# Hapl-o-Mat - Getting Started - Windows

Please also see the README.

## Hapl-o-Mat

Hapl-o-Mat is software for HLA haplotype inference coded in C++. Besides estimating haplotype frequencies via an expectation-maximization algorithm, it is capable of processing HLA genotype population data. This includes translation of alleles between various typing resolutions and resolving allelic and genotypic ambiguities. Both common formats for recording HLA genotypes, multiple allele (NMDP) codes and genotype list strings, are supported.

This guide explains how to use Hapl-o-Mat under Windows. For more information refer to our publications on Hapl-o-Mat:

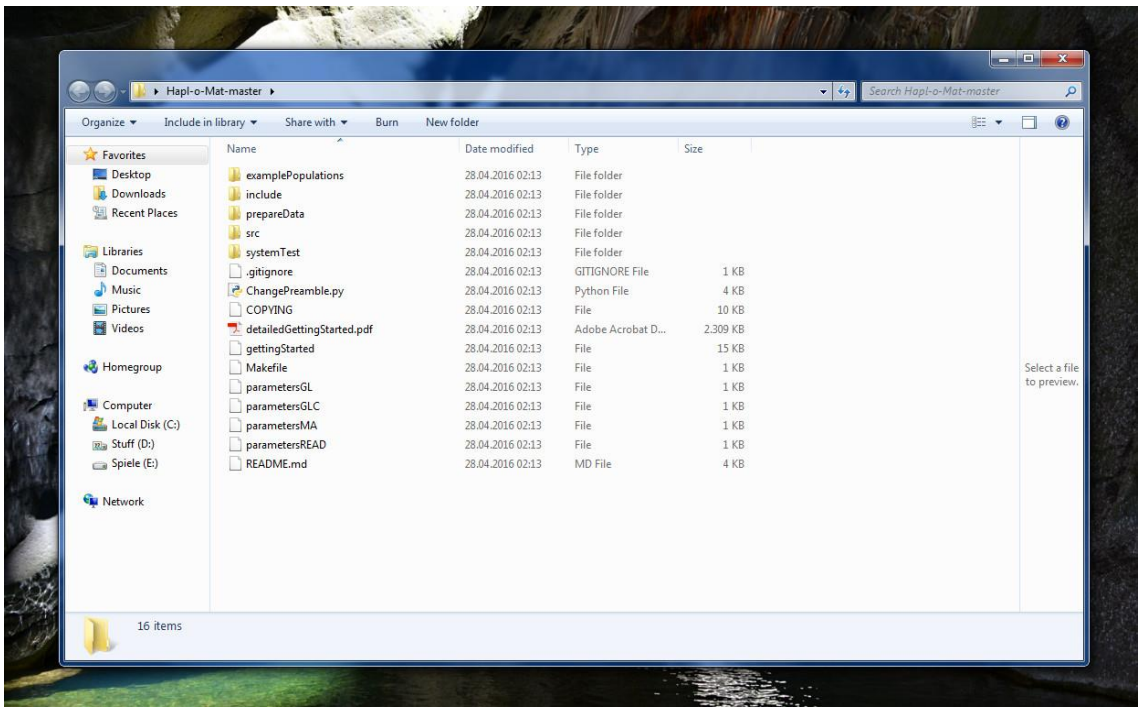
Journal article to come

C. Schaefer, A.H. Schmidt, J. Sauter: Hapl-O-mat: A Versatile Software for Haplotype Frequency Estimation. HLA (2016), 87, 236-320

## Getting Started

This guide is an introduction on how to use Hapl-o-Mat. In order to follow this guide, you need a Windows system and a C++ compiler supporting C++11. In this tutorial, we use Eclipse IDE for C/C++ Developers. You can get it here https://www.eclipse.org/downloads/.

After successfully downloading Hapl-o-Mat, look into the folder. You should see the following:



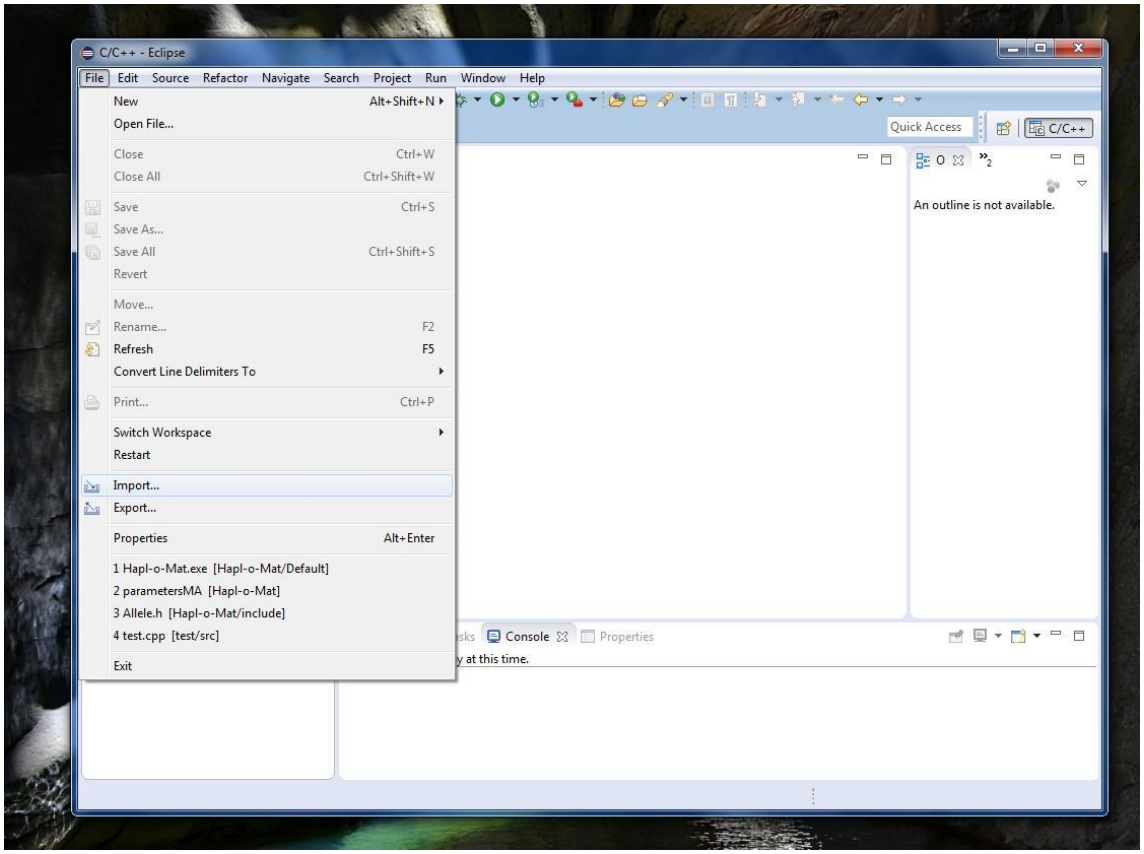

We give you some information on the files you find. Files which are important for using Hapl-o-Mat are marked as bold:

| File name | Description |
|---|---|
| ChangePreamble.py | A python script to adapt the preamble in all files. You are not going to use it. |
| COPYING | The GNU General Public License. |
| detailedGettingStarted | Guides for using Hapl-o-Mat under Windows and Linux |
| **examplePopulations** | Some genotype population data we are going to work with in the section Tutorials. |
| gettingStarted | A shorter form of this tutorial |
| include | A part of Hapl-o-Mat's source code. If you do not want to change code, do not touch it |
| Makefile | Instructions for building Hapl-o-Mat. |
| **parametersGL, parametersGLC, parametersMA, parametersMA** | Parameter files for Hapl-O-mat. We are going to discuss this in section Parameters |
| **prepareData** | Here is everything to create the data required by Hapl-o-Mat |
| **README** | Read me |
| src | A part of Hapl-o-Mat's source code. If you do not want to change code, do not touch it |
| systemTest | Run the system test after changing code to check, if you broke something. |

