

---

# TASSEL: Pipeline: *Guide to using Tassel Pipeline*

---

**Terry Casstevens** (*tmc46@cornell.edu*), **Zhiwu Zhang**, **Peter Bradbury**, **Dallas Kroon**,  
**and Edward Buckler**

Institute for Genomic Diversity, Cornell University, Ithaca, NY 14853-2703

May 28, 2008

Prerequisites .....	2
Source Code .....	2
Plugins.....	2
CombineDataSetsPlugin .....	2
CombineSimpleTableReportsPlugin.....	2
CreateTreePlugin.....	2
FileLoadPlugin.....	2
FilterAlignmentPlugin .....	3
FilterDataSetPlugin.....	3
FilterTaxaAlignmentPlugin .....	3
GLMPlugin .....	3
GenotypeTransformPlugin.....	4
Grid2dDisplayPlugin .....	4
IntersectionAlignmentPlugin .....	4
KinshipPlugin.....	5
LinkageDiseqDisplayPlugin .....	5
LinkageDisequilibriumPlugin.....	5
LogisticRegressionAssocPlugin.....	5
MLMPlugin.....	5
NumericalGenotypePlugin.....	6
PassThroughPlugin .....	6
SetColumnTypesPlugin .....	6
SequenceDiversityPlugin.....	6
StepwisePlugin.....	6
StepwiseAnalysisPlugin.....	6
SynonymizerPlugin.....	7
TableDisplayPlugin.....	7
TreeDisplayPlugin.....	7
UnionAlignmentPlugin .....	7
WriteXLSPlugin.....	8
chart.ChartDisplayPlugin.....	8
gdpc.ConvertGDPCToPALPlugin.....	8
gdpc.GDPCPlugin.....	8
numericaltransform.NumericalTransformPlugin.....	9
pedigree.PedigreePlugin.....	9
snpassay.ExtractSNPAssaysPlugin.....	9
Appendix A: Example Pipeline Diagrams .....	10
Appendix B: Example Source Code – MLM and GLM Pipeline .....	13
Appendix C: Example Source Code – Gramene Pipeline.....	18

## **Prerequisites**

- Java SDK 6.0 (<http://java.sun.com/javase/downloads/index.jsp>).

## **Source Code**

- <http://tassel.cvs.sourceforge.net/tassel/maizegenetics>

## **Plugins**

### **CombineDataSetsPlugin**

This combines multiple inputs into one output. Use method `receiveDataSetFrom` to specify a plugin that's output will be used as input to this plugin. Use method `receiveDataSetOnceFrom` to specify a plugin that's first output will be remembered and used in every output iteration. This plugin will wait until all specified inputs have been received, at which point, all inputs will be combined and sent as one output data set. After each iteration, all inputs received from plugins specified by method `receiveDataSetFrom` will be reset and this plugin will wait for the next set of inputs.

#### ***Inputs***

Any Object type is allowed.

#### ***Outputs***

Merged list of objects received as inputs.

### **CombineSimpleTableReportsPlugin**

Combines multiple SimpleTableReports into one.

#### ***Inputs***

One or more `net.maizegenetics.pal.report.SimpleTableReport`.

#### ***Outputs***

Resulting combined `net.maizegenetics.pal.report.SimpleTableReport`.

### **CreateTreePlugin**

Creates a tree or cladogram based on an alignment (sequence or markers). It calculates a distance matrix and then makes a tree either by Neighbor joining or UPGMA.

#### ***Inputs***

One or more `net.maizegenetics.pal.alignment.AnnotationAlignment`.

#### ***Outputs***

If neighbor joining is true, this returns `net.maizegenetics.pal.tree.NeighborJoiningTree`. Otherwise, this returns `net.maizegenetics.pal.tree.UPGMATree`. If save matrix is true, this also returns `net.maizegenetics.pal.distance.GeneralizedAlignmentDistanceMatrix`.

### **FileLoadPlugin**

This will load data from a file and package it into an PAL object.

### ***Inputs***

One or more java.io.File.

### ***Outputs***

This outputs one object at a time for each file input.

If file type: Sequence, returns net.maizegenetics.pal.alignment.AnnotationAlignment.

If file type: Polymorphism, returns net.maizegenetics.pal.alignment.Genotype.

If file type: Annotated, returns net.maizegenetics.pal.alignment.Genotype.

If file type: Numerical, returns net.maizegenetics.pal.alignment.SimplePhenotype.

If file type: SqrMatrix, returns net.maizegenetics.pal.distance.DistanceMatrix.

### **FilterAlignmentPlugin**

Generally used for taking a sequence alignment and extracting SNPs. It can also be used to extract indels, filter sites with lots of missing data, and filter by allele frequency.

### ***Inputs***

One or more net.maizegenetics.pal.alignment.AnnotationAlignment.

### ***Outputs***

Returns list of net.maizegenetics.pal.alignment.MarkerAlignment. One for each input.

### **FilterDataSetPlugin**

This filters the input data set and outputs a data set containing only the datums matching the names and types specified by the constructor.

### ***Inputs***

Data set with any list of object types.

### ***Outputs***

Data set containing only datums matching specified names and types.

### **FilterTaxaAlignmentPlugin**

This filters out all ids except those specified by method setIdsToKeep in the net.maizegenetics.pal.ids.IdGroup.

### ***Inputs***

One or more alignments (e.g. net.maizegenetics.pal.alignment.Alignment, or net.maizegenetics.pal.alignment.Phenotype) that support net.maizegenetics.pal.report.TableReport.

### ***Outputs***

For each input, returns same type of Alignment containing only ids specified by method setIdsToKeep.

### **GLMPlugin**

Run an association analysis between markers and phenotypes using a general linear model. Covariates to control for population structure can be defined.

### ***Inputs***

One or more net.maizegenetics.pal.alignment.MarkerPhenotype.

### ***Outputs***

Statistical results in a net.maizegenetics.pal.report.TableReport.

### **GenotypeTransformPlugin**

Converts a genotype alignment (each allele has a different space in a array) to a PAL alignment where genotypes are either recoded, collapsed, or split into separate alignments. This new alignments can then be analyzed by all the rest of the analysis tools.

### ***Inputs***

One or more net.maizegenetics.pal.alignment.Genotype.

### ***Outputs***

List of objects (one for each input).

If conversion method: Split - new alignments are created each gamete if phase is known. Number of output alignment = ploidy.

If conversion method: Dosage, Converts the SNPs into dosage (additive and dominant components). This is not fully supported in GLM and MLM yet (but it needs to be!). It does not work well for multiallelic markers like SSRs.

