User manual: Group 5

Ronan Constantine (1372967) Mthokozisi Sibanda (1407174)
Siddarth Singh (1156261) Harvey Muyangayanga (941446)

14th May 2018

1 Introduction

This manual describes the Genesis2 program created to generate PCA and admixture graphs from common tools such as eigenstrat.

The program was developed with the primary goal of providing a user friendly graphing tool and as an update of the older Genesis program developed by Wits University. Graphs can be edited to better represent data in an intuitive manner.

Git repository

The Git repository for our project is `https://github.com/FruitSenpai/Genesis`

2 User manual

2.1 Installing Genesis 2

Genesis2 does not require an installation or setup procedure and can be directly run from the root directory.

It does require the prior installation of Python 3.6.5rc1. To check for a python installation enter commandline and enter ”python” into the terminal. If it is present a message with ”Python” and a build number will appear.

Libraries that will need to be installed are matplotlib and wxpython. An installer will be provided if needed for Linux. If on Windows then please refer to the README.TXT contained within the Genesis

Additional help can be found on:

- https://wxpython.org/pages/downloads/
- https://matplotlib.org/users/installing.html

Genesis is compatible with both macOSX, Linux distros and Windows 10.Please note that only the Linux and windows versions have been tested.

2.2 Running Genesis 2.o

Windows and Linux If Python distro 3.6.x or higher is installed the user should be able to run the program by double clicking the file otherwise the program can be run using the command ”python Genesis2.py” in either terminal or command line. The File to run the program from is called Genesis2.py and is found in the scripts sub-folder of the main Genesis folder.

2.3 Menu Structure

When Genesis2 runs , a menu appears at the top. The items are File, Graph ad Help. File contains more options which are used to create and load new graphs or exit the program.

Added icons as well as frequently used menu items appear at the top of the Genesis 2 window. The items are briefly explained in the below diagram.