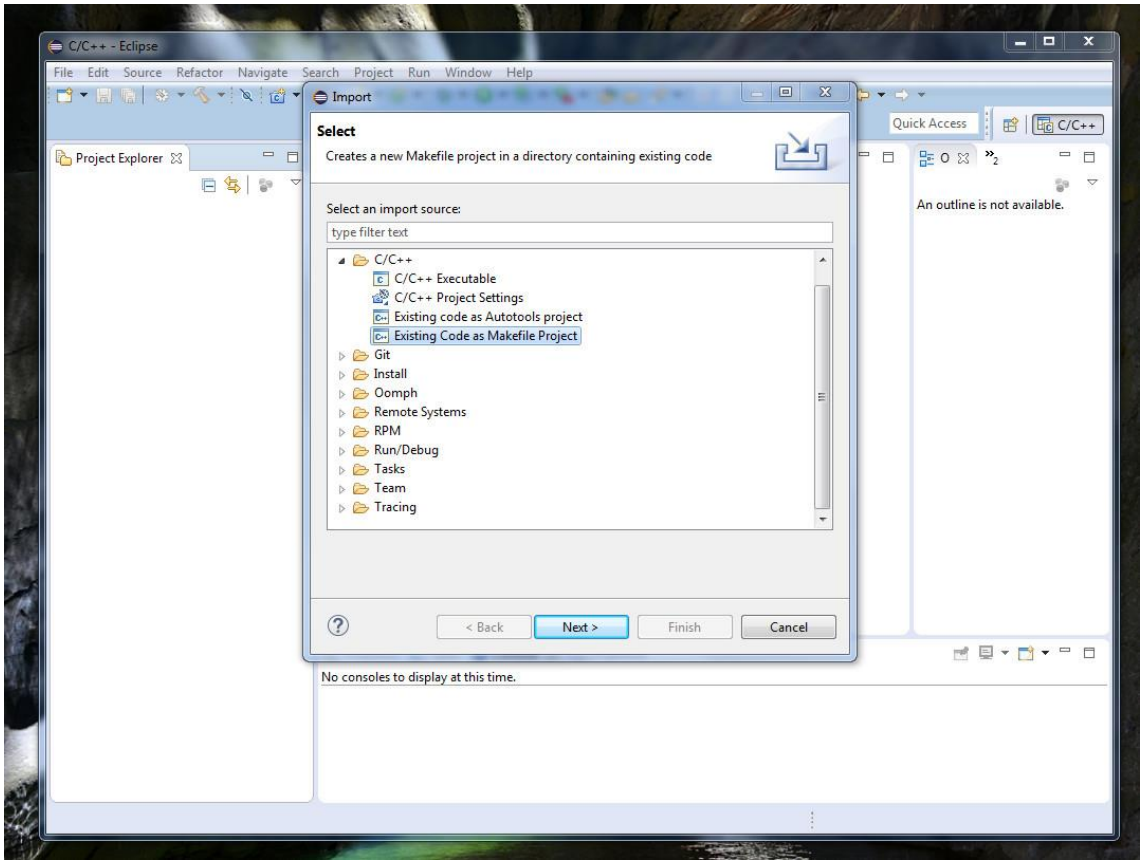
To estimate haplotype frequencies we only need to consider the folder prepareData and the files parametersGL, parametersGLC, parametersMA, and parametersREAD. To finish this tutorial we need the folder examplePopluations, too.

## Install Hapl-o-Mat

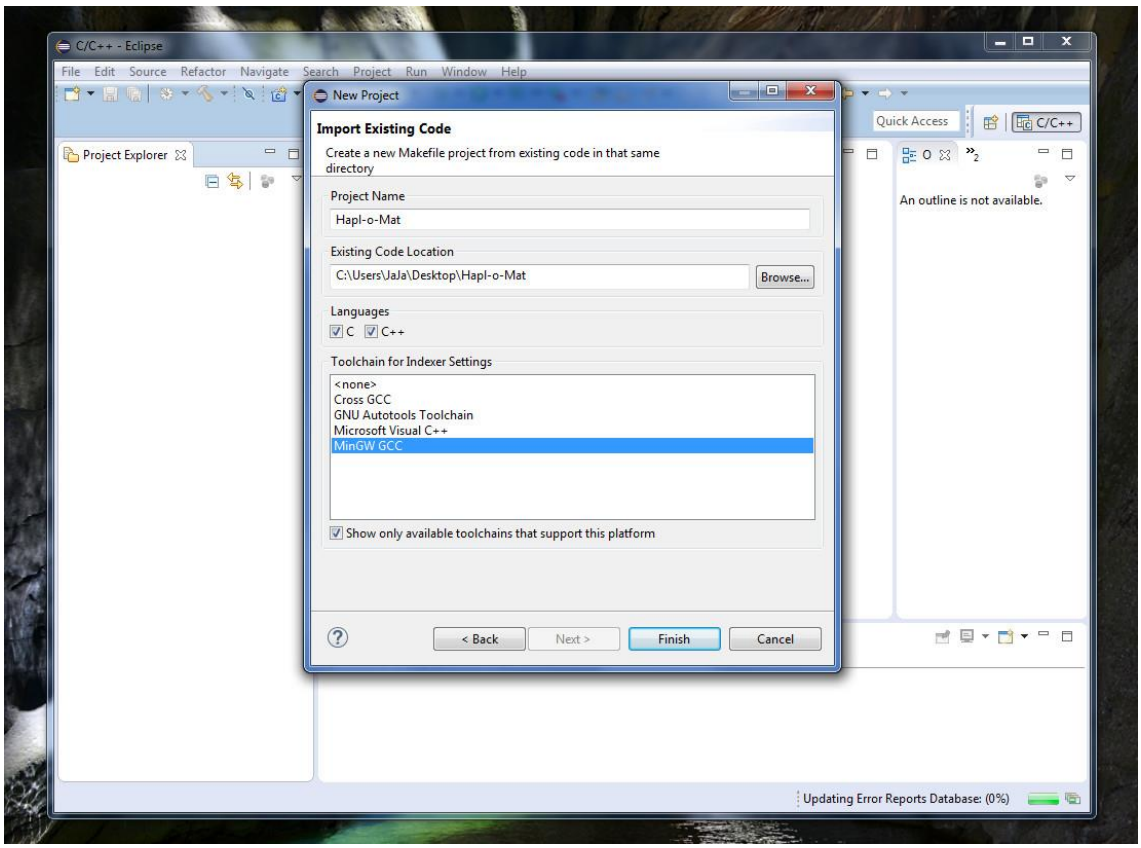
We compile Hapl-o-Mat using Eclipse. Start Eclipse and create a new project by clicking on File -> Import.



