:

- **Genomic blocks**

chr	haplotype	start	end	ancestral_group
chr01	0	1	29070452	g4
chr01	1	1	29070452	g4
chr02	0	1	29511734	g4
chr02	1	1	29511734	g4

- **Normalized curves**

chr	start	end	V	T	S
chr01	1145	189582	0.001671988	0.014082301	0.001638686
chr01	189593	356965	0.001244196	0.012867256	0.001810139
chr01	356968	488069	0.001117959	0.010035172	0.000759437
chr01	488097	633373	0.002678213	0.010470727	0.003896031

- **Chromosomes sizes and labels**

Chromosome data format, each column tab separated chr, len, centromereInf (optional), centromereSup (optional), label (optional)

chr	len	label
chr01	37945898	AB
chr02	34728925	AB
chr03	40528553	AB
chr04	34728925	AB
chr05	44598304	AB
chr06	46248384	AB
chr07	42818424	AB
chr08	38870123	AB

- **Optional files**

Users can provide their own color codes or use the online features (custom or color blind friendly palettes)

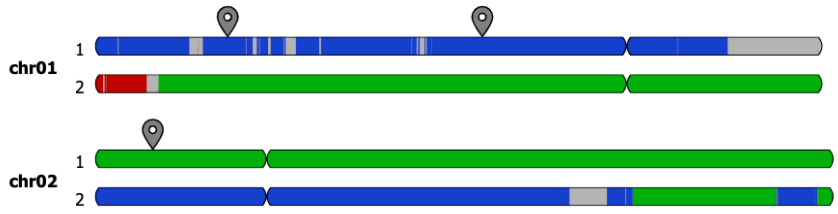
Color

group	name	hex
g1	group1	#000000
g2	group2	#ffc000
g3	group3	#1440cd
g4	group4	#00b009

Annotations

A list of genomic coordinates (e.g. genes of interest, QTLs) can be provided in a BED-like to visually spot the corresponding regions on the chromosomes. This can be particularly useful to check correlations between parental/ancestral blocks and genes/regions of interest.

chr01	5287838	5289224	gene	0	-
chr01	15485703	15486813	gene	0	+
chr02	2276353	2277821	gene	0	+

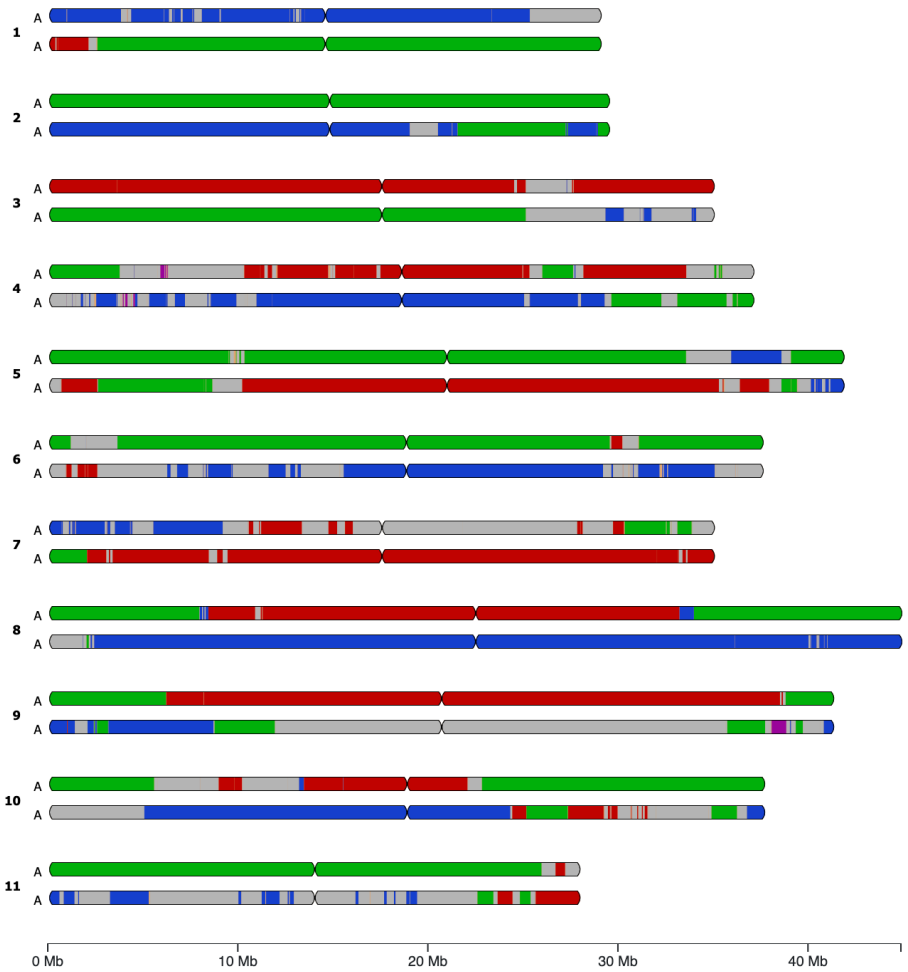


1.3 Data outputs

Once data is provided the chromosome diagram is generated on the fly. Chromosomes display colored blocks usually corresponding to their ancestral/parental origin. An interactive legend is present to label each group with a corresponding color. The user can modify the color of a group directly in the legend.

- **Blocks**

In the example below, the 11 chromosomes of an doploid organism is visualized. Three main colors (green, blue and red) are visible and corresponds to 3 distinct genepools that contributed to the genetic make up of this genotype. The segements in grey corresponds to unknown.



- **Curves**

In this mode, the graph represents the proportion of haplotypes of each ancestral origin along chromosomes. They are the results of a normalisation of the number of reads supporting each origin on a given window.

In this example, allelic ratio for a range of founding genepools are respresented by different colors for chromosome 1. Two genepools in green is the main contributor with smaller contribtuons from the blue and red gene pools.

1.4 Data curation and export

Uploaded datasets are automatically loaded in the text box of the GeMo menu, allowing users to update the content and reflect it on the image by clicking on the “update image” button.

In curve mode, users can visually set the threshold on the graph to recalculate the origin and size of clored block forming the mosacis. This can be particularly useful when multiple putative parental gene pools with unclear signals can create noisy mosaics or to switch segments from one haplotype to another for better consistency. Once a threshold is changed, the karyotype diagram is automatically updated.

For pre-loaded data, the curve mode can be activated only when the normalized curves dataset exists. In this case, a

toggle button labeled “Curve based mode” is present at the top of the user input form.

GeMo offers the possibility to download the latest version of the data sets and export the output graphics as SVG for publication purposes. In addition, data can be also stored temporarily online with a unique URL allowing to share it with multiple users.

1.5 Live demo

GeMo is available for free to use at https://gemo.southgreen.fr/ where anyone can upload its own data or test with pre-loaded mosaics/datasets.

1.6 Citation

Summo M, Comte A, Martin G, Weitz E, Perelle P, Droc G and Rouard M. GeMo: A mosaic genome painting tool for plant genomes. (in prep)

1.7 Acknowledgements

GeMo has been developed in the framework of the Genome Harvest project supported by the Agropolis fondation.

1.8 Troubleshootings and web browser compatibility

- Some issues were reported for color management when using the exported SVG with Inkscape.
- It is optimized for Chrome and works in Firefox and Edge but some design issues may occur with Safari.

The web interfaces were tested with the following platforms and web browsers:

OS	Version	Chrome	Firefox	Edge	Safari
Windows 10	10	88.0.4324.150	94.0.1	96.0.1054.29	n/a
Mac OS	11.2	97.0.4692.36	94.0.2	n/a	14.0.3

CHAPTER

TWO

QUICK START

The objective of this tutorial is to reproduce part of the results presented in Baurens et al (2019) and Ahmed et al (2019), using respectively VCFHunter and TraceAncestor.

The outputs of these programs can then be used in the GeMo webapps.

2.1 Installation requirements

This tutorial is developed to run on Linux or Apple (MAC OS X) operating systems. There are no versions planned for Windows.