Figure 1: Genesis2.o mainpage

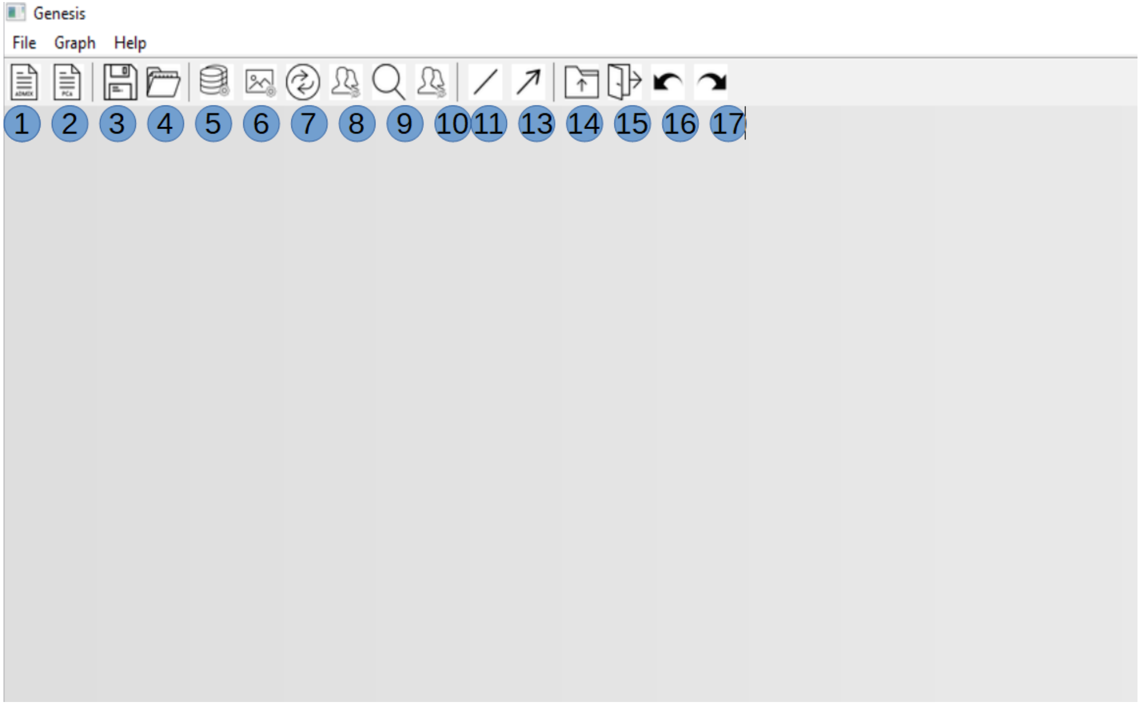


Table 1: Table of GUI components

Item Number	Item Name	Item Details
1	Admix	Opens window to create admix graph
2	PCA	Opens window to create PCA graph
3	Save	Opens window to save project
4	Load	Open window to load new prior projects
5	Data	Select new options to detail data
6	Appearence	Change appearence of graph onscreen
7	Refresh	Refresh graph to reflect changes
8	Select Individual	Search for specific individual in dataset
9	Search for group	search for specific group of individuals in data set
10	Search Hidden Individual	Search for specific hidden individual in data set
11	Line	Draw line on graph
12	Arrow	Draw arrow into graph
13	Export	Export current graph
14	Close Project	Exit current project
15	Undo	Undo last action done to graph
16	Redo	Redo last action that was undone on graph

2.4 Unimplemented components

To finish the intial version of the Genesis2.o project in time certain features were ommitted. The buttons or menu items to access these features are still present but will only create a dialogue prompt rather than opening a usable page. These features are listed below:

- PCA/Admix data options accessed by the Apprearence button under the Graph heading.
- Font options for graphs accessable by the appearence options.
- 3D graph generation
- undo and Redo
- Search for individual
- Search for hidden individual
- Show/Hide indvidual

2.5 Graph menu components

When a graph has been plotted a small toolbar will be placed at the bottom of the graph. The functions of the buttons will be explained below.

Table 2: Graph Components

Item Name	Item Details
Home	returns the graph view to its original state
Back	Goes back to the previous view if any
Forward	Goes to the next view if any
Pan	Allows the user to pan around the graph view
Zoom	Allows user to zoom in the graph view
Subplots	Allows user to change the scaling of the graph
Export	Allows for the graph to be exported as an image

2.6 Creating a graph

Use the following steps to create a graph on Genesis2

- Click the relevant graph creation button.
- Name the graph.
- Import the relevant files.
  - A PCA graph must have a pca file as the data file. It is heavily encouraged that the file be of type .pca.evec .
  - An Admix graph must have an admixture file and a fam file. It is heavily encouraged that the fam file be of type .fam and and the admixture file be of type .Q.n where n is a integer.
  - Phenotype files are optional. it is heavily encouraged that a phenotype file be of type .phe
- Choose applicable columns .
- Click ”Accept”.

2.7 Customization

The customization menu can be found by clicking the Appearance icon. This will detect what type of graph is currently shown and bring up the relevant customization options.

2.7.1 Pca customization

The following elements can be customized using the interface.

- Selected group.
- Group icon.
- Icon colour.
- Icon size .

2.7.2 Admix customization

The following elements can be customized using the interface.

- Select whether to customize the groups or ancestries.
- Select ancestry.
- Define colours of ancestries.
- Define order of ancestries.
- Sort ancestries by dominance.
- Place most dominant ancestry first.
- Select group.
- Define order of the groups
- Sort groups by dominance
- Place most dominant group first.
- Hide a certain group first

2.8 Annotation

To annotate follow the following steps

- Click on either the arrow or line button in the main menu
- Click and drag between two points to create an annotation
- Click the same button again to turn off annotation

Note: Annotations will not be saved when saving a graph

2.9 Saving

When the save button is clicked, the currently displayed graph will be saved. The graph can be saved anywhere on the computer's accessible storage. The pca and admixture graphs have different file extensions when being saved. Use a unique name when saving your graphs so as to not overwrite previously saved ones. All customizations done to the graph will be saved as well.

2.10 Loading

To load a graph click the load button. Navigate to whichever directory the project file has been saved.The dialogue box will originally be looking for an admixture project file but that can be changed to look for a pca file instead. Click the desired file to load it.

2.11 Exporting

An image of the graph can be exported by clicking the export button in the graph tool bar, located at the bottom of the graph. The image can be saved as a pdf or png.

2.12 Known errors

- If two graphs have the same name, the original one will lose all functionality
- The wrong file type can be imported and used as data. For this Alpha build of the project it is up to the users own discretion as to which file to use.
- Arrows tend to look skew as opposed to straight lines.

3 Technical manual

3.1 MVC

The design pattern used for this project was MVC. This design pattern splits the project into three sections that handle their own specific areas of the program.