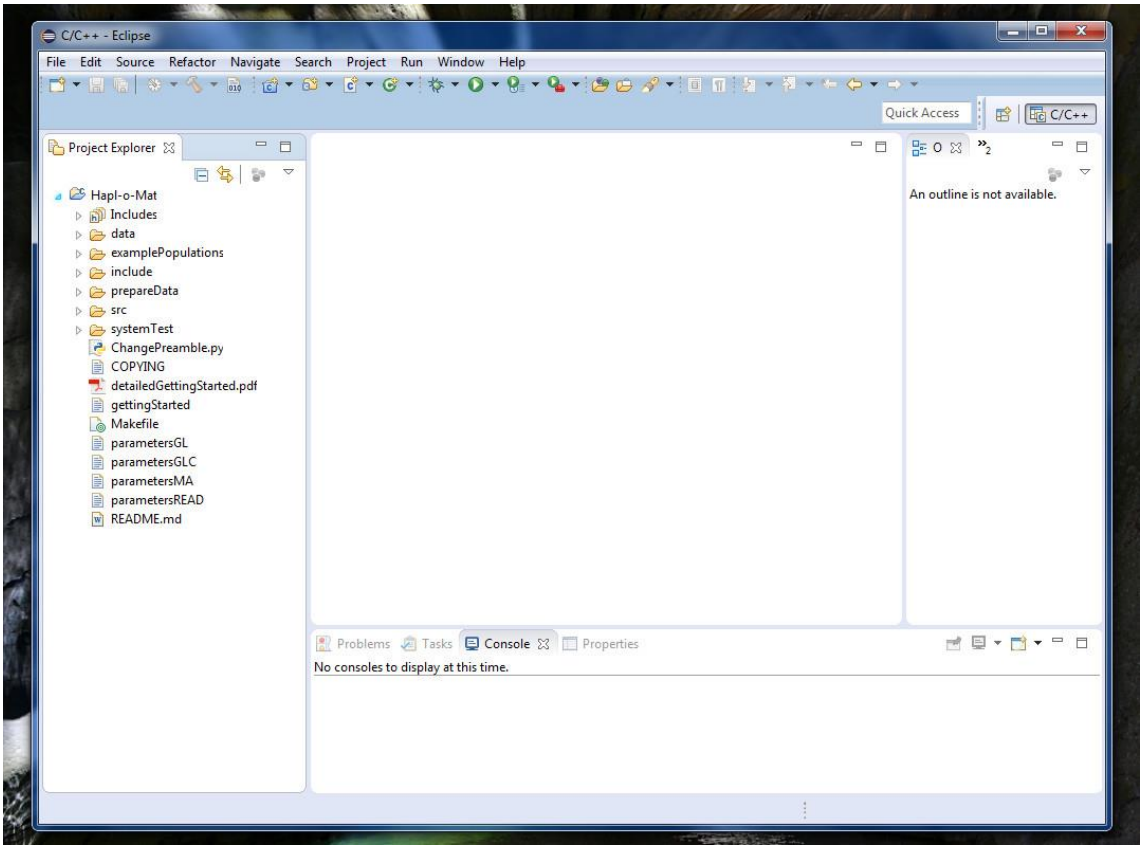

Choose Existing Code as Makefile Project in the pop-up window and then click Next.



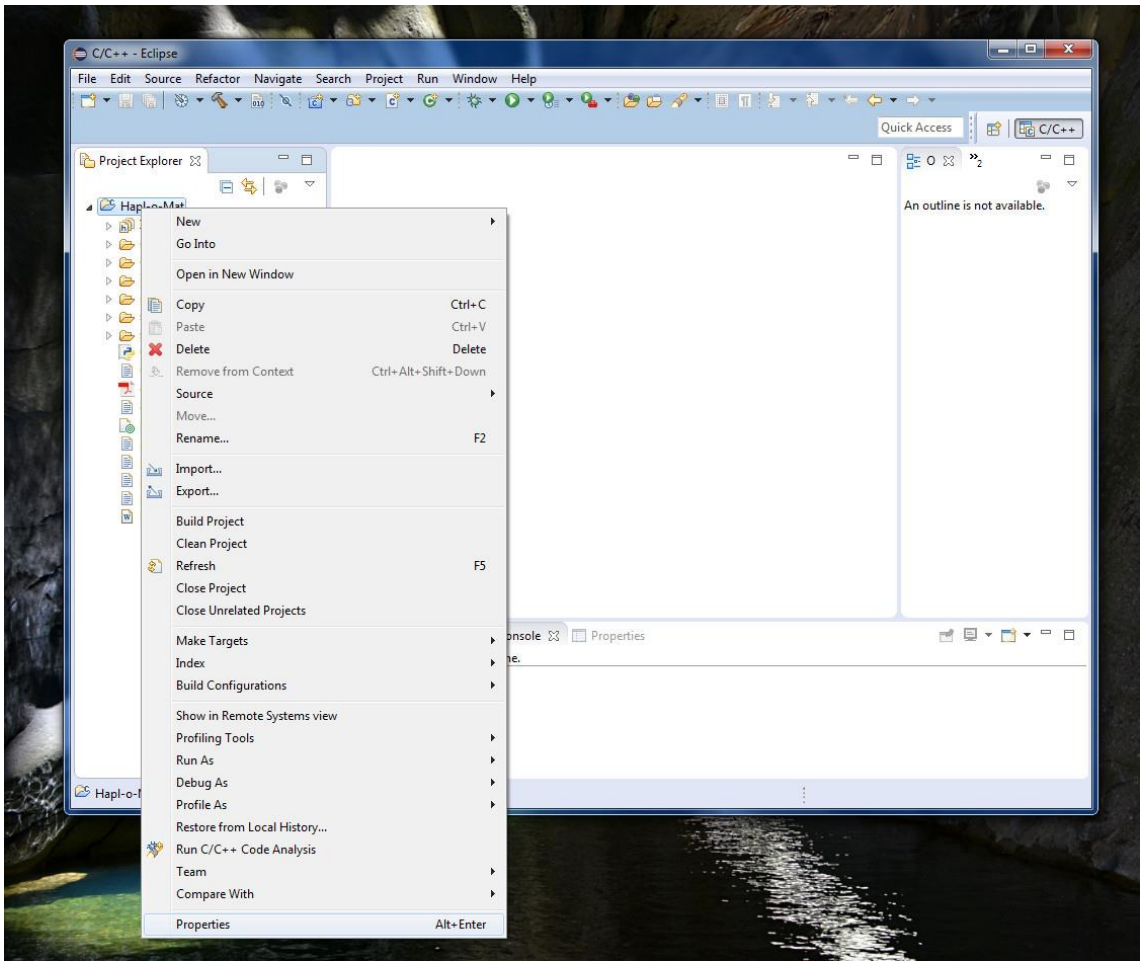
Enter the Project Name, e.g. Hapl-o-Mat, and browse to the location where you saved Hapl-o-Mat. Select MinGW GCC as Toolchain. Click Finish.



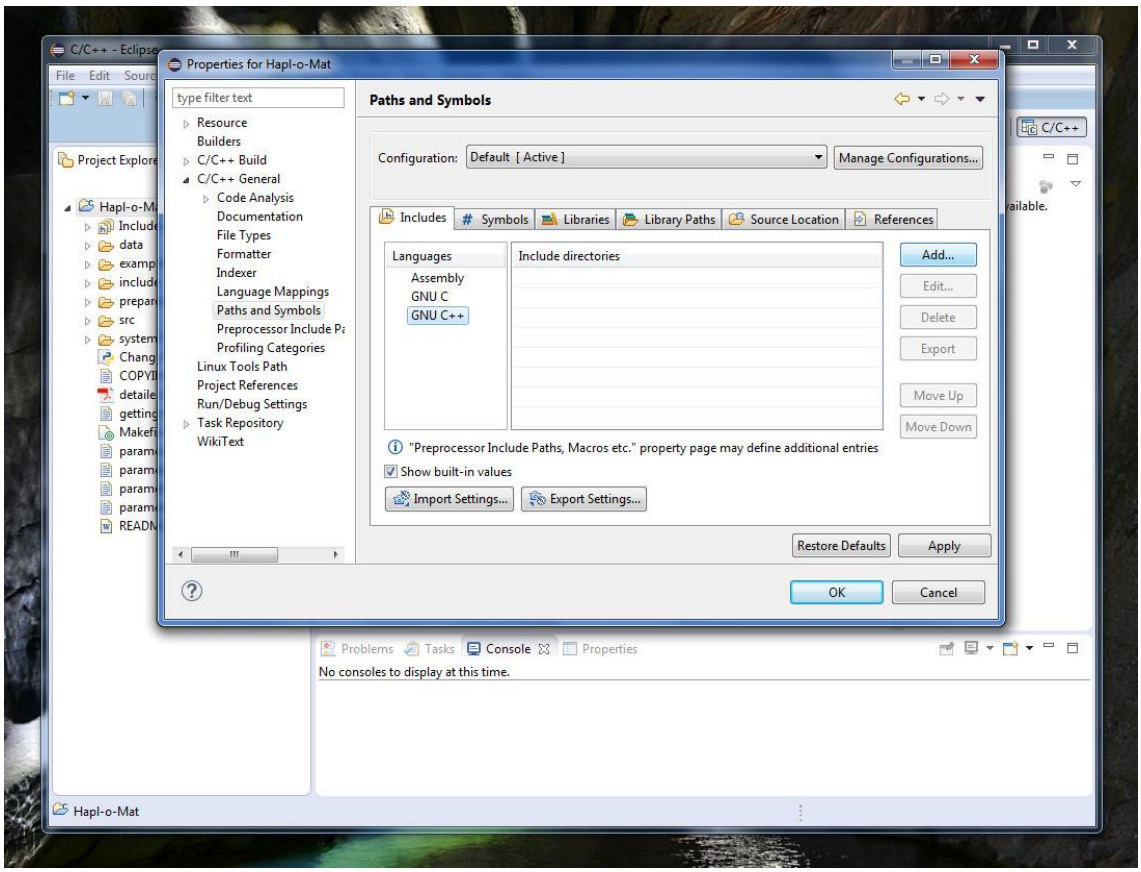

Now, you find all files and folders of Hapl-o-Mat in the Project Explorer. If your Eclipse does not show the Project Explorer, activate it via Window -> Show View -> Project Explorer.