If conversion method: GenState – Converts each locus into genotypic states(e.g. allele 1=G:allele 2=C is converted to a “G:C” genotype). This is supported by LD, MLM, and GLM.

If conversion method: OneGamete, ??????

### **Grid2dDisplayPlugin**

Still in development.

### ***Inputs***

One or more net.maizegenetics.pal.report.TableReport.

### ***Outputs***

?????????

### **IntersectionAlignmentPlugin**

This returns the intersection of given inputs.

### ***Inputs***

Zero or more net.maizegenetics.pal.alignment.AnnotationAlignment and zero or more net.maizegenetics.pal.alignment.Phenotype. Must be at least two inputs any combination of mentioned types.

### ***Outputs***

If all inputs are AnnotationAlignments, then this returns net.maizegenetics.pal.alignment.MarkerAlignment. If all inputs are Phenotypes, then this returns net.maizegenetics.pal.alignment.SimplePhenotype. If the inputs are mixed, then this returns net.maizegenetics.pal.alignment.SimpleMarkerPhenotype.

### **KinshipPlugin**

Still in development. This will estimate a kinship matrix based on marker data. Currently, for the pipeline load it in from file using FileLoadPlugin-Sqrmatrix.

#### ***Inputs***

One net.maizegenetics.pal.alignment.MarkerPhenotype.

#### ***Outputs***

????????????

### **LinkageDiseqDisplayPlugin**

Displays a plot of linkage disequilibrium for all pairs of sites. It also make a little graphic of the marker locations at the bottom.

#### ***Inputs***

One net.maizegenetics.pal.popgen.LinkageDisequilibrium.

#### ***Outputs***

Result image of net.maizegenetics.pal.gui.LinkageDisequilibriumComponent is saved to file if specified by method setSaveFile().

### **LinkageDisequilibriumPlugin**

Calculates basic measures of linkage disequilibrium –  $r^2$ ,  $D'$ , and P-values.

#### ***Inputs***

One or more net.maizegenetics.pal.alignment.AnnotationAlignment.

#### ***Outputs***

Returned one at a time, one net.maizegenetics.pal.popgen.LinkageDisequilibrium object for each input.

### **LogisticRegressionAssocPlugin**

#### ***Inputs***

One or more net.maizegenetics.pal.alignment.MarkerPhenotype.

#### ***Outputs***

One at a time for each input, returns a net.maizegenetics.pal.report.SimpleTableReport of net.maizegenetics.stats.logisticReg.LocusEnvironmentPermutation and a net.maizegenetics.pal.report.SimpleTableReport of net.maizegenetics.stats.logisticReg.LogisticRatio.

### **MLMPlugin**

Run an association analysis between markers and phenotypes using a mixed linear model. Covariates to control for population structure can be defined (Q), as can co-ancestry matrix as a random effect (K).

#### ***Inputs***

One or more net.maizegenetics.pal.alignment.MarkerPhenotype and one net.maizegenetics.pal.distance.DistanceMatrix.

### *Outputs*

Statistical results in a net.maizegenetics.pal.report.TableReport.

### **NumericalGenotypePlugin**

TODO.

### *Inputs*

### *Outputs*

### **PassThroughPlugin**

TODO.

### *Inputs*

### *Outputs*

### **SetColumnTypesPlugin**

TODO.

### *Inputs*

### *Outputs*

### **SequenceDiversityPlugin**

Estimate nucleotide diversity and the allele frequency distribution.  $\pi$  and  $\theta$  are estimated. Tests of selection could be included easily (e.g. Tajima and Fu&Li).

### *Inputs*

One or more net.maizegenetics.pal.alignment.AnnotationAlignment.

### *Outputs*

One at a time for each input, returns a net.maizegenetics.pal.report.SimpleTableReport of net.maizegenetics.pal.popgen.PolymorphismDistribution and a net.maizegenetics.pal.report.SimpleTableReport of net.maizegenetics.pal.popgen.DiversityAnalyses.

### **StepwisePlugin**

TODO.

### *Inputs*

### *Outputs*

### **StepwiseAnalysisPlugin**

TODO.

### *Inputs*

### *Outputs*

## **SynonymizerPlugin**

### *Inputs*

Zero or one `net.maizegenetics.pal.ids.IdentifierSynonymizer`. If only a `IdentifierSynonymizer` is supplied, then this runs interactively. If no `IdentifierSynonymizer` and more than one `net.maizegenetics.pal.ids.IdGroup`, then the first `IdGroup` becomes the synonyms applied to the other `IdGroups`. If a `IdentifierSynonymizer` and one or more `IdGroup`, then the synonyms are applied to the `IdGroups`.

### *Outputs*

`net.maizegenetics.pal.ids.IdentifierSynonymizer`

## **TableDisplayPlugin**

This saves `TableReports` to the filesystem. Alternatively ran interactively, the report is displayed in a dialog.

### *Inputs*

One or more `net.maizegenetics.pal.report.TableReport`.

### *Outputs*

Saves each input to a file specified by method `addSaveFile()`.

## **TreeDisplayPlugin**

This draws trees based on the input tree definition. It can be either a rectangular or circular tree. These can be output to many different formats (e.g. SVG, PNG, JPG).

### *Inputs*

One `net.maizegenetics.pal.tree.Tree`. `net.maizegenetics.pal.alignment.Phenotype` can be used to plot phenotype values onto the tree.

### *Outputs*

Non-interactively, this saves tree to file specified by method `setSaveFile()`.

## **UnionAlignmentPlugin**

This returns the union of given inputs.

### *Inputs*

Zero or more `net.maizegenetics.pal.alignment.AnnotationAlignment` and zero or more `net.maizegenetics.pal.alignment.Phenotype`. Must be at least two inputs any combination of mentioned types.

### *Outputs*

If all inputs are `AnnotationAlignments`, then this returns `net.maizegenetics.pal.alignment.MarkerAlignment`. If all inputs are `Phenotypes`, then this returns `net.maizegenetics.pal.alignment.SimplePhenotype`. If the inputs are mixed, then this returns `net.maizegenetics.pal.alignment.SimpleMarkerPhenotype`.

## **WriteXLSPlugin**

TODO.

*Inputs*

*Outputs*

## **chart.ChartDisplayPlugin**

*Inputs*

One net.maizegenetics.pal.report.TableReport.

*Outputs*

If chart mode: Histogram, save resulting net.maizegenetics.baseplugins.chart.HistogramPanel as file.

If chart mode: XYScatter, save resulting net.maizegenetics.baseplugins.chart.XYScatterPanel as file.

If chart mode: BarChart, save resulting net.maizegenetics.baseplugins.chart.BarChartPanel as file.