3.1.1 Model

The model classes for the program reside in the FileManagement folder. These classes are responsible for the fetching and processing of data into a form that we can more easily utilise. The folder contains 4 files aside from the blank init.py file. These are:

- File.py
- FileImporter.py
- FileManager.py
- ValidityChecker.py

Genesis2 uses 3 input file types of which 2 are mandatory. The data is processed using a set of scripts located in the File Management Folder. These scripts take in the data files and convert them into 2D arrays for use in the main program. They also contain a ”Validity Checker” script which goes through data files on access in order to determine if the file is of the correct format to ensure the validity of data. However this script has been removed for this alpha version to allow the user more freedom to enter their own files at their own discretion. This is due to not knowing all possible file types that can result in the data that is needed

3.1.2 View

The view classes contain all the code required to generate the user interface of the program. The Genesis2.py file is found in the root directory of the program and acts as a linking point for all the processing and data management functions in the program. Through this class all additional frames found in the GUIFrames folder are used in order to fetch data and link such data to be displayed back on the main screen.These classes make up the view portion of the program and are what the user can interact with to run the program. The scripts found in the GUIFrames folder are:

- AdmixData.py
- AdmixMain.py
- AppearenceFrame.py
- DataHolder.py
- ExportFrame.py
- PCAAppear.py
- PCADataFrame.py
- PCAMain.py

3.1.3 Controller

The controller classes consist of the scripts that are used to process the raw data in order to be displayed to the user.These scripts act as the gateway between the models raw data and the views displaying of the various graphs and frames. The scripts are contained in the Graph folder along with 2 other folders to separate the PCA and Admix software.Inside the Graph folder:

- GraphSaver.py
- GraphCreate.py
- admix(Folder)
- PcaGraphing(Folder)

Inside the admix folder the following files are found

- AdmixAncestry.py
- AdmixCreateGraph.py
- AdmixDataExtractor.py
- AdmixGraph.py
- AdmixGroup.py
- AdmixIndividual.py
- stack.py

Inside the PcaGraphing folder the following are found

- PcaDataExtractor.py
- PcaGraph.py
- PcaGroup.py

Finally there is also an ”Annotation.py” file within the Annotation folder. This deals with drawing lines and arrows onto the graph to show something is important.

4 Class Description

4.1 File Input

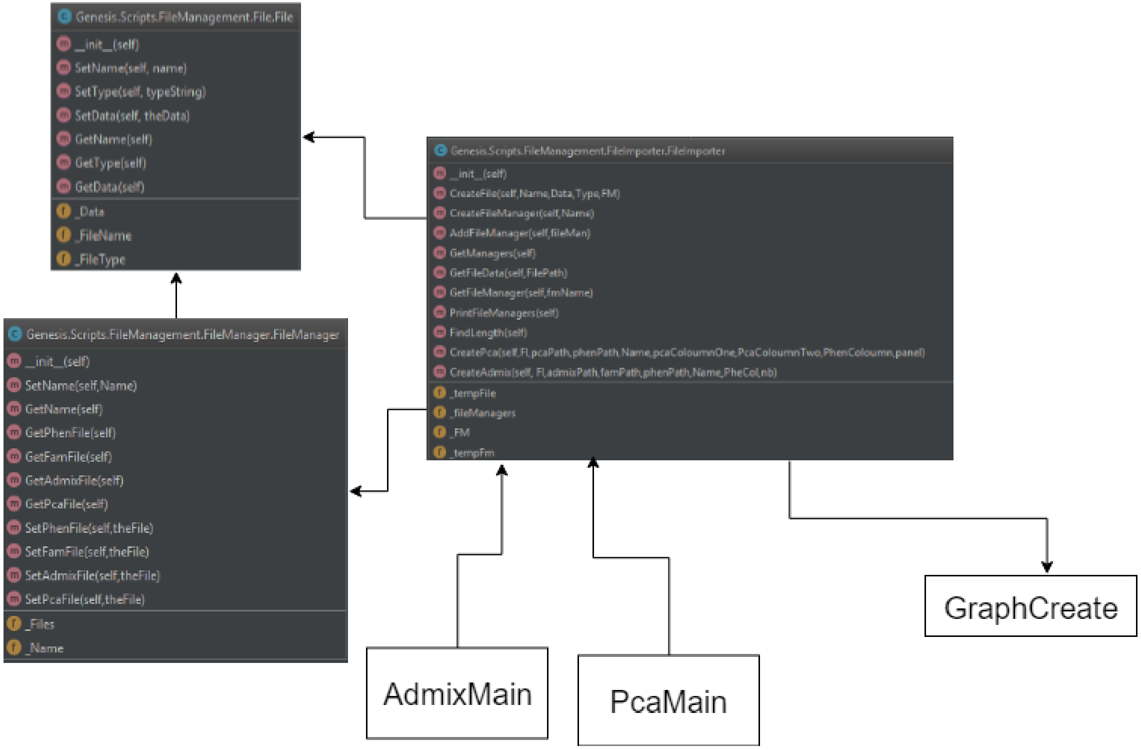
By default certain dialogues will be set to display specific file types however once the dialogue is open the use can choose to make all file types visible. An error catching system was implemented to check filetype however it was determined for the purpose of a beta build it would be beneficial to remove this functionality so that the user could test the outputs with undesirable data despite the knowledge that it may have strange side effects on the output.