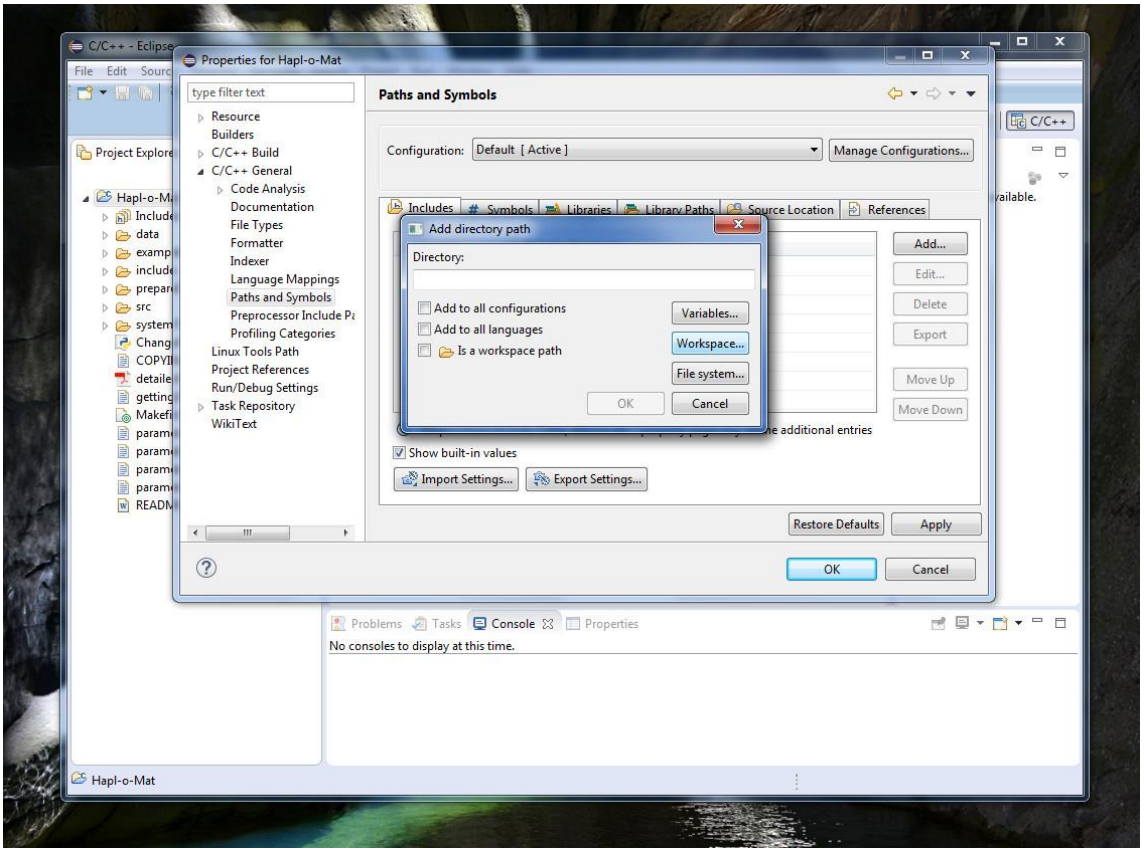

Next, we have to adapt some settings. Right-click into the Project Explorer and choose Properties.



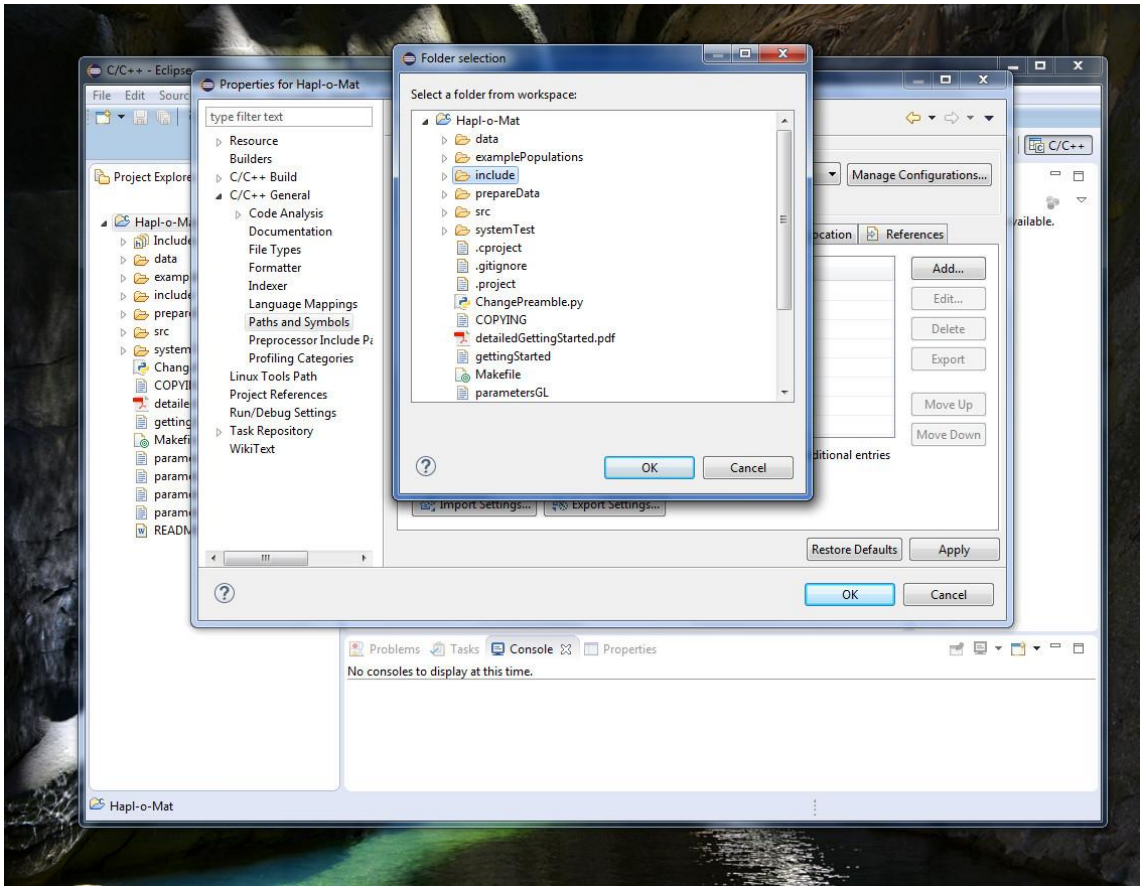
A new window pops up. In the left column choose C/C++ General -> Paths and Symbols. Then, click Add in the right column.