Software requirements:

- Perl 5 for TraceAncestor
- Python 3 for VCFHunter

2.1.1 Testing your Perl installation

To test that Perl 5 is installed, enter on the command line

```
perl -version
```

2.1.2 Testing your Python installation

To test that Python 3 is installed, enter on the command line

```
python3 --version
```

Now, you can clone the repository, create a virtualenv and install several additionnal package using pip.

```
git clone https://github.com/gdroc/GeMo_tutorials.git
cd GeMo_tutorials
python3 -m venv $PWD/venv
source venv/bin/activate
pip install numpy
pip install matplotlib
pip install scipy
```

2.2 Download Dataset

For this tutorial, Dataset that will be used by TraceAncestor or by VCFHunter are accessible on Zenodo https://doi.org/10.5281/zenodo.6539270

To download this, you only need to launch the script download_dataset.pl without any parameter

```
perl download_dataset.pl
```

This script create a new directory data

```
data/
├── Ahmed_et_al_2019_color.txt
├── Ahmed_et_al_2019_individuals.txt
├── Ahmed_et_al_2019_origin.txt
├── Ahmed_et_al_2019.vcf
├── Baurens_et_al_2019_color.txt
├── Baurens_et_al_2019_individuals.txt
├── Baurens_et_al_2019_origin.txt
├── Baurens_et_al_2019_chromosome.txt
└── Baurens_et_al_2019.vcf
```

These files are require for this tutorial to run VCFHunter or TraceAncestor

2.3 Input

- **Baurens_et_al_2019_origin.txt** : A two column file with individuals in the first column and group tag (i.e. origin) in the second column

individuals	origin
P2	AA
T01	BB
T02	BB
T03	AA
T04	AA
T05	AA
T06	AA
T07	AA
T08	BB

- **Baurens_et_al_2019.vcf** : A vcf file with ancestral and admixed individuals

```
grep #CHROM data/Baurens_et_al_2019.vcf
#CHROM  POS  ID  REF  ALT  QUAL  FILTER  INFO  FORMAT  ACC48-FPG␣
↪  ACC48-FPN  ACC48-P_Ceylan  ACC48-Red_Yade  DYN163-Kunnan  DYN275-Pelipita␣
↪DYN359-Safet_Velchi  GP1  GP2  P1  P2  T01  T02  T03  T04  ␣
↪ T05  T06  T07  T08  T10  T11
```

- **Baurens_et_al_2019_individuals.txt** : A two column file with individuals to scan for origin (same as defined in the VCF headerline) in the first column and the ploidy in the second column.
- **Baurens_et_al_2019_color.txt** : A color file with 4 columns: col1=group and the three last column corresponded to RGB code.

group	name	r	g	b
AA	acuminata	0	255	0
BB	balbisiana	255	0	0

2.4 Run workflow using create_gemo_input.pl

```
perl create_gemo_input.pl --help
Parameters :
    -v, --vcf         A vcf file [required]
    -o, --origin      A two column file with individuals in the first column and group␣
↪tag (i.e. origin) in the second column [Required]
    -i, --individuals List of individuals to scan from vcf, as defined in the VCF␣
↪headerline [Required]
    -m, --method      Permissible values: vcfhunter traceancestor (String). Default␣
↪vcfhunter
    -c, --color       A color file with 4 columns: col1=group and the three last column␣
↪corresponded to RGB code.
    -t, --threads     Number of threads
    -d, --dirout      Path to the output directory (Default method option name)
    -h, --help        display this help
```

1. With VCFHunter method

You must use the dataset prefixed with Baurens_et_al.

```
perl create_gemo_input.pl --vcf data/Baurens_et_al_2019.vcf --origin data/Baurens_et_al_
↪2019_origin.txt --individuals data/Baurens_et_al_2019_individuals.txt --method␣
↪vcfhunter --color data/Baurens_et_al_2019_color.txt --threads 4
```

2. With TraceAncestor method

You must use the dataset prefixed with with Ahmed_et_al.

```
perl create_gemo_input.pl --vcf data/Ahmed_et_al_2019.vcf --origin data/Ahmed_et_al_2019_
↪origin.txt --individuals data/Ahmed_et_al_2019_individuals.txt --method traceancestor -
↪-color data/Ahmed_et_al_2019_color.txt
```

2.5 Explanation of outputs

A directory was create depending on parameter dirout (default method name)

For example, for VCFHunter, for each individual present in the file data/Baurens_et_al_2019_individuals.txt, 4 outputs are produced in this directory, prefixed with the name of indivual :

- **DYN163-Kunnan_ideo.txt** : A text file of the position of genomic blocks the ancestry mosaic with a succession of genomic blocks along the chromosome

chr	haplotype	start	end	ancestral_group
chr01	0	0	20888	AA
chr01	0	20888	451633	AA
chr01	0	451633	848109	AA
chr01	0	848109	1198648	AA
chr01	0	1198648	1555128	un
chr01	0	1555128	1899887	AA
chr01	0	1899887	2296417	un
chr01	0	2296417	2759817	un

- **DYN163-Kunnan_chrom.txt** : A tab file with name, length and karyotype based on ploidy (optionaly the location of centromere).

chr	len	centromereInf	centromereSup	label
chr01	29070452	14535226	14535228	AB
chr02	29511734	14755867	14755869	AB
chr03	35020413	17510206	17510208	AB
chr04	37105743	18552871	18552873	AB
chr05	41853232	20926616	20926618	AB
chr06	37593364	18796682	18796684	AB
chr07	35028021	17514010	17514012	AB
chr08	44889171	22444585	22444587	AB
chr09	41306725	20653362	20653364	AB
chr10	37674811	18837405	18837407	AB
chr11	27954350	13977175	13977177	AB

- **BDYN163-Kunnan_color.txt** : Frequency of ancestors alleles along chromosome for the particular hybrid focused.

group	name	hex
AA	acuminata	#00ff00
BB	balbisiana	#ff0000
un	un	#bdbdbd

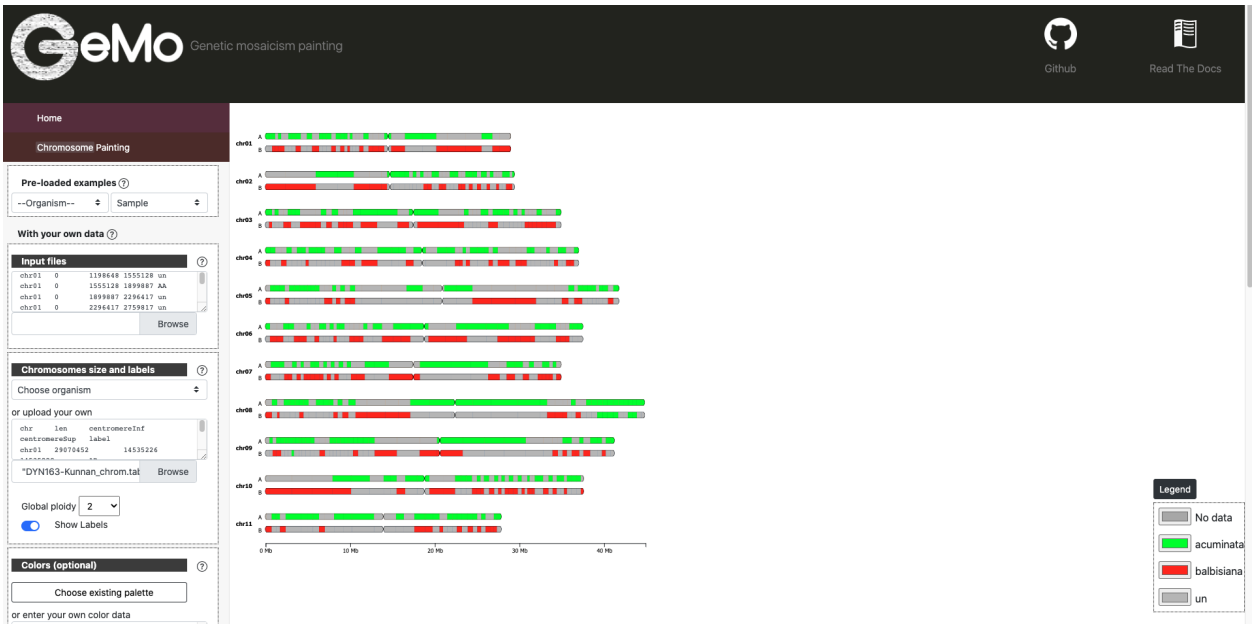
- **DYN163-Kunnan_curve.txt** : Frequency of ancestors alleles along chromosome for the GeMo visualization tool.

chr	start	end	AA	BB
chr01	20888	525207	0.660757486645395	0.30378982223766354
chr01	525207	1086954	0.6425583592191819	0.3508607451997505
chr01	1086954	1563263	0.7355412887547506	0.2661255866893344
chr01	1563263	2058335	0.6136974042002844	0.3851682528896984
chr01	2058335	2638987	0.5543371247412866	0.39469329280411
chr01	2638987	3190388	0.6752108036341729	0.3208947817296506
chr01	3190388	3905155	0.6951554613138214	0.3155181655339866
chr01	3905155	4800522	0.6813746934348566	0.32271710110143237