4.2 File Management

There are several files within the ”File Management” folder that are used in the Model potion of the design pattern. These files are used to extract data from the imported files and create the various graphs.

- File.py is used as a general class to hold the data of one imported file, along with its type and name.
- FileManager.py is used to store all data for one Graph object. Each File manager stores multiple Files that hold the various imported data for its respective graph.
- The FileImporter is created once as its needed. It stores the FileManager for each graph and is in charge of importing the data and placing it into a 2D list so that the data can be handled correctly.

Figure 2: File manager UML diagram


Genesis.Scripts.FileManagement.File.File
__init__(self)
SetName(self, name)
SetType(self, typeString)
SetData(self, theData)
GetName(self)
GetType(self)
GetData(self)
_Data
_FileName
_FileType
Genesis.Scripts.FileManagement.FileImporter.FileImporter
__init__(self)
CreateFile(self,Name,Data,Type,FM)
CreateFileManager(self,Name)
AddFileManager(self,fileMan)
GetManagers(self)
GetFileData(self,FilePath)
GetFileManager(self,fmName)
PrintFileManagers(self)
FindLength(self)
CreatePca(self,FI,pcaPath,phenPath,Name,pcaColoumnOne,PcaColoumnTwo,PhenColoumn,panel)
CreateAdmix(self, FI,admixPath,famPath,phenPath,Name,PheCol,nb)
_tempFile
_fileManagers
_FM
_tempFm
Genesis.Scripts.FileManagement.FileManager.FileManager
__init__(self)
SetName(self,Name)
GetName(self)
GetPhenFile(self)
GetFamFile(self)
GetAdmixFile(self)
GetPcaFile(self)
SetPhenFile(self,theFile)
SetFamFile(self,theFile)
SetAdmixFile(self,theFile)
SetPcaFile(self,theFile)
_Files
_Name
GraphCreate
AdmixMain
PcaMain

4.3 Display Classes

The scripts pertaining to user interface are contained in the ”FileManagement” folder and the ”Genesis2.py” folder in the root directory. Each frame that is accessible by the user is its own individual script which is called when it needs to be shown.

Two Additional folders can be found in the ”GUIFrames” folder which allow access to the classes dealing with the customisation of the admix and PCA graphs. Inside these folders are 2 scripts, one the script to generate the customisation option frame and another to transmit these changes to the current graph being used.

The ”Genesis2” class is used as the start file for the program. It contains functions that open 2 other frames which allow for the importing of data in order to create a PCA or admix graph. The class contains a frame on which a notebook with tabs in embedded. This notebook is then used to display graphs which are directly plotted onto it as well as allowing users to draw arrows or lines on these graphs. The panel the notebook is embedded on is also programmed to react to mouseclicks The ”AdmixMain” and ”PCAMain” classes allow the user to open dialog boxes using buttons which then allow the user to select files from the directory to use in plotting the graph. Files of the wrong type will cause the generation of an error message which will remain on screen until clicked.