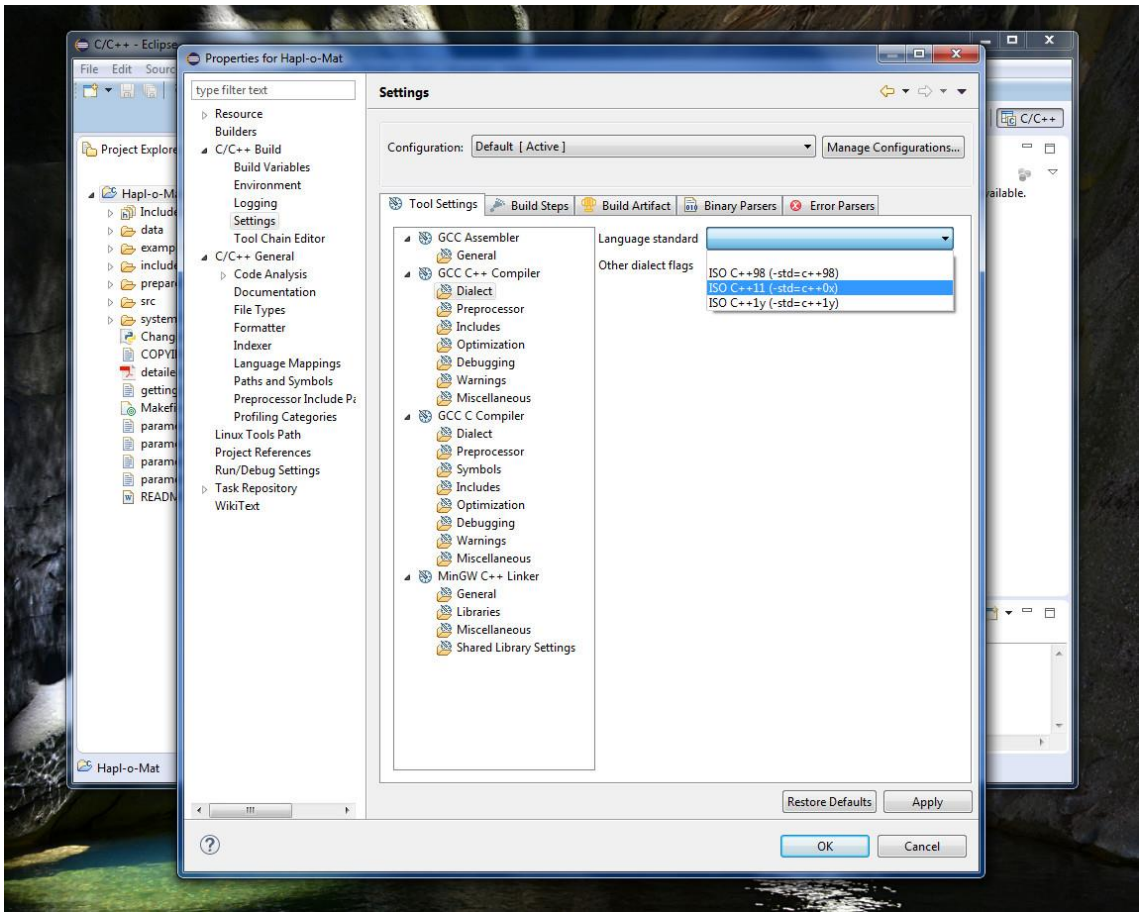

A new window pops up, where you click on Workspace….



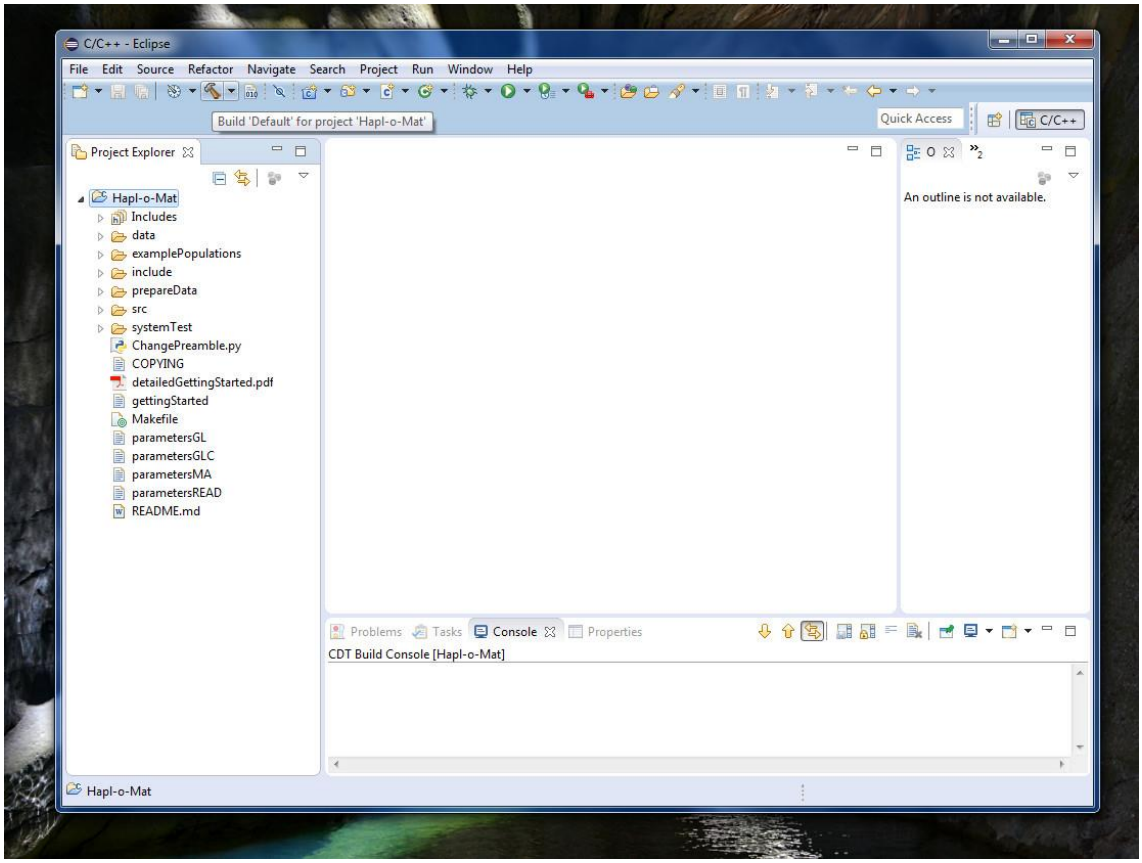

In the following window choose folder include.



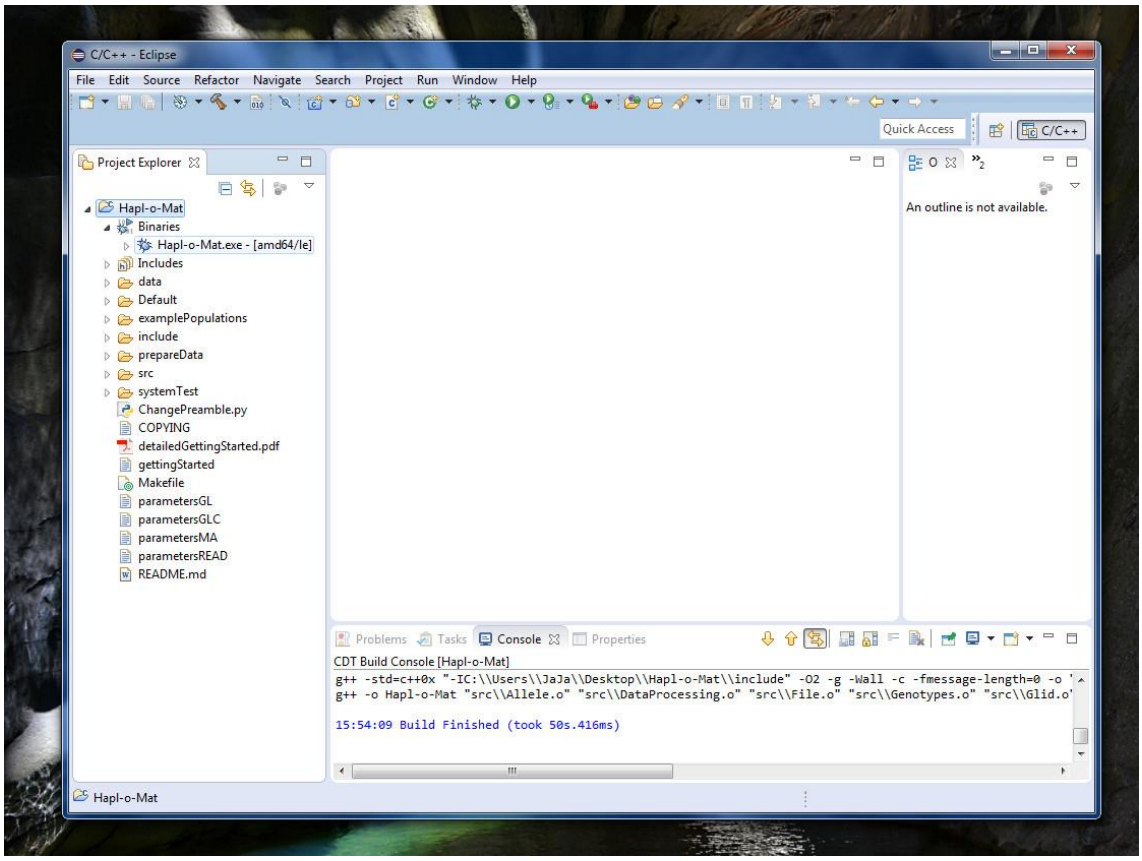
Click two times OK and then choose in the left column C/C++ Build -> Settings. In the right column choose  GCC C/C++ Compiler -> Dialect and then choose ISO C++11(-std=c++0x) as Language standard. Click two times OK.