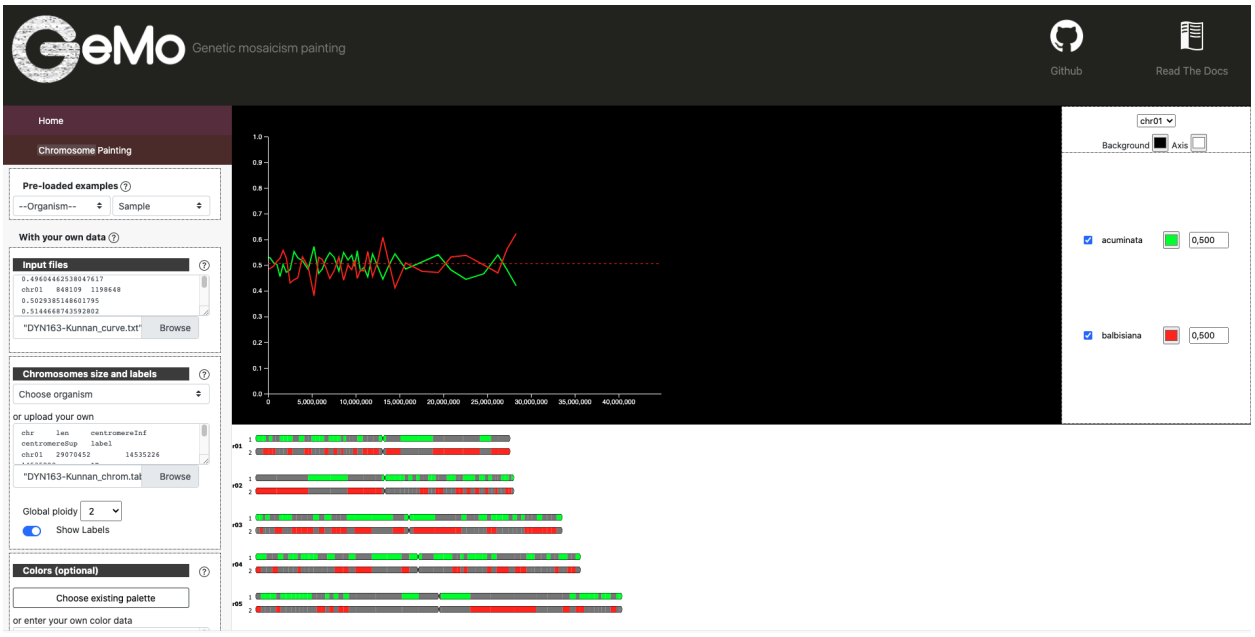
2.6 Visualization and block refinement with GeMo

Go to GeMo WebApp

- **Ideogram Mode**



- **Curve mode**



2.7 References

- Summo, Marilyne. (2022). GeMo : a web-based platform for the visualization and curation of mosaic genomes [Data set]. Zenodo.
- Baurens,F.-C. et al.(2019) Recombination and Large Structural Variations Shape Interspecific Edible Bananas Genomes. Mol Biol Evol, 36, 97–111.
- Martin et al., 2020a. Martin G, Cardi C, Sarah G, Ricci S, Jenny C, Fondi E, Perrier X, Glaszmann J-C, D'Hont A, Yahiaoui N. 2020. Genome ancestry mosaics reveal multiple and cryptic contributors to cultivated banana. Plant J. 102:1008–1025.
- Ahmed,D. et al. (2019) Genotyping by sequencing can reveal the complex mosaic genomes in gene pools resulting from reticulate evolution: a case study in diploid and polyploid citrus. Annals of Botany, 123, 1231–1251.

CHAPTER

THREE

CHROMOSOME PAINTING USING NON ADMIXED ANCESTRAL ACCESSIONS (VCFHUNTER)

The aims of this tutorial are to showing how data should be processed to be then visualized with the GeMo

3.1 Installation

Install VCFHunter following the documentation presented above:

```
git clone https://github.com/gdroc/GeMo_tutorials.git
cd GeMo_tutorials
python3 -m venv $PWD/venv
source venv/bin/activate
pip install numpy
pip install matplotlib
pip install scipy
```

3.2 Download datasets

Two ways :

- Download Baurens_et_al_2019.zip available on Zenodo

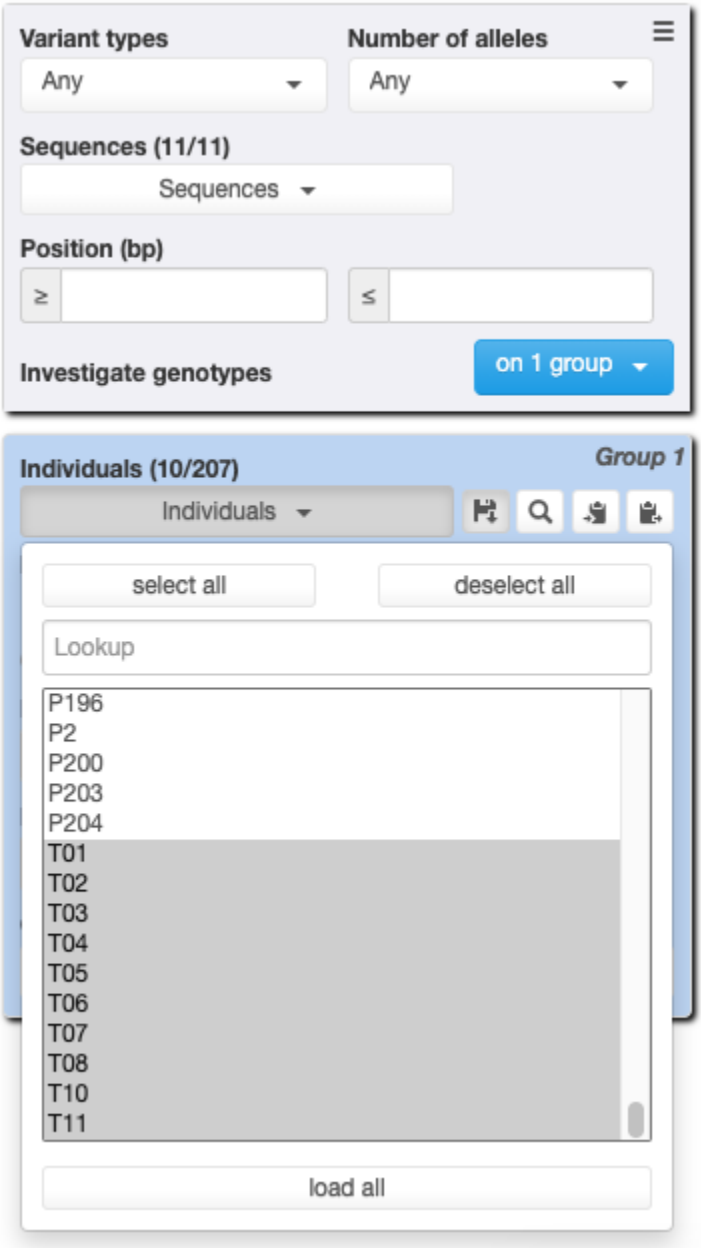
```
mkdir data
cd data
wget https://zenodo.org/record/6542870/files/Baurens_et_al_2019.zip
unzip Baurens_et_al_2019.zip
ls Baurens_et_al_2019.vcf > Vcf.conf
```