Figure 3: Display Class UML diagram

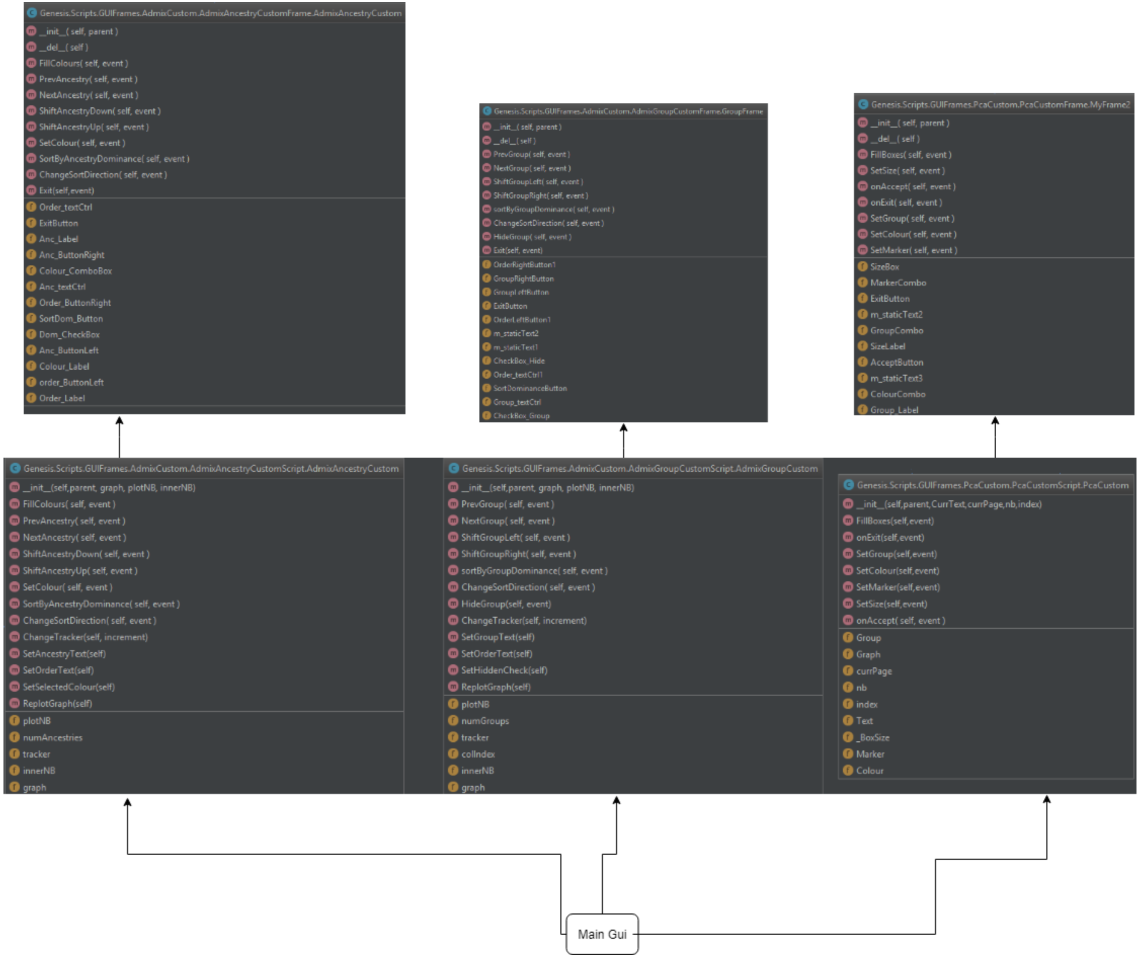
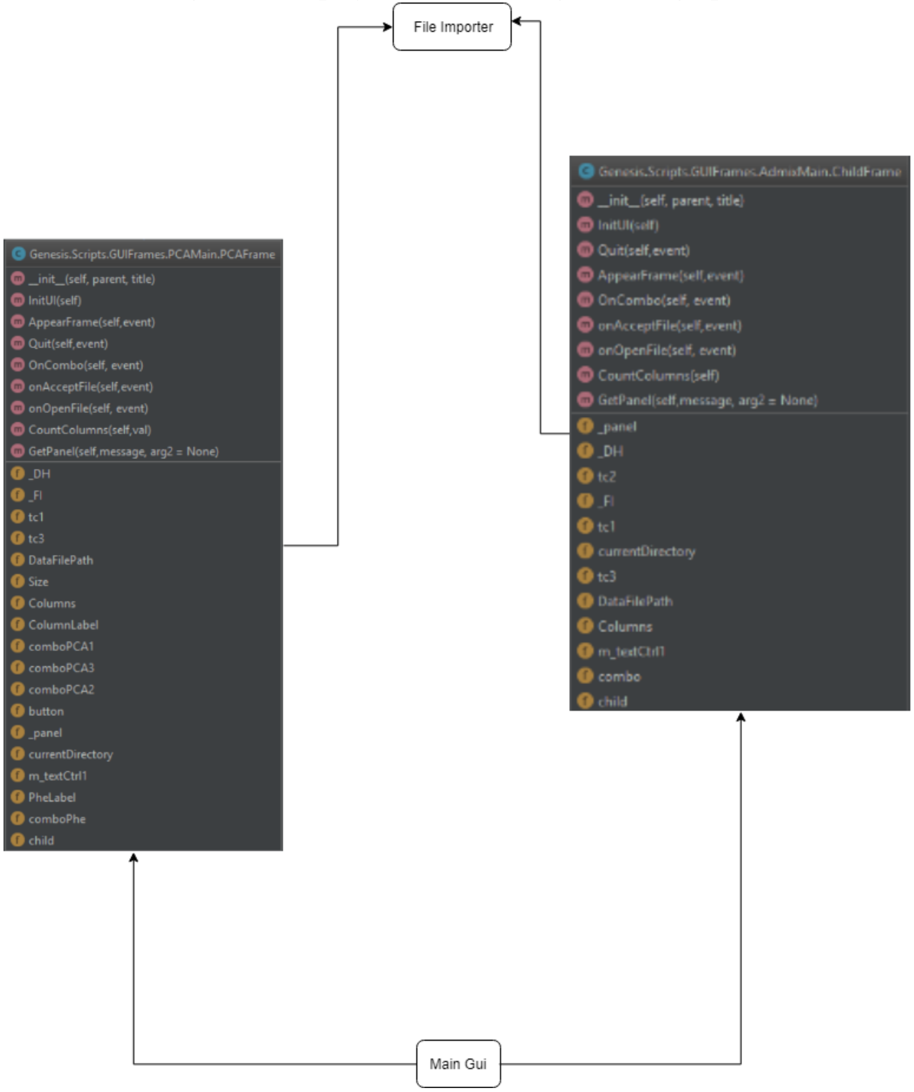