If chart mode: PieChart, save resulting net.maizegenetics.baseplugins.chart.PieChartPanel as file.

## **gdpc.ConvertGDPCToPALPlugin**

This converts GDPC entities into PAL entities.

*Inputs*

Either one gov.usda.gdpc.GenotypeGroup or one gov.usda.gdpc.GenotypeTable. Also, optional java.lang.Integer can be input to specify index of GenotypeGroup or GenotypeTable to use.

*Outputs*

Resulting net.maizegenetics.pal.alignment.SimpleAnnotatedAlignment.

## **gdpc.GDPCPlugin**

This adapts GDPC to be a TASSEL plugin. Non-interactively, this exposes the GDPC API as a TASSEL plugin. Interactively, this creates a GDPC Browser for access to GDPC data sources.

*Inputs*

One gov.usda.gdpc.Filter or gov.usda.gdpc.DBElementGroup.

*Outputs*

If input gov.usda.gdpc.EnvironmentExperimentFilter, then returns gov.usda.gdpc.EnvironmentExperimentGroup.

If input gov.usda.gdpc.EnvironmentExperimentGroup, gov.usda.gdpc.TaxonGroup, and gov.usda.gdpc.PhenotypeOntologyGroup, then returns gov.usda.gdpc.PhenotypeGroup.

If input gov.usda.gdpc.GenotypeExperimentFilter, then returns gov.usda.gdpc.GenotypeExperimentGroup.

If input gov.usda.gdpc.GenotypeExperimentGroup and gov.usda.gdpc.TaxonGroup, then returns gov.usda.gdpc.GenotypeGroup.

If input gov.usda.gdpc.LocalityFilter, then returns gov.usda.gdpc.LocalityGroup.

If input gov.usda.gdpc.LocusFilter, then returns gov.usda.gdpc.LocusGroup.

If input gov.usda.gdpc.PhenotypeOntologyFilter, then returns gov.usda.gdpc.PhenotypeOntologyGroup.

If input gov.usda.gdpc.Property, then returns gov.usda.gdpc.DistinctPropertyValues.

If input gov.usda.gdpc.StudyFilter, then returns gov.usda.gdpc.StudyGroup.

If input gov.usda.gdpc.TaxonFilter, then returns gov.usda.gdpc.TaxonGroup.

If input gov.usda.gdpc.TaxonParentFilter, then returns gov.usda.gdpc.TaxonParentGroup.

### **numericaltransform.NumericalTransformPlugin**

This does numerical transformations of data sets, e.g. principal components, standardization, normalization, log, and power transformations. It is currently not friendly to the pipeline running.

#### ***Inputs***

One or more net.maizegenetics.pal.alignment.Phenotype.

#### ***Outputs***

???????????

### **pedigree.PedigreePlugin**

Still in development.

#### ***Inputs***

#### ***Outputs***

### **snpassay.ExtractSNPAssaysPlugin**

#### ***Inputs***

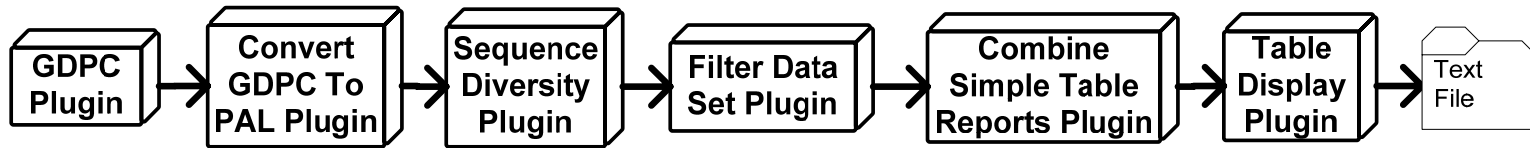
One or more net.maizegenetics.pal.alignment.AnnotationAlignment.

#### ***Outputs***

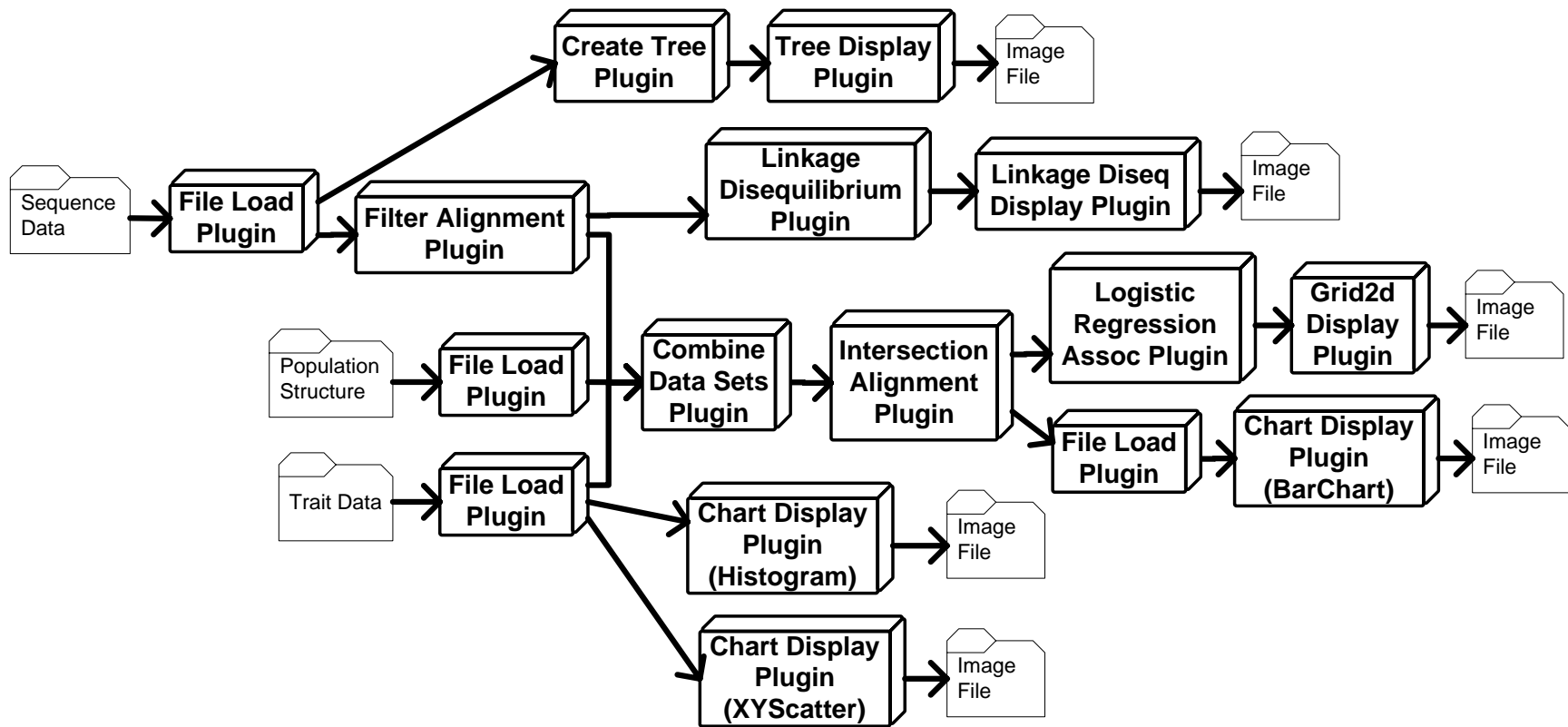
One at a time, this returns one net.maizegenetics.pal.report.SimpleTableReport for each input.