Goto *Identification of private alleles and formatting output for more analysis*

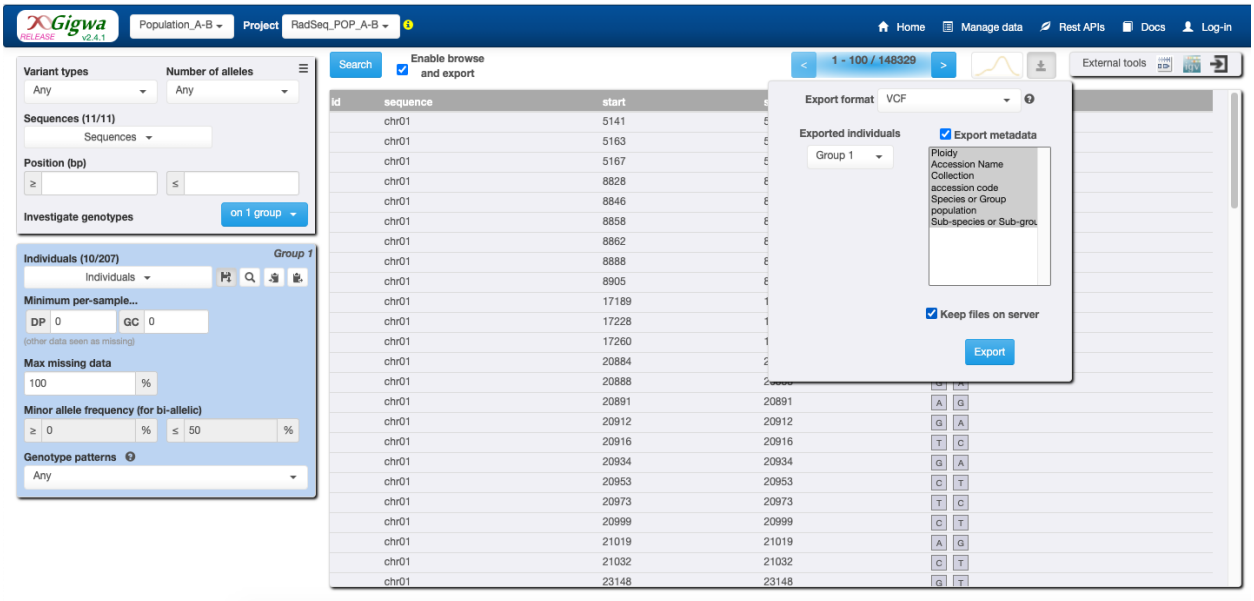
- Create input dataset using Gigwa, a web application for managing and exploring high-density genotyping data, to download a VCF

Select the database Populations_A_B

Select the accessions P2 and T01 to T11 on the Indivuals drop down menu, and click on Search button



Download result (check radio "Export Metadata" and "Keep file on servers")



Copy the link, and create a repository on your terminal

```
mkdir data
cd data
wget --no-check-certificate https://www.crop-diversity.org/gigwa/genofilt/tmpOutput/
↪anonymousUser/b429763f507dc1bb2b169d7da5cf1804/Population_A-B__project1__2021-10-12__
↪148329variants__VCF.zip
unzip Population_A-B__project1__2021-10-12__148329variants__VCF.zip
cut -f 1,6 Population_A-B__21individuals_metadata.tsv > Baurens_et_al_2019_origin.txt
sed -i 's:balbisiana:BB:' Baurens_et_al_2019_origin.txt
sed -i 's:acuminata:AA:' Baurens_et_al_2019_origin.txt
ls Population_A-B__148329variants__10individuals.vcf > Vcf.conf
```

VCF content

```
grep "^#CHROM" Population_A-B__148329variants__21individuals.vcf
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO    FORMAT  ACC48-FPG   ACC48-FPN   ACC48-P_
↪Ceylan  ACC48-Red_Yade  DYN163-Kunnan   DYN275-Pelipita DYN359-Safet_Velchi GP1 GP2 P1␣
↪ P2  T01 T02 T03 T04 T05 T06 T07 T08 T10 T11
```

3.3 Workflow

The principle of this analysis is to :

- Identify specific allele of distinct genetic pools,
- Calculate the expected allelic ratio of these alleles in these genetic pools,
- Calculate the observed allelic ratio a/several given accessions
- Normalize these observed ratios using expected ratio to infer the number of haplotypes of each genetic pools that are present on a given windows of the studied accession.

Files obtained at the end of the process can be given to GeMo tools to visualize data and optimize parameters.

Input

- Baurens_et_al_2019_origin.txt
- Vcf.conf is a file which contained path to vcf files which will be used for e-chromosome painting.
- Baurens_et_al_2019_chromosome.txt (tabulated file with the chromosome name and length)
- Baurens_et_al_2019_color.txt

group	name	r	g	b
AA	acuminata	0	255	0
BB	balbisiana	255	0	0

3.3.1 Identification of private alleles and formatting output for more analysis

```
bin/IdentPrivateAllele.py -c data/Vcf.conf -g Baurens_et_al_2019_origin.txt -o step1 -a␣
↪y  -m y
```

In this first step, the program use genotyping information provided in vcf files passed in *Vcf.conf* file and the file *Origin.tab* containing the corresponding genetic pools of some accessions of the vcf to identify alleles specific of each pools.

Outputs can be found in directory passed in *-o* option. For each accessions identified as belonging to a genetic pool a directory is created.

```
tree step1
step1
├── P2
│   ├── P2_ratio.tab.gz
│   └── tmp_1_P2_stats.tab
├── T01
│   ├── T01_ratio.tab.gz
│   └── tmp_1_T01_stats.tab
├── T02
│   ├── T02_ratio.tab.gz
│   └── tmp_1_T02_stats.tab
├── T03
│   ├── T03_ratio.tab.gz
│   └── tmp_1_T03_stats.tab
├── T04
│   ├── T04_ratio.tab.gz
│   └── tmp_1_T04_stats.tab
├── T05
│   ├── T05_ratio.tab.gz
│   └── tmp_1_T05_stats.tab
├── T06
│   ├── T06_ratio.tab.gz
│   └── tmp_1_T06_stats.tab
├── T07
│   ├── T07_ratio.tab.gz
│   └── tmp_1_T07_stats.tab
├── T08
│   ├── T08_ratio.tab.gz
│   └── tmp_1_T08_stats.tab
├── T10
```

(continues on next page)

(continued from previous page)

```
    ├── T10_ratio.tab.gz
    └── tmp_1_T10_stats.tab
└── T11
    ├── T11_ratio.tab.gz
    └── tmp_1_T11_stats.tab
```

3.3.2 Determination of expected read ratio for each ancestral position based on ancestral accessions merged together

```
bin/allele_ratio_group.py -g Baurens_et_al_2019_origin.txt -p _ratio.tab.gz -o step2 -i␣
↪step1
```

In this second step the program take the input of specific allele identified in each accessions used to define genetic pools (ratio.tab.gz files of *step1* folder) and calculate an average expected allele ratio (globally a proxy of the fixation level of the allele) in the genetic pool the allele belongs.

A tabulated file is generated per genetic pool with the following format:

chromosome	position	allele	genetic pool	average allelic ratio observed	number of ancestral accessions
chr02	15033812	A	AA	0.9959677 419354839	8
chr02	17722345	G	AA	1.0	8
chr09	39501254	T	AA	1.0	8
chr05	17536961	T	AA	1.0	8
chr06	10144735	A	AA	0.9931737 588652483	8
chr08	4718673	T	AA	0.9932432 432432432	8
chr10	37498708	T	AA	0.9239074 518611573	8

3.3.3 Calculation of observed ratio in other accessions

The third step is to calculate, for each position in which an allele specific of a genetic pool was identified, the observed allelic ratio in a studied accession. In this example we calculate this ratio on the Kunnan accession.

```
bin/allele_ratio_per_acc.py -c Vcf.conf -g Baurens_et_al_2019_origin.txt -i step2 -o␣
↪step3 -a DYN163-Kunnan
```

The output can be found in the *step3* folder passed in *-o* option. This tabulated file contained 6 columns: column 1 corresponded to the chromosome, column 2 is the position of the allele, column 3 is the allele, column 4 corresponded to the observed allele frequency in the accession, column 5 is the expected allele frequency calculated at step 2 and column 6 is the genetic group to which the allele has been attributed.

For example : zmore step3/DYN163-Kunnan_ratio.tab.gz

chr	pos	allele	obs_ratio	exp_ratio	grp
chr01	20888	A	0.0	0.23513227513227516	BB
chr01	20916	C	0.14754098360655737	0.28604868303910713	BB
chr01	21019	G	0.21875	0.3700537473602161	BB
chr01	67413	T	0.5818181818181818	1.0	AA
chr01	67413	A	0.41818181818181815	1.0	BB
chr01	67461	G	0.0	0.975	AA
chr01	89923	G	0.6842105263157895	1.0	AA
chr01	89923	T	0.3157894736842105	1.0	BB
chr01	89958	T	0.6842105263157895	1.0	AA

3.3.4 Calculation on sliding of the normalized observed ratio and ancestral blocs

In this step, in a given sliding windows, the observed average allelic ratio is calculated for each genetic pool and normalized by the expected allelic ratio. The resulting value is used to infer the number of haplotypes from the studied genetic pool present in the studied accession.