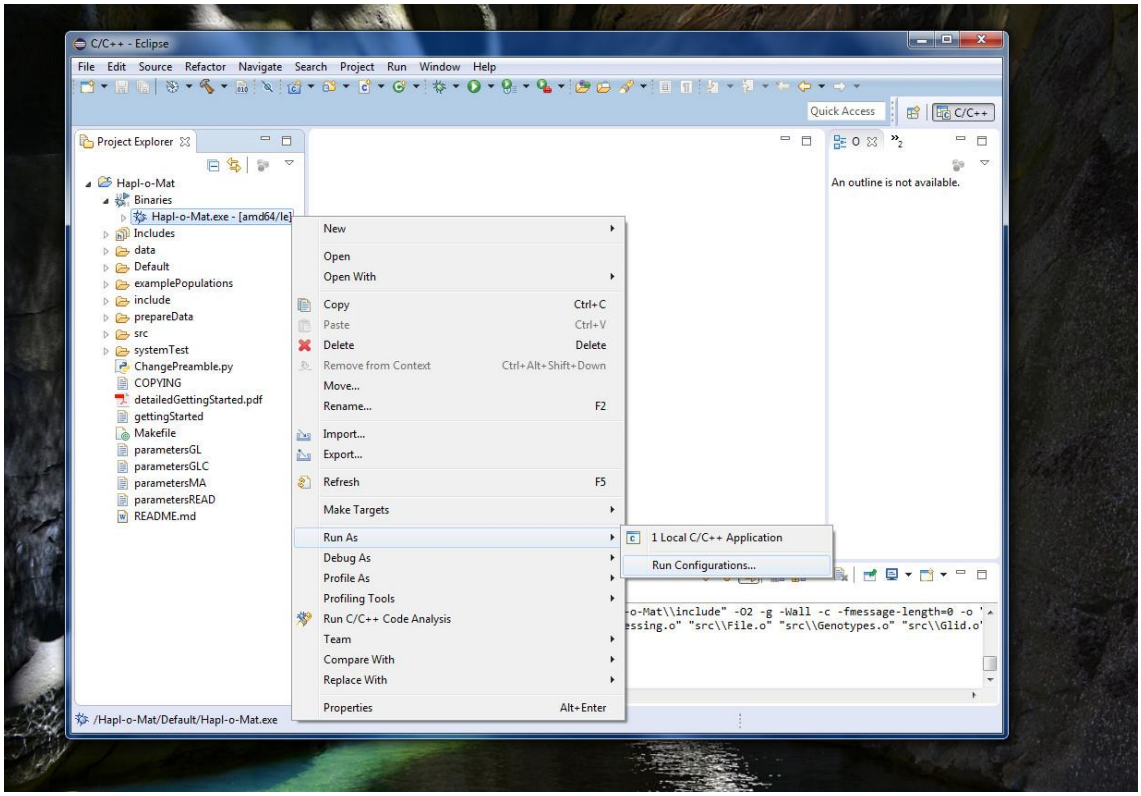

Finally, we compile Hapl-o-Mat by clicking the hammer-symbol in the upper navigation bar.



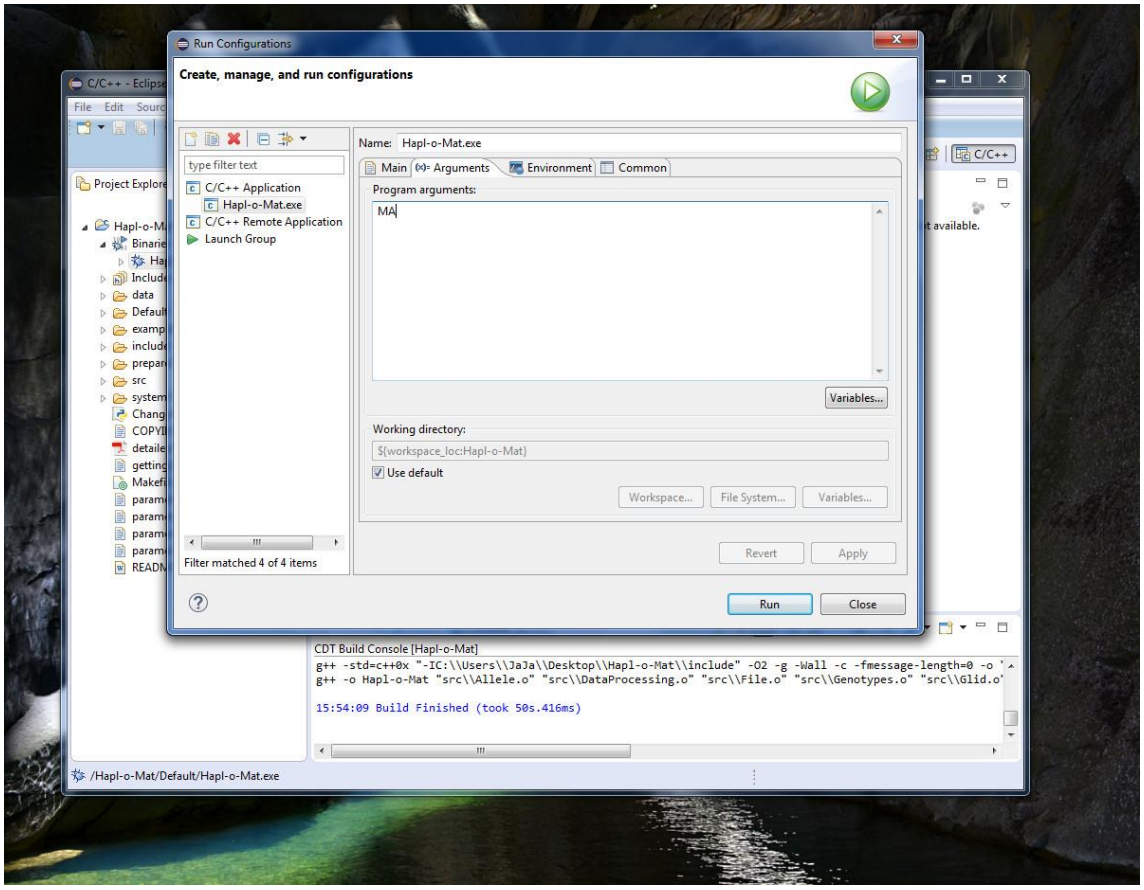

The executable Hapl-o-Mat.exe appeared.