Figure 4: Display Class UML diagrams for graphs



4.4 Graphing Classes

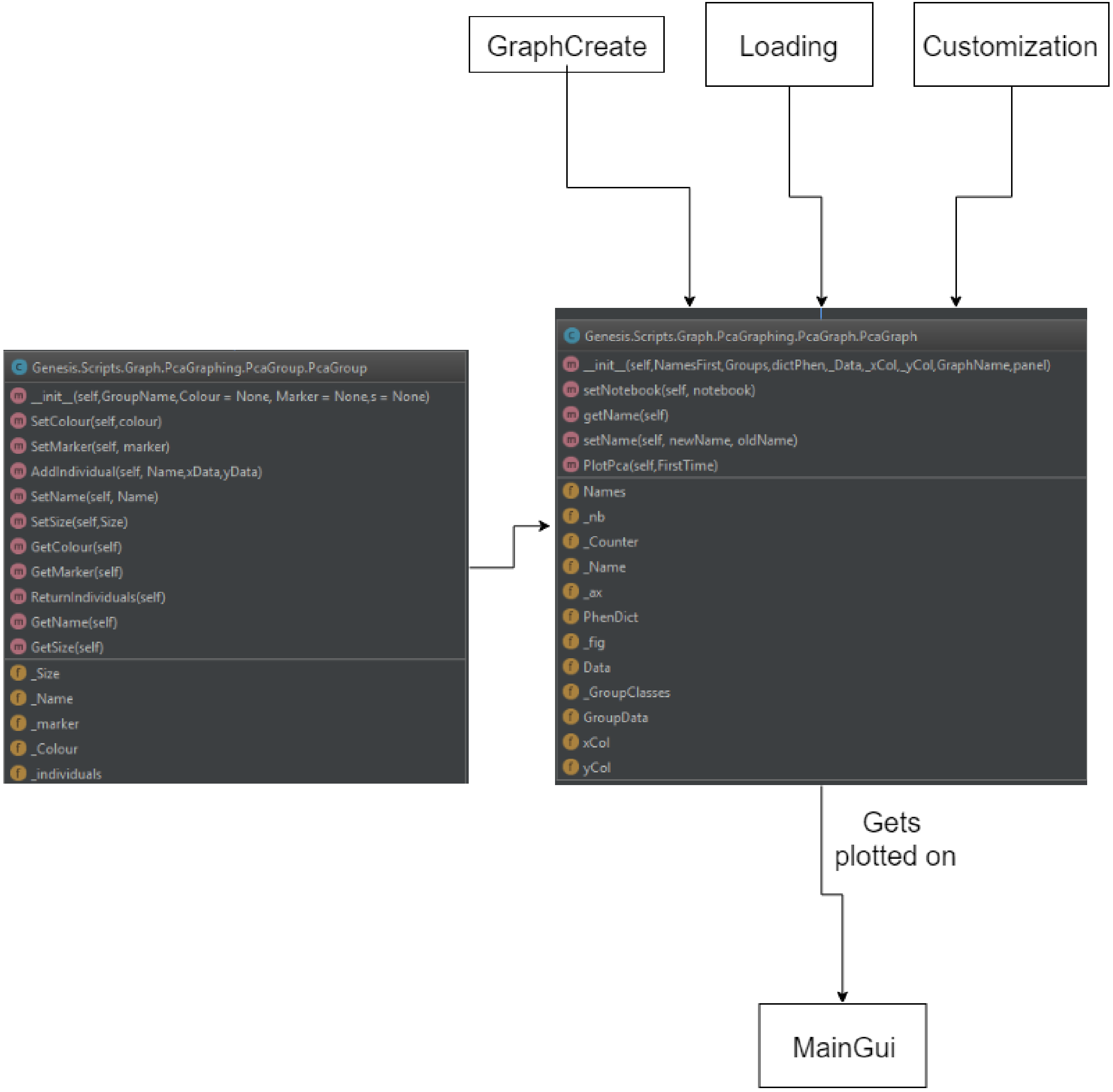
The scripts responsible for the interpreting of data from the graph files are found in the folder ”Graph”. Inside this folder are 2 scripts to create and save new graphs and2 folders containing the scripts to generate admix and PCAgraphs.