Output are of two types:

- **<accession>_win_ratio.tab.gz** file containing normalized values for each genetic pools in the given windows. This file contained 4 + X columns, X being the number of genetic pools tested. The column 1 contained the chromosome name, column 2 contained the position of the central allele in the windows, column 3 contained the start position of the windows and column 4 contained the end position of the windows. Columns 5 to end contained the normalized ratio calculated for the accessions. A columns per genetic pool.
- **<accession>_<chromosome>_<haplotype>.tab** contained the hypothesized haplotypes from this accession given results from *tab.gz* file. Haplotype are hypothetic ones that tries to minimize recombinations events between distinct genetic pools. These files are formatted as follows: column 1 contained accession name, column 2 contained chromosome ID, column 3, 4 and 5 contained start, end, and origin of a region.

```
mkdir step4
bin/PaintArp.py -a DYN163-Kunnan -r step3/DYN163-Kunnan_ratio.tab.gz -c Baurens_et_al_
↪2019_color.txt -o step4/DYN163-Kunnan -w 12 -O 0 -s Baurens_et_al_2019_chromosome.txt
```

3.3.5 File formatting for GeMo visualization

This steps aims at reformatting the files so that they are compatible with GeMo tool. GeMo tool performs two tasks, the first one consists in drawing ancestral block identified at step 4. The second one also draw these blocks but allowed refinement of these block using custom and adjustable parameters. For block drawing of step 4 we will reformat block files so that they match expectation with GeMo. For this run the following command line:

```
mkdir step5
bin/convertForIdeo.py --name DYN163-Kunnan --dir step4 --col Baurens_et_al_2019_color.
↪txt --size Baurens_et_al_2019_chromosome.txt --prefix step5/DYN163-Kunnan --plo 2
```

This command generate several files with the following names:

- **<accession_id>_ideo.txt** that contained block that could be drawn with GeMo (data section),
- **<accession_id>_curve.txt** that contained block that could be drawn with GeMo (data section),
- **<accession_id>_ideoProb.txt** that contained block that could be drawn with GeMo (data section),
- **<accession_id>_chrom.txt** that contained information required to draw chromosomes.

- **<accession_id>_color.txt** contained color information that could be used to draw blocks with custom color.

3.4 References

- Baurens,F.-C. et al.(2019) Recombination and Large Structural Variations Shape Interspecific Edible Bananas Genomes. Mol Biol Evol, 36, 97–111.
- Martin et al., 2020a. Martin G, Cardi C, Sarah G, Ricci S, Jenny C, Fondi E, Perrier X, Glaszmann J-C, D'Hont A, Yahiaoui N. 2020. Genome ancestry mosaics reveal multiple and cryptic contributors to cultivated banana. Plant J. 102:1008–1025.

CHAPTER

FOUR

CHROMOSOME PAINTING USING TRACEANCESTOR

TraceAncestor is a suite of script that allows to estimate the allelic dosage of ancestral alleles in hybrid individuals and then to perform chromosome painting.

4.1 Installation

```
git clone https://github.com/gdroc/GeMo_tutorials.git
cd GeMo_tutorials
```

Download dataset, you only need to launch the script download_dataset.pl without any parameter

```
perl download_dataset.pl
```

This script create a new directory data

```
data/
├── Ahmed_et_al_2019_color.txt
├── Ahmed_et_al_2019_individuals.txt
├── Ahmed_et_al_2019_origin.txt
├── Ahmed_et_al_2019.vcf
```

4.2 Workflow

4.2.1 vcf2gst.pl

Usage

This script is used to define GST values from individuals that are identified as pure breed for an ancestor.

Must be used on pure breed. If there is introgressed part on the genome of the individual, the part must be removed before analysis.

```
bin/vcf2gst.pl --help

Parameters :
   --vcf       vcf containing the ancestors and other individuals to scan [Required]
   --ancestor  A two column file with individuals in the first column and group tag (i.e.
↪ origin) in the second column [Required]
```

(continues on next page)

(continued from previous page)

```
  --depth     minimal depth for a snp to be used in the analysis (Default 5)
  --output    output file name (Default GSTmatrix.txt)
  --help
```

Input

–ancestor Ancestor file (Required)

A two column file with individuals in the first column and group tag (i.e. origin) in the second column

individuals	origin
De_Chios	Mandarin
Shekwasha	Mandarin
Sunki	Mandarin
Cleopatra	Mandarin
Pink	Pummello
Timor	Pummello
Tahitian	Pummello
Deep_red	Pummello
Corsican	Citron
Buddha_Hand	Citron

–vcf VCF file (Required)

Now, you can run the following command

```
perl bin/vcf2gst.pl --ancestor data/Ahmed_et_al_2019_origin.txt --vcf data/Ahmed_et_al_
↪2019.vcf --output GSTMatrix.txt
```

Output

The output is a CSV file containing GST (inter-population differentiation parameter) information:

with :

- #CHROM = chromosome name
- POS = position of DSNP
- REF = Base of the reference allele of this DSNP
- ALT = Base of the alternative allele of this DSNP
- %Nref = Percentage of maximal missing data for this DSNP
- GST = value of GST (inter-population differentiation parameter) (With 1,2,3 the ancestors names)
- F = Alternative allele frequency for each ancestor (With 1,2,3 the ancestors names)

4.2.2 prefilter.pl

Usage

This script is used to define a matrix of ancestry informative markers from the matrix gotten at the step 1.

```
bin/prefilter.pl --help
Parameters :
    --matrix    GST matrix [Required]
    --gst       threshold for gst (Default : 0.9)
    --missing   threshold for missing data (Default 0.3)
    --output    output file name (Default Diagnosis_matrix)
    --help      display this help
```

Now, you can run the following command

```
perl bin/prefilter.pl --matrix GSTMatrix.txt --output Diagnosis_matrix.txt
```

Output

A matrix containing all the ancestry informative markers for every ancestors.

with:

- ancestor = Ancestor names
- chromosome = Chromosome numbers
- position = Position of the SNP marker
- allele = Base of the ancestral allele

4.2.3 TraceAncestor.pl

Usage

```
bin/TraceAncestor.pl --help

Parameters :
    --matrix     Diagnosis matrix [Required]
    --vcf       vcf of the hybrid population
    --individuals    A two column file with individuals to scan for origin (same as␣
↪defined in the VCF headerline) in the first column and the ploidy in the second column␣
↪[Required]
    --window    number of markers by window (Default 10)
    --lod       LOD value to conclude for one hypothesis (Default 3)
    --freq      theoretical frequency used to calcul the LOD (Default 0.99)
    --cut       number of K bases in one window (Default 100)
    --dirout    Directory output (Default result)
    --help      display this help
```

Input

–individuals A two column file with individuals to scan for origin (same as defined in the VCF headerline) in the first column and the ploidy in the second column.

Now, you can run the following command

```
perl bin/TraceAncestor.pl --matrix Diagnosis_matrix.txt --vcf data/Ahmed_et_al_2019.vcf -
↪-individuals data/Ahmed_et_al_2019_individuals.txt
```

Output

For each individual present in the file data/Ahmed_et_al_2019_individuals.txt, 4 outputs are produced, prefixed with the name of indivual :

- Bergamot_ideo.txt : A text file of the position of genomic blocks the ancestry mosaic with a succession of genomic blocks along the chromosome

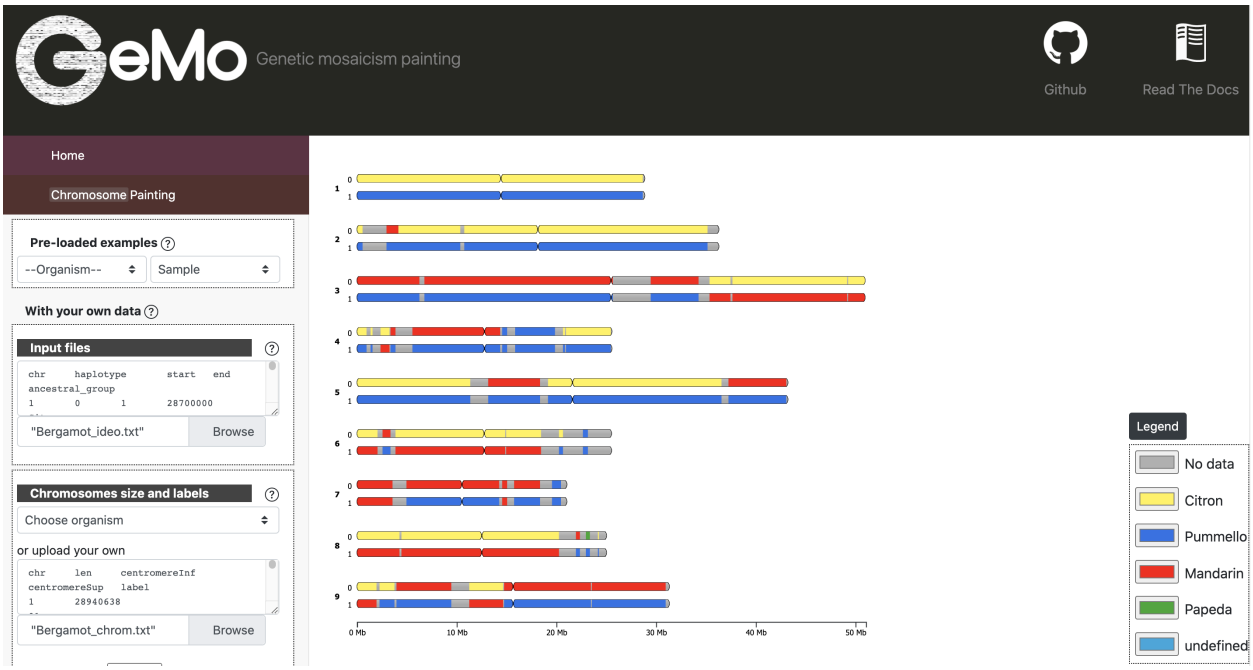
chr	haplotype	start	end	ancestral_group
1	0	1	28700000	Citron
1	1	1	28700000	Pummello
2	0	1	600000	Citron
2	0	3000001	4200000	Mandarin
2	0	4200001	10400000	Citron
2	0	10800001	35200000	Citron