## Appendix A: Example Pipeline Diagrams

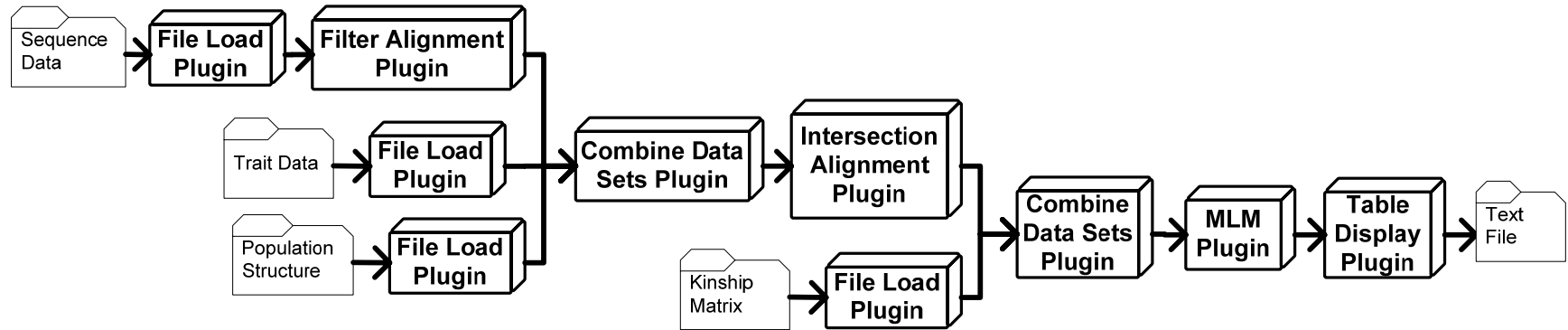
### Diversity Pipeline



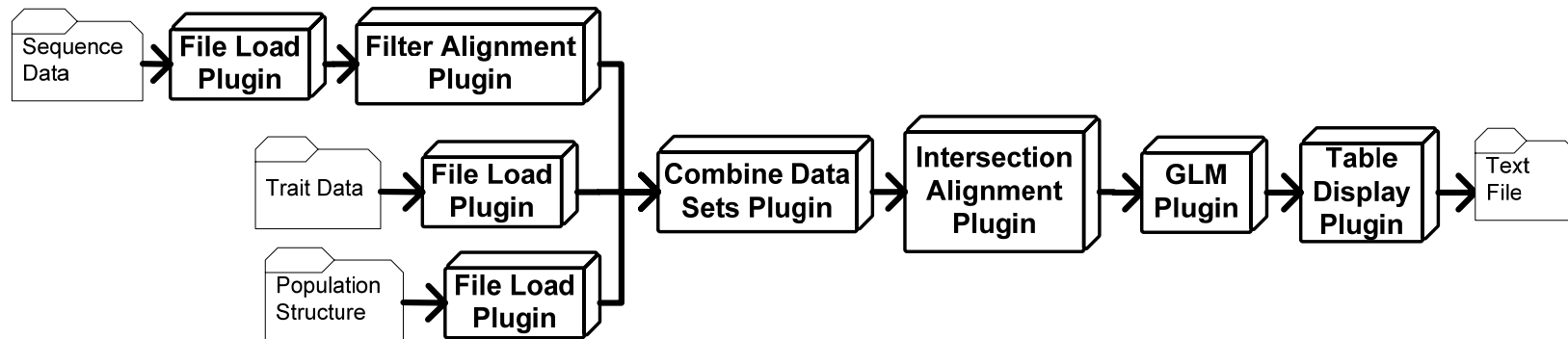
### Graphing Pipeline



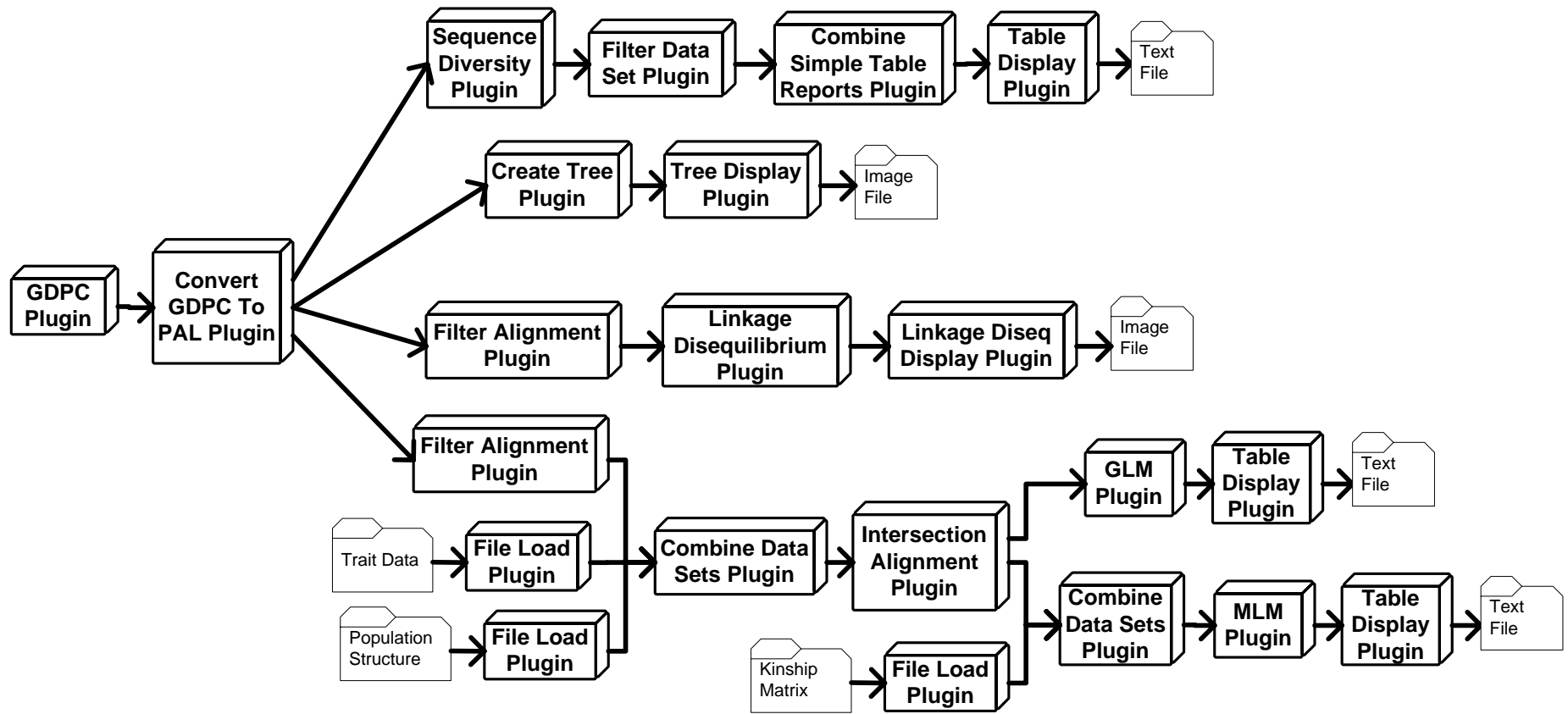
## MLM Pipeline



## GLM Pipeline



# Gramene Pipeline



## Appendix B: Example Source Code – MLM and GLM Pipeline

```
1 /*
2  * MLMGLMPipeline.java
3  *
4  * Created on May 23, 2008
5  *
6  */
7
8 package net.maizegenetics.baseplugins.test;
9
10
11 import net.maizegenetics.baseplugins.*;
12 import net.maizegenetics.baseplugins.database.WriteAnaSampleTablePlugin;
13 import net.maizegenetics.baseplugins.database.WriteAnaTraitAssocTablePlugin;
14 import net.maizegenetics.plugindef.AbstractPlugin;
15 import net.maizegenetics.plugindef.DataSet;
16 import net.maizegenetics.plugindef.Datum;
17
18 import net.maizegenetics.stats.GLM.DataDefinitionDialog;
19 import net.maizegenetics.stats.GLM.ModelBuilder;
20
21 import java.io.File;
22
23
24
25 /**
26  *
27  * @author Terry, Peter, Zhiwu, and Ed
28  */
29 public class MLMGLMPipeline {
30
31     //private GenotypeTransformPlugin myTransformer;
32
33     private int myIteration = -1;
34     private String myExpName = "";
35
36     private WriteAnaSampleTablePlugin myWriteAnaTableResult = null;
37     private CombineDataSetsPlugin mySampleCombinePlugin = null;
38
39     private CombineDataSetsPlugin myGLMWriteCombinePlugin = null;
40
41     private CombineDataSetsPlugin myMLMWriteCombinePlugin = null;
42
43     private CombineDataSetsPlugin myMLMCombineInputPlugin = null;
44     private CombineDataSetsPlugin myMLMCombinePlugin = null;
45     private MLMPlugin myMLMResult = null;
46     private TableDisplayPlugin myMLMTableDisplay = null;
47
48     private CombineDataSetsPlugin myGLMCombinePlugin = null;
49     private GLMPlugin myGLMResult = null;
50     private TableDisplayPlugin myGLMTableDisplay = null;
51
52     private WriteXLSPPlugin myWriteXLSPPlugin = null;
53
54
55     private String myTraitFile = null;
56
57     private String mySNPFile = null;
58
59     private String myKinshipFile = null;
60
61     private String myPopulationFile = null;
62
63     private String myOutputFile = null;
64
65     private boolean myDoMLM = false;
66
67     private boolean myDoGLM = false;
68
69
70     /**