The ”PCAGraph” folder is split into 3 scripts; The PcaData Extractor, PcaGraph and PcaGroup.

- The DataExtractor returns usable data from the data files by finding the number of individuals in a group and all groups present in the phenotype file along with creating a list of individuals from the PCA file.
- PcaGraph is then used to plot the graph and calls a class ”DataHolder” in the ”GUIFrame” folder which is used to transfer the data for the plotted graph into the main GUI frame. This object holds all the data needed to recreate the graph and from it the graph can be replotted without needing the ”FileImporter”.

- PcaGroup is used to group the dataset using colours and unique markers in the other classes.

Figure 5: Display Class UML diagram for Pca Graph



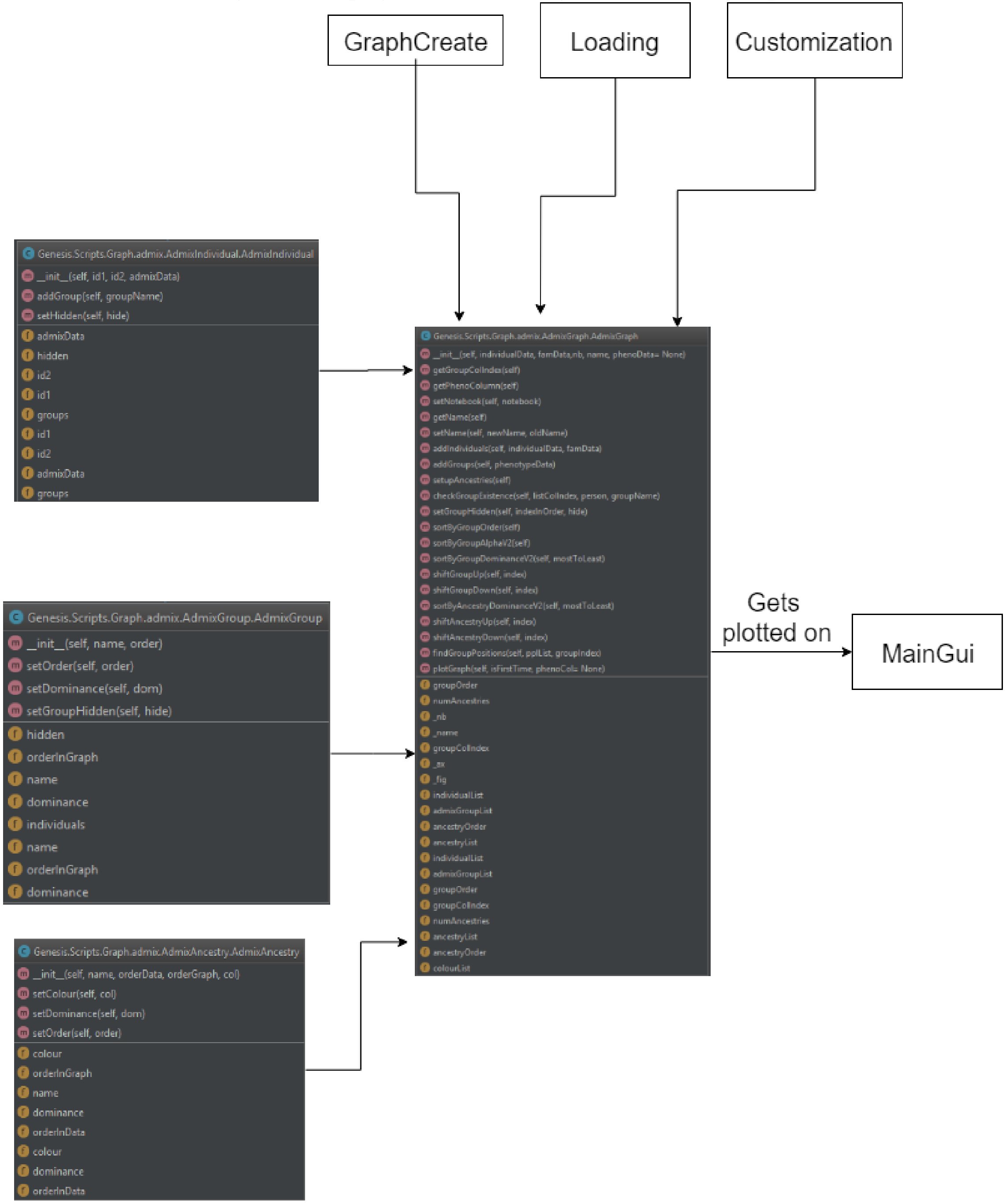
```
Genesis.Scripts.Graph.PcaGraphing.PcaGroup.PcaGroup
__init__(self,GroupName,Colour = None, Marker = None,s = None)
SetColour(self,colour)
SetMarker(self, marker)
AddIndividual(self, Name,xData,yData)
SetName(self, Name)
SetSize(self,Size)
GetColour(self)
GetMarker(self)
ReturnIndividuals(self)
GetName(self)
GetSize(self)
_Size
_Name
_marker
_Colour
_individuals
```

```
Genesis.Scripts.Graph.PcaGraphing.PcaGraph.PcaGraph
__init__(self,NamesFirst,Groups,dictPhen,_Data,_xCol,_yCol,GraphName,panel)
setNotebook(self, notebook)
getName(self)
setName(self, newName, oldName)
PlotPca(self,FirstTime)
Names
_nb
_Counter
_Name
_ax
PhenDict
_fig
Data
_GroupClasses
GroupData
xCol
yCol
```