- Bergamot_chrom.txt : A tab file with name, length and karyotype based on ploidy.
- Bergamot_ancestor.txt : Frequency of ancestors alleles along chromosome for the particular hybrid focused.
- Bergamot_curve.txt : Frequency of ancestors alleles along chromosome for the GeMo visualization tool.

4.3 Visualization and block refinement with GeMo

Go to GeMo WebApp

- Load data has follow



4.4 References

- Ahmed,D. et al. (2019) Genotyping by sequencing can reveal the complex mosaic genomes in gene pools resulting from reticulate evolution: a case study in diploid and polyploid citrus. Annals of Botany, 123, 1231–1251.

CHAPTER

FIVE

PCA ANALYSIS

5.1 Installation

```
pip install Bio
pip install sklearn
```

5.2 Dependencies

plink

R Package ade4

```
R
install.packages("ade4")
```

5.3 Datasets

- Download Rice 3K RG 404k CoreSNP Dataset, all chromosomes

```
cd data
wget https://s3.amazonaws.com/3kricegenome/snpseek-dl/3krg-base-filt-core-v0.7/core_v0.7.
↪bed.gz
wget https://s3.amazonaws.com/3kricegenome/snpseek-dl/3krg-base-filt-core-v0.7/core_v0.7.
↪bim.gz
wget https://s3.amazonaws.com/3kricegenome/snpseek-dl/3krg-base-filt-core-v0.7/core_v0.7.
↪fam.gz

gunzip core_v0.7.bed.gz
gunzip core_v0.7.bim.gz
gunzip core_v0.7.fam.gz
```

- Download information for a subset of these accession

```
wget https://raw.githubusercontent.com/SantosJGND/Galaxy_KDE_classifier/v1.2/Downstream_
↪functions/Analyses_Jsubtrop_self_KDE/Order_core.txt
grep -v "COUNTRY" Order_core.txt | cut -f 2 > sample.txt
```

5.4 Workflow

- Convert to vcf using plink

```
plink --bfile core_v0.7 --recode vcf-iid --keep-fam sample.txt --out core_v0.7
```

- Adjust some missing value on vcf file

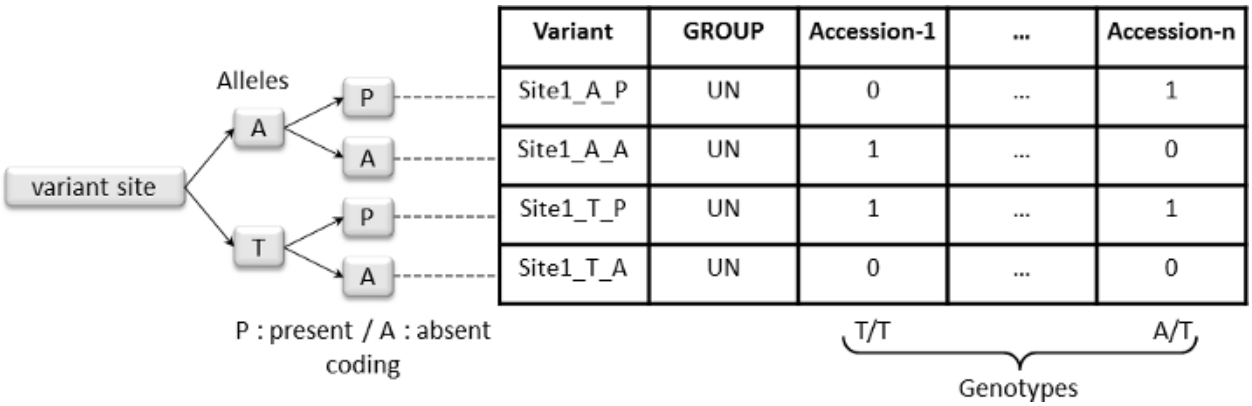
```
sed -i 's=GT=GT:AD:DP=' core_v0.7.vcf
sed -i 's=0/0=0/0:20,0:20=g' core_v0.7.vcf
sed -i 's=0/1=0/1:10,10:20=g' core_v0.7.vcf
sed -i 's=1/1=1/1:0,20:20=g' core_v0.7.vcf
sed -i 's=\.\/\.=\.\/\.:\.,\.:\.=g' core_v0.7.vcf
```

The first step of the Chromosome painting is to perform a PCA analysis on the vcf file to cluster the alleles and the accession.

Create a folder in which the analysis will be performed and run the following command line:

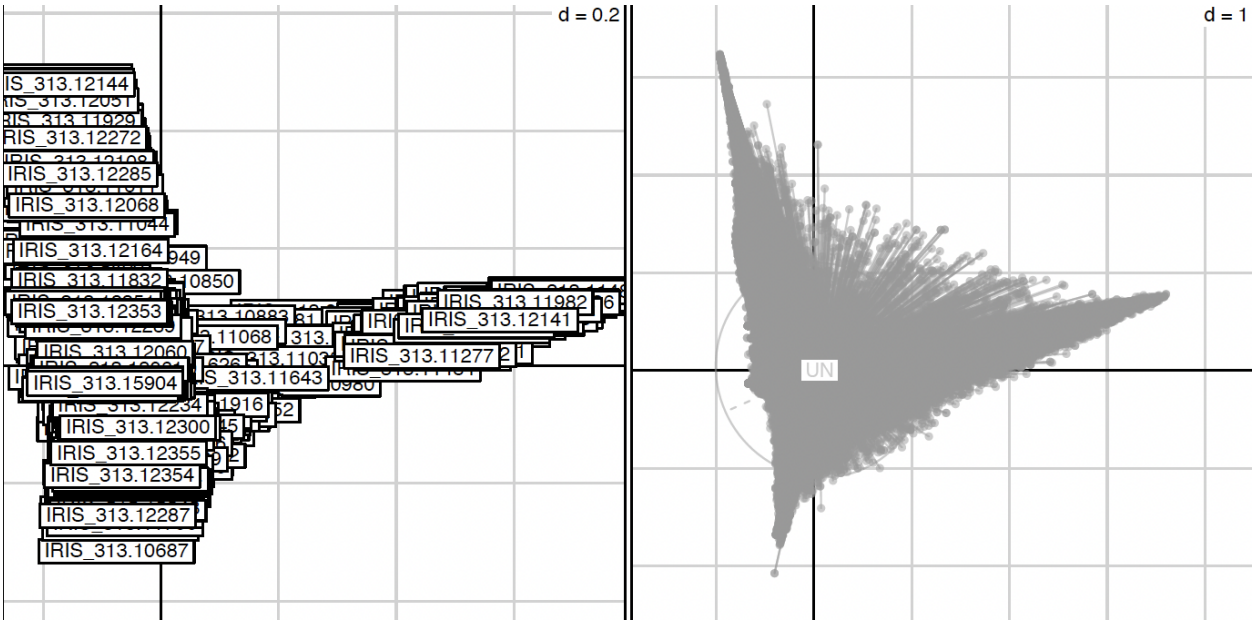
```
mkdir PCA
bin/vcf2struct.1.0.py --vcf data/core_v0.7.vcf --names data/sample.txt --type FACTORIAL -
↪-prefix PCA/Analysis --nAxes 6 --mulType coa
```

The last command line run the factorial analysis (–type FACTORIAL option). During this analysis the vcf file is recoded as followed : For each allele at each variants site two markers were generated; One marker for the presence of the allele (0/1 coded) and one for the absence of the allele (0/1 coded).