71      * Creates a new instance of MLMGLMPipeline
72      */
73     public MLMGLMPipeline(String args[]) {
```

```

74     handleArgs(args);
75     init();
76     myWriteXLSPugin.saveFile();
77     System.exit(0);
78 }
79
80
81 public static void main(String args[]) {
82     MLMGLMPipeline pipeline = new MLMGLMPipeline(args);
83 }
84
85
86 private void handleArgs(String args[]) {
87
88     int index = 0;
89     while (index < args.length) {
90
91         try {
92
93             String current = args[index++];
94
95             if (!current.startsWith("-")) {
96                 printUsage();
97                 System.exit(0);
98             }
99
100            if (current.equalsIgnoreCase("-t")) {
101                myTraitFile = args[index++].trim();
102            } else if (current.equalsIgnoreCase("-s")) {
103                mySNPFile = args[index++].trim();
104            } else if (current.equalsIgnoreCase("-k")) {
105                myKinshipFile = args[index++].trim();
106            } else if (current.equalsIgnoreCase("-o")) {
107                myOutputFile = args[index++].trim();
108            } else if (current.equalsIgnoreCase("-q")) {
109                myPopulationFile = args[index++].trim();
110            } else if (current.equalsIgnoreCase("-mlm")) {
111                myDoMLM = true;
112            } else if (current.equalsIgnoreCase("-glm")) {
113                myDoGLM = true;
114            } else {
115                printUsage();
116                System.exit(0);
117            }
118
119        }
120
121        catch (Exception e) {
122            e.printStackTrace();
123        }
124
125    }
126
127    if (myTraitFile == null) {
128        printUsage();
129        System.out.println("Error: Trait File must be specified.");
130        System.exit(0);
131    }
132
133    if (mySNPFile == null) {
134        printUsage();
135        System.out.println("Error: SNP File must be specified.");
136        System.exit(0);
137    }
138
139    if (myKinshipFile == null) {
140        printUsage();
141        System.out.println("Error: Kinship File must be specified.");
142        System.exit(0);
143    }
144
145    if (myPopulationFile == null) {
146        printUsage();
147        System.out.println("Error: Population File must be specified.");
148        System.exit(0);
149    }