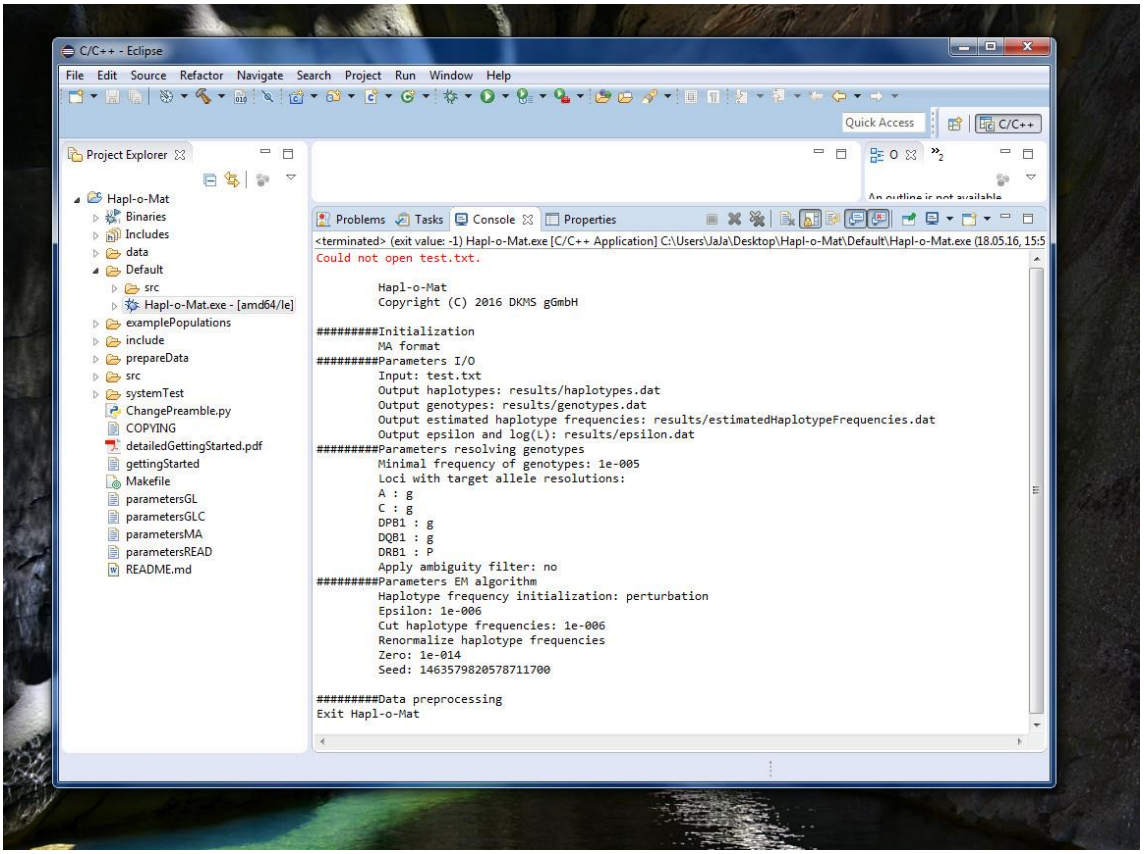

Right-click on it and choose Run As -> Run Configurations….




Click on the tab Arguments and enter MA into the field Program arguments.




Then click Run. Hapl-o-Mat runs now, but produces an error message, since we did not specify any input files. We tackle this later in the tutorial.




Congratulations, you compiled Hapl-o-Mat under Windows.

## Data Preparation

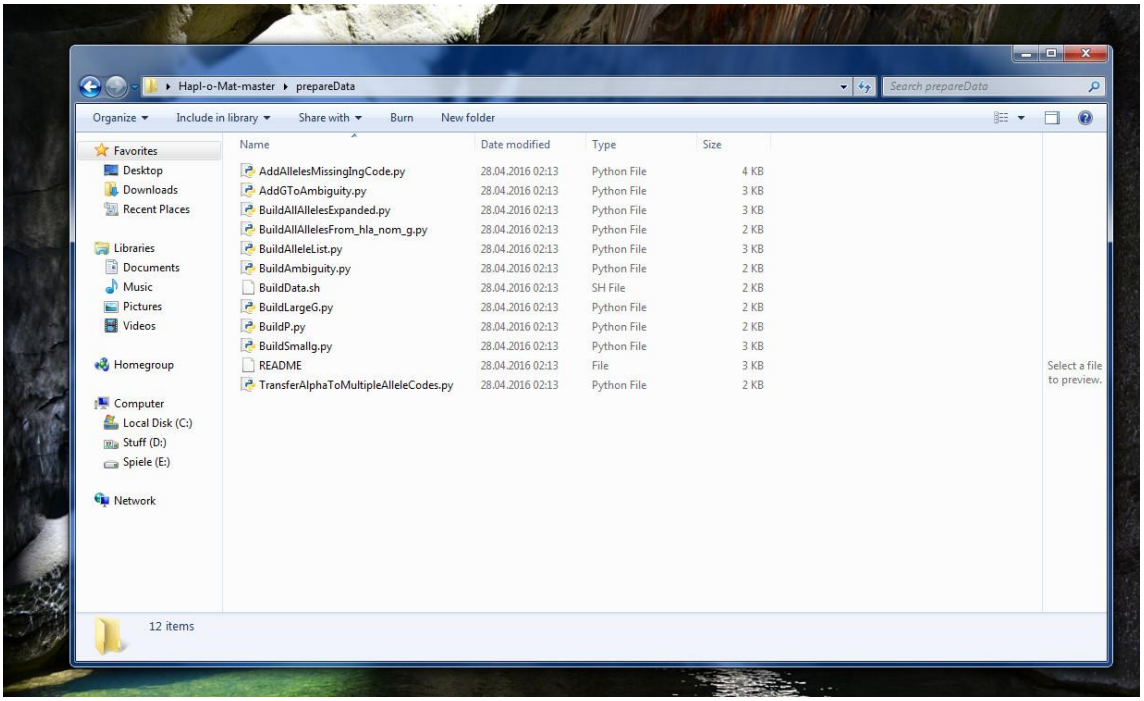
Hapl-o-Mat relies on information on the HLA nomenclature. This information is provided by data files, which we are going to create. As the HLA nomenclature evolves over time, e.g. by finding new alleles or adding new NMDP codes, it is important to update data from time to time. Hapl-o-Mat relies on the following files, which must be placed in the folder “Hapl-o-Mat/data” (we create the folder “data” later):

| **File name** | **Description** |
|---|---|
| AllAllelesExpanded.txt | A list of relevant existing HLA alleles with their enclosed more-digit typing resolutions |
| AlleleList.txt | If your input data in GL format includes a missing single-locus genotype, it can be replaced by combining all alleles of the same locus from this file |
| Ambiguity.txt | Data basis for the ambiguity filter |
| LargeG.txt | A list of G-groups with their enclosed alleles in 8-digit resolution |
| MultipleAlleleCodes.txt | A list of multiple allele codes and their translation to alleles |
| P.txt | A list of P-groups with their enclosed alleles in 8-digit resolution |
| Smallg.txt | A list of g-groups with their enclosed alleles in 8-digit resolution |

In the following we are going to create these data files.

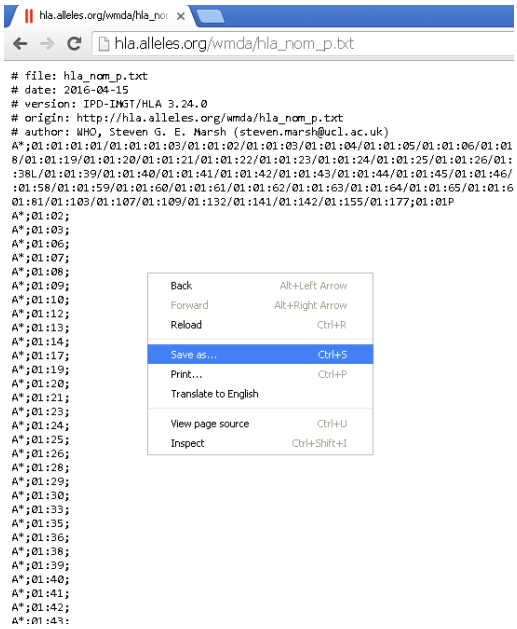
## Download Data

Check the folder “prepareData”.