Only alleles present or absent in **part** (not all) of selected accessions were included in the final matrix file named **PCA/Analysis_matrix_4_PCA.tab** in this example. An additional column named "GROUP" can be identified. This column is filled with "UN" value if no –group argument is passed. We will explain later this argument.

The factorial analysis (here a COA, –mulType option) was performed on the transposed matrix using R (The R script is generated by the script and can be found here: **PCA/Analysis_multivariate.R**). R warning messages and command lines are recorded in the file named **Analysis_multivariate.Rout**. Graphical outputs of the analysis were draw and for example accessions and alleles can be projected along axis in the following picture.



Correspond to the file : **PCA/Analysis_axis_1_vs_2.pdf**

In this example the left graph represent accessions projected along axis 1 and 2 and the right represent the allele projected along synthetic axis. A graphical representation is performed for each axis combinations and each file is named according to the following nomenclature ***prefix + _axis_X_vs_Y.pdf***. Several pdf for accessions along axis only is also generated and are named according to the following nomenclature ***prefix + _axis_X_vs_Y_accessions.pdf***.

Individual and variables coordinates for the selected 6 first axis (–nAxes option) are recorded in files named **PCA/Analysis_individuals_coordinates.tab** and **PCA/Analysis_variables_coordinates.tab** respectively. A third file named **PCA/Analysis_variables_coordinates_scaled.tab** containing allele scaled coordinates (columns centered and reduced) along synthetic axis is generated.

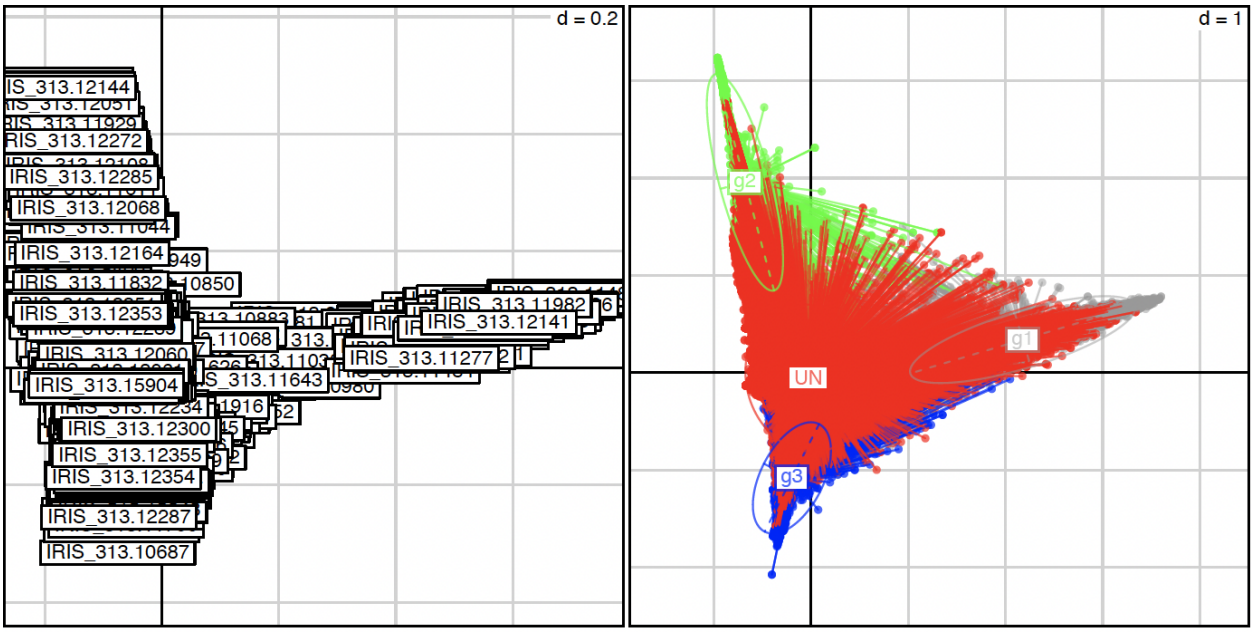
```
sort -k 2n,2 PCA/Analysis_individuals_coordinates.tab | cut -f 1 -d " " | tail -10 | sed
↪'s:\"::g' | sed  's=\.=-=' | sed "s:$:\tg1:" > group1.txt
sort -k 3n,3 PCA/Analysis_individuals_coordinates.tab | cut -f 1 -d " " | tail -10 | sed
↪'s:\"::g' | sed  's=\.=-=' | sed "s:$:\tg2:" > group2.txt
sort -k 3nr,3 PCA/Analysis_individuals_coordinates.tab | cut -f 1 -d " " | tail -10 |␣
↪sed 's:\"::g' | sed  's=\.=-=' | sed "s:$:\tg3:" > group3.txt

echo '["group"]' > data/origin.txt
cat group1.txt group2.txt group3.txt >> data/origin.txt
echo '["color"]' >> data/origin.txt
echo -e "g1\tred=0:green=1:blue=0:alpha=0.7" >> data/origin.txt
echo -e "g2\tred=0:green=0:blue=1:alpha=0.7" >> data/origin.txt
echo -e "g3\tred=1:green=0:blue=0:alpha=0.7" >> data/origin.txt
```

The –group option

We assume that in some case you have additional informations on your dataset such as which accessions are admixed and which accessions are likely to be the ancestral one. And maybe you want to verify/project this information in your analysis. This can be done passing a configuration file with two section to the –group option. This file can be found in the data/config/ folder and is named AncestryInfo.tab. You can have a look at the file if you want but basically the two sections are named [group] and [color] and contained respectively the accession suspected grouping and a color (in RGB proportion) you want to attribute to each group. Accessions with no group should filled with “UN” value.

> **Warning:** Group name should be written in upper case (due to R sorting).

```
mkdir -p PCA_group
bin/vcf2struct.1.0.py --vcf data/core_v0.7.vcf --names data/sample.txt --type FACTORIAL -
↪-prefix PCA_group/Analysis --nAxes 6 --mulType coa --group data/origin.txt
```



Mean Shift clustering Now that allele have been projected along synthetic axes, it is time to cluster these alleles. The idea is that the structure reflected by the synthetic axis represent the ancestral structure. In this context, the alleles at the extremities of the cloud of points will be the ancestral ones. These alleles can be clustered using several approaches. In this tutorial we will use a Mean Shift clustering approach.

```
bin/vcf2struct.1.0.py --type SNP_CLUST-MeanShift --VarCoord PCA_group/Analysis_variables_
↪coordinates.tab --dAxes 1:2 --mat PCA_group/Analysis_matrix_4_PCA.tab --thread 8 --
↪prefix PCA_group/Analysis --quantile 0.15
```

The Mean Shift clustering is performed with only the 2 first axes of the COA (–dAxes 1:2) because the analysis showed that most of the inertia is on these axes. With a mean shift approach, the number of group is automatically detected.

During the process, several informations are returned to standard output, but at the end of the process three main informations are returned:

- the number of alleles used for the analysis. Allele present or absent in all accessions are removed.
- the number of estimated clusters which can be found in the line:

```
Performing MeanShift
Bandwidth estimation: 0.5199882678747445
number of estimated clusters : 4
```

- the number of allele grouped within each group is returned and should look like as followed:

```
Group g0 contained 28363 dots
Group g1 contained 8704 dots
```

(continues on next page)

(continued from previous page)

```
Group g2 contained 3444 dots
Group g3 contained 3300 dots
```