```

```

150
151     if (myOutputFile == null) {
152         printUsage();
153         System.out.println("Error: Output File must be specified.");
154         System.exit(0);
155     }
156
157 }
158
159
160 private void printUsage() {
161     System.out.println("Usage: java net.maizegenetics.baseplugins.test.MLMGLMPipeline -t <trait file> -s
<SNP file> -k <kinship file> -q <population file> -o <output .xls file> [-glm] [-mlm]");
162 }
163
164
165 private void init() {
166     setupWriteXLS();
167
168     if (myDoMLM) {
169         setupMLM();
170     }
171
172     if (myDoGLM) {
173         setupGLM();
174     }
175
176     tracePipeline();
177
178 }
179
180
181
182 private void tracePipeline() {
183
184     System.out.println("Pipeline Setup....");
185
186     if (myWriteXLSPlugin != null) {
187         ((AbstractPlugin) myWriteXLSPlugin).reverseTrace(0);
188     }
189
190 }
191
192 private void setupWriteXLS() {
193
194     myWriteXLSPlugin = new WriteXLSPlugin(myOutputFile);
195 }
196
197
198 private void setupMLM() {
199
200     FileLoadPlugin filePlugin1 = new FileLoadPlugin(null, false);
201     filePlugin1.setTheFileType(FileLoadPlugin.TasselFileType.Sequence);
202     filePlugin1.setOpenFiles(new File[] {new File(mySNPFile)});
203
204     FileLoadPlugin filePlugin2 = new FileLoadPlugin(null, false);
205     filePlugin2.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
206     filePlugin2.setOpenFiles(new File[] {new File(myTraitFile)});
207
208     FileLoadPlugin filePlugin3 = new FileLoadPlugin(null, false);
209     filePlugin3.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
210     filePlugin3.setOpenFiles(new File[] {new File(myPopulationFile)});
211
212     FileLoadPlugin filePlugin4 = new FileLoadPlugin(null, false);
213     filePlugin4.setTheFileType(FileLoadPlugin.TasselFileType.SqrMatrix);
214     filePlugin4.setOpenFiles(new File[] {new File(myKinshipFile)});
215
216
217     myMLMResult = new MLMPlugin(null, false);
218
219     SetColumnTypesPlugin setColumnTypesPlugin = new SetColumnTypesPlugin(myMLMResult);
220     setColumnTypesPlugin.receiveTraitData(filePlugin2);
221     setColumnTypesPlugin.receivePopulationStructure(filePlugin3);
222
223     myMLMResult.setAnalyzeByColumn(true);
224     myMLMResult.setMaximumNumOfIteration(200);

```

```

225
226     IntersectionAlignmentPlugin intersectPlugin = new IntersectionAlignmentPlugin();
227
228     myMLMCombinePlugin = new CombineDataSetsPlugin();
229     myMLMCombinePlugin.receiveDataSetFrom(filePlugin1);
230     myMLMCombinePlugin.receiveInput(setColumnTypesPlugin);
231
232     intersectPlugin.receiveInput(myMLMCombinePlugin);
233
234     myMLMCombineInputPlugin = new CombineDataSetsPlugin();
235     myMLMCombineInputPlugin.receiveDataSetFrom(intersectPlugin);
236     myMLMCombineInputPlugin.receiveDataSetOnceFrom(filePlugin4);
237
238     myMLMResult.receiveInput(myMLMCombineInputPlugin);
239
240     PassThroughPlugin methodPassThrough = new PassThroughPlugin();
241
242     myMLMWriteCombinePlugin = new CombineDataSetsPlugin();
243     myMLMWriteCombinePlugin.receiveDataSetFrom(myMLMResult);
244     myMLMWriteCombinePlugin.receiveDataSetOnceFrom(methodPassThrough);
245
246     Datum method = new Datum(WriteAnaTraitAssocTablePlugin.METHOD_TAG, "MLM", null);
247     methodPassThrough.performFunction(new DataSet(method, null));
248
249     myWriteXLSPlugin.receiveInput(myMLMWriteCombinePlugin);
250
251     filePlugin3.performFunction(null);
252     filePlugin1.performFunction(null);
253     filePlugin2.performFunction(null);
254     filePlugin4.performFunction(null);
255
256 }
257
258
259 private void setupGLM() {
260
261     FileLoadPlugin filePlugin1 = new FileLoadPlugin(null, false);
262     filePlugin1.setTheFileType(FileLoadPlugin.TasselFileType.Sequence);
263     filePlugin1.setOpenFiles(new File[] {new File(mySNPFFile)});
264
265     FileLoadPlugin filePlugin2 = new FileLoadPlugin(null, false);
266     filePlugin2.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
267     filePlugin2.setOpenFiles(new File[] {new File(myTraitFile)});
268
269     FileLoadPlugin filePlugin3 = new FileLoadPlugin(null, false);
270     filePlugin3.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
271     filePlugin3.setOpenFiles(new File[] {new File(myPopulationFile)});
272
273
274     myGLMResult = new GLMPlugin(null, false);
275
276     SetColumnTypesPlugin setColumnTypesPlugin = new SetColumnTypesPlugin(myGLMResult);
277     setColumnTypesPlugin.receiveTraitData(filePlugin2);
278     setColumnTypesPlugin.receivePopulationStructure(filePlugin3);
279
280     myGLMResult.setAnalyzeByColumn(false);
281     myGLMResult.setAreMarkersInModel(true);
282
283
284     IntersectionAlignmentPlugin intersectPlugin = new IntersectionAlignmentPlugin();
285
286     myGLMResult.receiveInput(intersectPlugin);
287
288     myGLMCombinePlugin = new CombineDataSetsPlugin();
289     myGLMCombinePlugin.receiveDataSetFrom(filePlugin1);
290     myGLMCombinePlugin.receiveInput(setColumnTypesPlugin);
291
292     intersectPlugin.receiveInput(myGLMCombinePlugin);
293
294     PassThroughPlugin methodPassThrough = new PassThroughPlugin();
295
296     myGLMWriteCombinePlugin = new CombineDataSetsPlugin();
297     myGLMWriteCombinePlugin.receiveDataSetFrom(myGLMResult);
298     myGLMWriteCombinePlugin.receiveDataSetOnceFrom(methodPassThrough);
299
300     Datum method = new Datum(WriteAnaTraitAssocTablePlugin.METHOD_TAG, "GLM", null);

```

```
301     methodPassThrough.performFunction(new DataSet(method, null));
302
303     myWriteXLSPlugin.receiveInput(myGLMWriteCombinePlugin);
304
305     filePlugin3.performFunction(null);
306     filePlugin1.performFunction(null);
307     filePlugin2.performFunction(null);
308
309 }
310
311 }
312
```