The ”admix” folder contains 7 files: ”AdmixAncestry”,”AdmixCreateGraph”,”AdmixDataExtractor”, ”AdmixGraph”,”AdmixGroup”,”Admixindividual” and ”stack”.

- The ”AdmixGraph”,”AdmixGroup” and ”AdmixDataExtractor” perform the same functionality as their PCA equivalents
- The ”AdmixAncestry” assigns values of colour, dominance and order in data to the entries into the admix graph
- The ”AdmixIndividual” assigns a key to the input data which uniquely identifies an individual. The class also creates a list of all the individual’s ancestry points, the groups it belongs to and then adds the group name to a group list

Figure 6: Display Class UML diagram for Admix Graph



```
Genesis.Scripts.Graph.admix.AdmixIndividual.AdmixIndividual
m __init__(self, id1, id2, admixData)
m addGroup(self, groupName)
m setHidden(self, hide)
f admixData
f hidden
f id2
f id1
f groups
f id1
f id2
f admixData
f groups
```

```
Genesis.Scripts.Graph.admix.AdmixGroup.AdmixGroup
m __init__(self, name, order)
m setOrder(self, order)
m setDominance(self, dom)
m setGroupHidden(self, hide)
f hidden
f orderInGraph
f name
f dominance
f individuals
f name
f orderInGraph
f dominance
```

```
Genesis.Scripts.Graph.admix.AdmixAncestry.AdmixAncestry
m __init__(self, name, orderData, orderGraph, col)
m setColour(self, col)
m setDominance(self, dom)
m setOrder(self, order)
f colour
f orderInGraph
f name
f dominance
f orderInData
f colour
f dominance
f orderInData
```

```
Genesis.Scripts.Graph.admix.AdmixGraph.AdmixGraph
m __init__(self, individualData, famData,nb, name, phenoData= None)
m getGroupColIndex(self)
m getPhenoColumn(self)
m setNotebook(self, notebook)
m getName(self)
m setName(self, newName, oldName)
m addIndividuals(self, individualData, famData)
m addGroups(self, phenotypeData)
m setupAncestries(self)
m checkGroupExistence(self, listColIndex, person, groupName)
m setGroupHidden(self, indexInOrder, hide)
m sortByGroupOrder(self)
m sortByGroupAlphaV2(self)
m sortByGroupDominanceV2(self, mostToLeast)
m shiftGroupUp(self, index)
m shiftGroupDown(self, index)
m sortByAncestryDominanceV2(self, mostToLeast)
m shiftAncestryUp(self, index)
m shiftAncestryDown(self, index)
m findGroupPositions(self, pplList, groupIndex)
m plotGraph(self, isFirstTime, phenoCol= None)
f groupOrder
f numAncestries
f _nb
f _name
f groupColIndex
f _ax
f _fig
f individualList
f admixGroupList
f ancestryOrder
f ancestryList
f individualList
f admixGroupList
f groupOrder
f groupColIndex
f numAncestries
f ancestryList
f ancestryOrder
f colourList
```