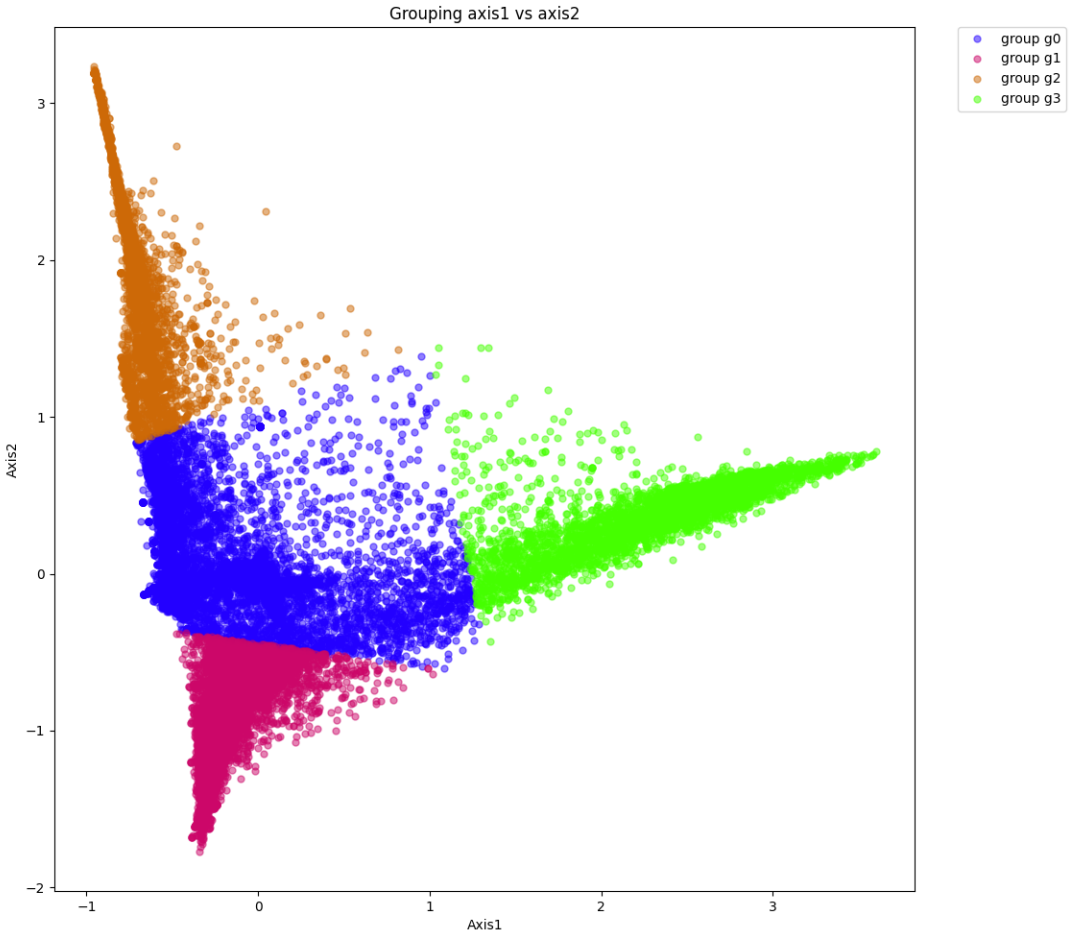
Five file are generated and can be found in the PCA_group folder:

- **PCA_group/Analysis_kMean_allele.tab** file which correspond to the PCA_group/Analysis_matrix_4_PCA.tab in which the allele grouping has been recorded.
- **PCA_group/Analysis_centroid_coordinates.tab** file which regroup the centroids coordinates.
- **PCA_group/Analysis_centroid_iteration_grouping.tab** file which records for each centroid its grouping.
- **PCA_group/Analysis_group_color.tab** file that attribute a color to the groups.
- **PCA_group/Analysis_kMean_gp_prop.tab** file that report for each allele the probability to be in each groups. This is not a "real" probability, the idea was to have a statistics in case you want to filter alleles. This value was calculated as the inverse of the euclidian distance of one point and each centroids and these values were normalized so that the sum is equal to 1.

Visualization of the allele grouping can be done as followed:

```
./bin/vcf2struct.1.0.py --type VISUALIZE_VAR_2D --VarCoord PCA_group/Analysis_variables_
↪coordinates.tab --dAxes 1:2 --mat PCA_group/Analysis_kMean_allele.tab --group PCA_
↪group/Analysis_group_color.tab --prefix PCA_group/AlleleGrouping
```

And corresponding representation :

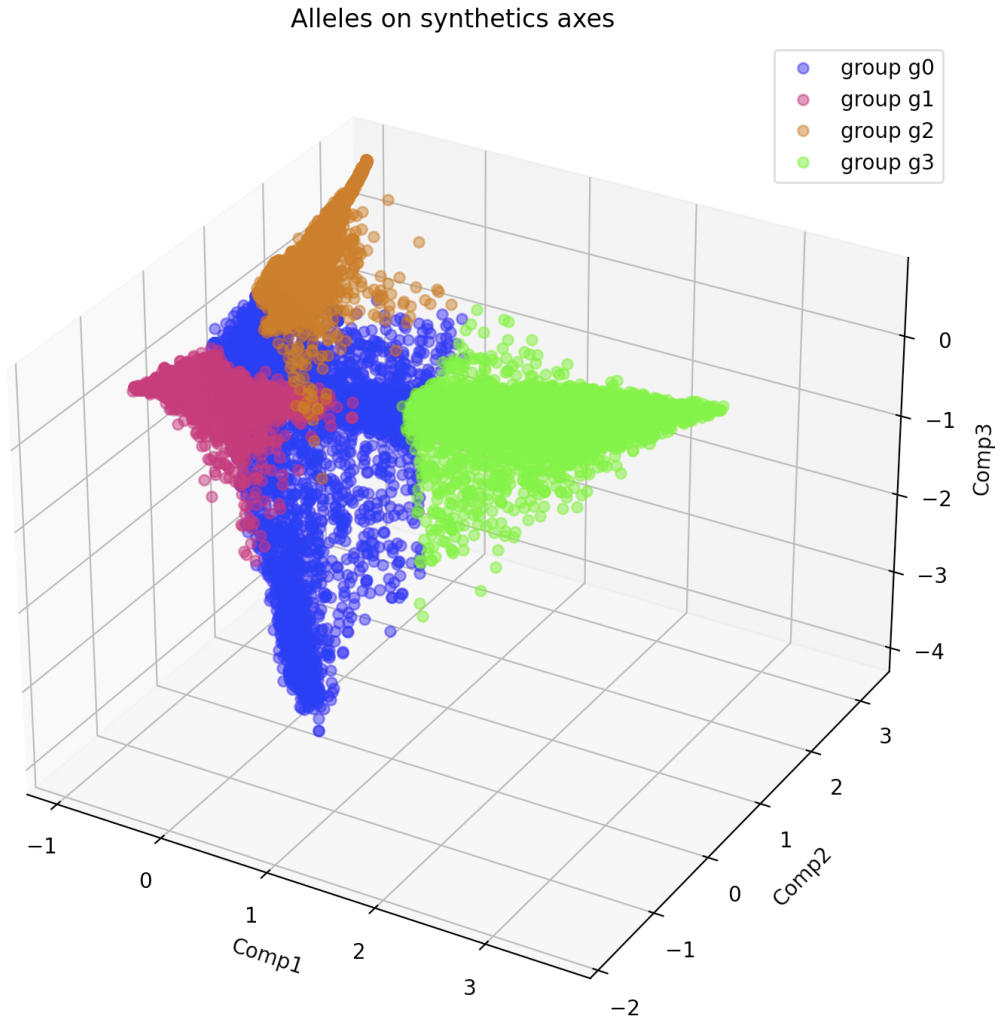


PCA_group/AlleleGrouping_axis1_vs_axis2.png

It is not necessary to have a 3d visualization but we can try the command anyway:

```
./bin/vcf2struct.1.0.py --type VISUALIZE_VAR_3D --VarCoord PCA_group/Analysis_variables_
↪coordinates.tab --dAxes 1:2:3 --mat PCA_group/Analysis_kMean_allele.tab --group PCA_
↪group/Analysis_group_color.tab
```

A window which should look like this should open:



This 3d visualization can be rotated with the mouse.

CHAPTER

SIX

LOCAL INSTALL

6.1 Prerequisites

To install GeMo on your computer you need a local server environment like MAMP.

You will also need to install Python 3 and Node. We recommand to install NVM to manage Node and NPM versions.

6.2 Clone the GeMo repository

```
git clone https://github.com/SouthGreenPlatform/GeMo.git
cd GeMo
```

6.3 Install Node dependencies

```
npm install
npm ci
```

6.4 Create required directories

```
mkdir tmp
mkdir tmp/gemo_run
mkdir tmp/gemo_saved
```

6.5 Launch node server

In GeMo directory :

```
npm run server
```

6.6 Configure socket variable

In the GeMo directory, modify the index.php file to connect to your local node server :

```
var socket = io('http://localhost:9070');
```

6.7 Configure MAMP

Start MAMP and click the "Start" button in the toolbar. In `MAMP > Preferences... > Web Server` the Document root is set to `/Applications/MAMP/htdocs`. You can change the path to point on the GeMo directorie.

Your local GeMo is now accessible in your web browser : http://localhost:8888/