## Appendix C: Example Source Code – Gramene Pipeline

```
1 /*
2  * GramenePipeline.java
3  *
4  * Created on May 23, 2007
5  *
6  */
7
8 package net.maizegenetics.baseplugins.test;
9
10
11 import gov.usda.gdpc.*;
12 import gov.usda.gdpc.axis2.Axis2Proxy;
13
14 import net.maizegenetics.baseplugins.*;
15 import net.maizegenetics.baseplugins.gdpc.ConvertGDPCToPALPlugin;
16 import net.maizegenetics.baseplugins.gdpc.GDPCPlugin;
17
18 import net.maizegenetics.stats.GLM.DataDefinitionDialog;
19 import net.maizegenetics.stats.GLM.ModelBuilder;
20
21 import net.maizegenetics.tassel.Settings;
22
23 import java.io.File;
24
25
26 /**
27  *
28  * @author Terry, Peter, Zhiwu, and Ed
29  */
30 public class GramenePipeline {
31
32     private ConvertGDPCToPALPlugin myPALPlugin;
33     private GDPCPlugin myGDPCPlugin;
34     private GenotypeExperimentGroup myExperiments;
35     private TaxonGroup myTaxa;
36
37     private final Settings mySettings = new Settings();
38
39     private final String myOutputDir = "C:/local/no_backup/GramenePipeline/";
40     private final String myMLMInputDir = "C:/local/no_backup/MLMPipelineData/";
41
42     private int myIteration = 0;
43     private String myExpName = "";
44
45     private TableDisplayPlugin myDiversityResult = null;
46     private TreeDisplayPlugin myTreeDisplayResult = null;
47     private LinkageDiseqDisplayPlugin myLDResult = null;
48     private TableDisplayPlugin myMLMResult = null;
49     private TableDisplayPlugin myGLMResult = null;
50
51
52     /** Creates a new instance of GramenePipeline */
53     public GramenePipeline() {
54         init();
55         execute();
56         System.exit(0);
57     }
58
59
60     public static void main(String args[]) {
61         GramenePipeline pipeline = new GramenePipeline();
62     }
63
64
65     private void init() {
66         setupGDPCInput();
67         //setupDiversity();
68         //setupTreeDisplay();
69         //setupLinkageDiseq();
70         //setupMLM();
71         setupGLM();
72     }
73
```

```

74
75     private void execute() {
76         for (int i=0; i<5; i++) {
77             myIteration = i;
78             GenotypeExperiment singleExperiment = (GenotypeExperiment) myExperiments.get(i);
79             myExpName = singleExperiment.getName();
80             setFileNames();
81             GenotypeExperimentGroup expItrGroup = new DefaultGenotypeExperimentGroup(new GenotypeExperiment
[] {singleExperiment});
82             GenotypeTable genotypeTable = myGDPCPlugin.getGenotypeTable(expItrGroup, myTaxa);
83         }
84     }
85
86     private void setFileNames() {
87
88         String formattedIteration = getFormattedIteration();
89
90         if (myDiversityResult != null) {
91             String divFile = myOutputDir + "div" + formattedIteration + "_" + myExpName + ".txt";
92             myDiversityResult.setSaveFile(new File(divFile));
93         }
94
95         if (myTreeDisplayResult != null) {
96             String treeFile = myOutputDir + "tree" + formattedIteration + "_" + myExpName + ".png";
97             myTreeDisplayResult.setSaveFile(new File(treeFile));
98         }
99
100         if (myLDResult != null) {
101             String ldFile = myOutputDir + "LD" + formattedIteration + "_" + myExpName + ".png";
102             myLDResult.setSaveFile(new File(ldFile));
103         }
104
105         if (myMLMResult != null) {
106             myMLMResult.clearSaveFiles();
107             String assoFile = myOutputDir + "MLMasso" + formattedIteration + "_" + myExpName + ".txt";
108             String varFile = myOutputDir + "MLMvar" + formattedIteration + "_" + myExpName + ".txt";
109             System.out.println("assoFile: " + assoFile);
110             myMLMResult.addSaveFile(new File(assoFile));
111             myMLMResult.addSaveFile(new File(varFile));
112         }
113
114         if (myGLMResult != null) {
115             String glmFile = myOutputDir + "GLM" + formattedIteration + "_" + myExpName + ".txt";
116             myGLMResult.setSaveFile(new File(glmFile));
117         }
118     }
119
120 }
121
122     private String getFormattedIteration() {
123
124         StringBuilder buffer = new StringBuilder();
125         String num = String.valueOf(myIteration);
126
127         for (int i=0, n=5-num.length(); i<n; i++) {
128             buffer.append("0");
129         }
130
131         buffer.append(num);
132
133         return buffer.toString();
134     }
135
136 }
137
138     private void setupGDPCInput() {
139
140         myGDPCPlugin = new GDPCPlugin();
141         myGDPCPlugin.addDBConnection(new Axis2Proxy("PUT URL HERE"));
142
143         GenotypeExperimentFilter expFilter = new GenotypeExperimentFilter();
144         expFilter.addValue(new FilterSingleValue(GenotypeExperimentProperty.POLY_TYPE,
GenotypeExperimentProperty.POLY_TYPE_SEQUENCE));
145         myExperiments = myGDPCPlugin.getGenotypeExperimentGroup(expFilter);
146     }
147

```

```

148     TaxonFilter taxonFilter = new TaxonFilter();
149     taxonFilter.addValue(new FilterSingleValue(TaxonProperty.GERMPLASM_TYPE, "Inbred"));
150     myTaxa = myGDPCPlugin.getTaxonGroup(taxonFilter);
151
152     myPALPlugin = new ConvertGDPCToPALPlugin();
153     myPALPlugin.receiveInput(myGDPCPlugin);
154 }
155
156
157
158 private void setupDiversity() {
159
160     SequenceDiversityPlugin diversityPlugin = new SequenceDiversityPlugin(null, false, mySettings);
161     diversityPlugin.receiveInput(myPALPlugin);
162
163     FilterDataSetPlugin filterPlugin = new FilterDataSetPlugin(new String []
164 {"Diversity:SimpleAnnotatedAlignment"}, null);
165     filterPlugin.receiveInput(diversityPlugin);
166
167     CombineSimpleTableReportsPlugin reportPlugin = new CombineSimpleTableReportsPlugin();
168     reportPlugin.receiveInput(filterPlugin);
169
170     myDiversityResult = new TableDisplayPlugin(null, false, mySettings);
171     myDiversityResult.setDelimiter(",");
172     myDiversityResult.receiveInput(reportPlugin);
173 }
174
175
176 private void setupTreeDisplay() {
177
178     CreateTreePlugin createTreePlugin=new CreateTreePlugin(null,false);
179     createTreePlugin.setNeighborJoining(false);
180     createTreePlugin.receiveInput(myPALPlugin);
181
182     myTreeDisplayResult = new TreeDisplayPlugin(null, false, mySettings);
183     myTreeDisplayResult.setOutformat(AbstractDisplayPlugin.Outformat.png);
184     myTreeDisplayResult.receiveInput(createTreePlugin);
185 }
186
187
188
189 private void setupLinkageDiseq() {
190
191     FilterAlignmentPlugin alignPlugin = new FilterAlignmentPlugin(null, false, mySettings);
192     alignPlugin.receiveInput(myPALPlugin);
193
194     LinkageDisequilibriumPlugin theLDP=new LinkageDisequilibriumPlugin(null, false, mySettings);
195     theLDP.receiveInput(alignPlugin);
196
197     myLDRResult = new LinkageDiseqDisplayPlugin(null, false, mySettings);
198     myLDRResult.setOutformat(AbstractDisplayPlugin.Outformat.png);
199     myLDRResult.receiveInput(theLDP);
200 }
201
202
203
204 private void setupMLM() {
205
206     FilterAlignmentPlugin alignPlugin = new FilterAlignmentPlugin(null, false, mySettings);
207     alignPlugin.setFilterMinorSNPs(true);
208     alignPlugin.setMinFreq(0.01);
209     alignPlugin.setMinCount(10);
210     alignPlugin.receiveInput(myPALPlugin);
211
212     FileLoadPlugin filePlugin2 = new FileLoadPlugin(null, false, mySettings);
213     filePlugin2.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
214     filePlugin2.setOpenFiles(new File[] {new File(myMLMInputDir + "three_traits_rn.txt")});
215
216
217     FileLoadPlugin filePlugin3 = new FileLoadPlugin(null, false, mySettings);
218     filePlugin3.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
219     filePlugin3.setOpenFiles(new File[] {new File(myMLMInputDir + "popStructure_taxa286_rn.txt")});
220
221     FileLoadPlugin filePlugin4 = new FileLoadPlugin(null, false, mySettings);
222     filePlugin4.setTheFileType(FileLoadPlugin.TasselFileType.SqrMatrix);

```

```

223     filePlugin4.setOpenFiles(new File[] {new File(myMLMInputDir +
"kinship_277taxa_by_Spagedi_rn_sq.txt")});
224
225     CombineDataSetsPlugin combinePlugin = new CombineDataSetsPlugin();
226     combinePlugin.receiveDataSetFrom(alignPlugin);
227     combinePlugin.receiveDataSetOnceFrom(filePlugin2);
228     combinePlugin.receiveDataSetOnceFrom(filePlugin3);
229
230     IntersectionAlignmentPlugin intersectPlugin = new IntersectionAlignmentPlugin();
231     intersectPlugin.receiveInput(combinePlugin);
232
233     CombineDataSetsPlugin allDataPlugin = new CombineDataSetsPlugin();
234     allDataPlugin.receiveDataSetFrom(intersectPlugin);
235     allDataPlugin.receiveDataSetOnceFrom(filePlugin4);
236
237     MLMPlugin mlm = new MLMPlugin(null, false);
238     mlm.receiveInput(allDataPlugin);
239     String[] columnTypes = new String[]{DataDefinitionDialog.TYPE_EXCLUDE,
240     DataDefinitionDialog.TYPE_DATA,
241     DataDefinitionDialog.TYPE_EXCLUDE,
242     DataDefinitionDialog.TYPE_COVARIATE,
243     DataDefinitionDialog.TYPE_COVARIATE,
244     DataDefinitionDialog.TYPE_EXCLUDE};
245     mlm.setColumnTypes(columnTypes);
246     mlm.setAnalyzeByColumn(false);
247     mlm.addFactors(new int[]{ModelBuilder.INT_MARKER});
248     mlm.addFactors(new int[]{3});
249     mlm.addFactors(new int[]{4});
250
251     myMLMResult = new TableDisplayPlugin(null, false, mySettings);
252     myMLMResult.setDelimiter("\t");
253     myMLMResult.receiveInput(mlm);
254
255     filePlugin2.performFunction(null);
256     filePlugin3.performFunction(null);
257     filePlugin4.performFunction(null);
258
259 }
260
261 private void setupGLM() {
262
263     FilterAlignmentPlugin alignPlugin = new FilterAlignmentPlugin(null, false, mySettings);
264     alignPlugin.setStart(1000);
265     alignPlugin.setFilterMinorSNPs(true);
266     alignPlugin.setMinFreq(0.1);
267     alignPlugin.setMinCount(60);
268     alignPlugin.receiveInput(myPALPlugin);
269
270     FileLoadPlugin filePlugin2 = new FileLoadPlugin(null, false, mySettings);
271     filePlugin2.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
272     filePlugin2.setOpenFiles(new File[] {new File(myMLMInputDir + "three_traits_rn.txt")});
273
274     FileLoadPlugin filePlugin3 = new FileLoadPlugin(null, false, mySettings);
275     filePlugin3.setTheFileType(FileLoadPlugin.TasselFileType.Numerical);
276     filePlugin3.setOpenFiles(new File[] {new File(myMLMInputDir + "popStructure_taxa286_rn.txt")});
277
278     CombineDataSetsPlugin combinePlugin = new CombineDataSetsPlugin();
279     combinePlugin.receiveDataSetFrom(alignPlugin);
280     combinePlugin.receiveDataSetOnceFrom(filePlugin2);
281     combinePlugin.receiveDataSetOnceFrom(filePlugin3);
282
283     IntersectionAlignmentPlugin intersectPlugin = new IntersectionAlignmentPlugin();
284     intersectPlugin.receiveInput(combinePlugin);
285
286     GLMPlugin glm = new GLMPlugin(null, false);
287     String[] columnTypes = new String[]{DataDefinitionDialog.TYPE_COVARIATE,
288     DataDefinitionDialog.TYPE_COVARIATE,
289     DataDefinitionDialog.TYPE_EXCLUDE,
290     DataDefinitionDialog.TYPE_DATA,
291     DataDefinitionDialog.TYPE_DATA,
292     DataDefinitionDialog.TYPE_DATA};
293     glm.setColumnTypes(columnTypes);
294     glm.setAnalyzeByColumn(false);
295     glm.setAreMarkersInModel(true);
296     glm.addFactors(new int[]{ModelBuilder.INT_MARKER});

```

```
298     glm.addFactors(new int[]{5});
299     glm.addFactors(new int[]{6});
300     glm.receiveInput(intersectPlugin);
301
302     glm.addFTests(0, ModelBuilder.INT_RESIDUAL, 0);
303
304     myGLMResult = new TableDisplayPlugin(null, false, mySettings);
305     myGLMResult.setDelimiter("\t");
306     myGLMResult.receiveInput(glm);
307
308     filePlugin2.performFunction(null);
309     filePlugin3.performFunction(null);
310
311 }
312
313 }
314
```