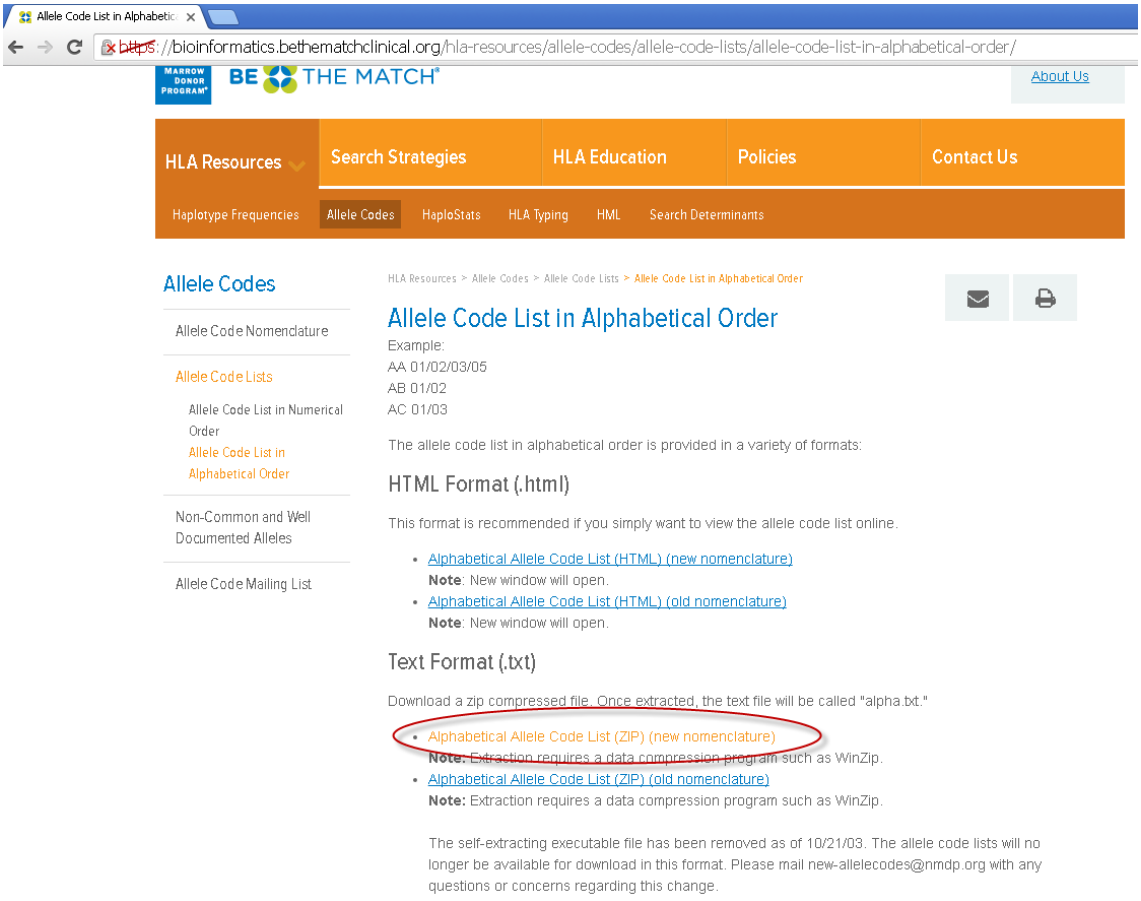

You find a number of Python scripts (files ending with .py) which we are going to use to produce the data. But first we need some information from the web. Save all files in one folder, e.g. “InputData”.

1) Go to the website http://hla.alleles.org/wmda/hla_nom_p.txt and save the file hla_nom_p.txt by right-clicking and choosing “Save as…”



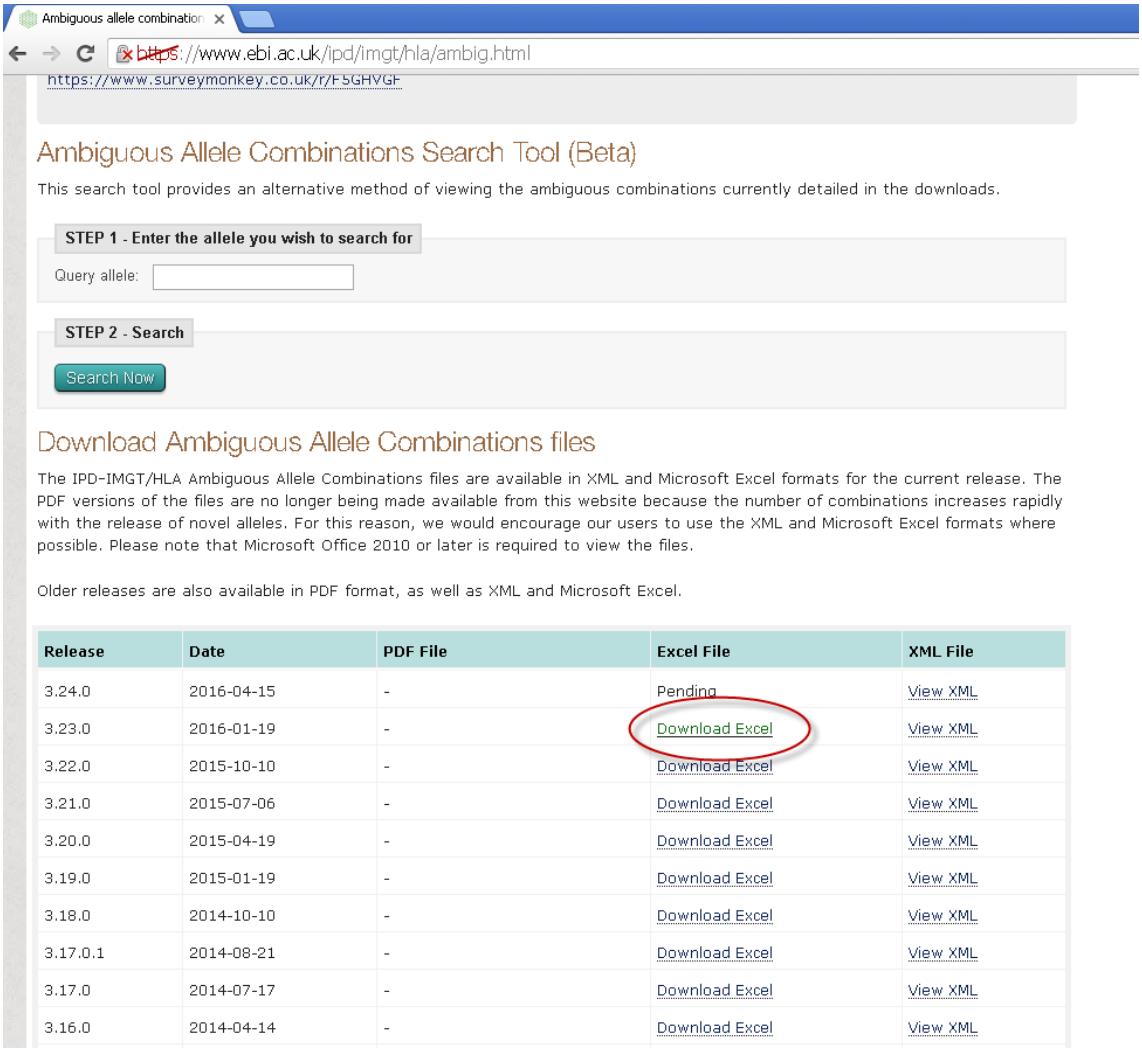

2) Go to the website http://hla.alleles.org/wmda/hla_nom_g.txt and save the file hla_nom_g.txt (same as in 1))

3) Go to the website https://bioinformatics.bethematchclinical.org/HLA-Resources/Allele-Codes/Allele-Code-Lists/Allele-Code-List-in-Alphabetical-Order/. Click on "Alphabetical Allele Code List (ZIP) (new nomenclature)" and save alpha.v3.zip.

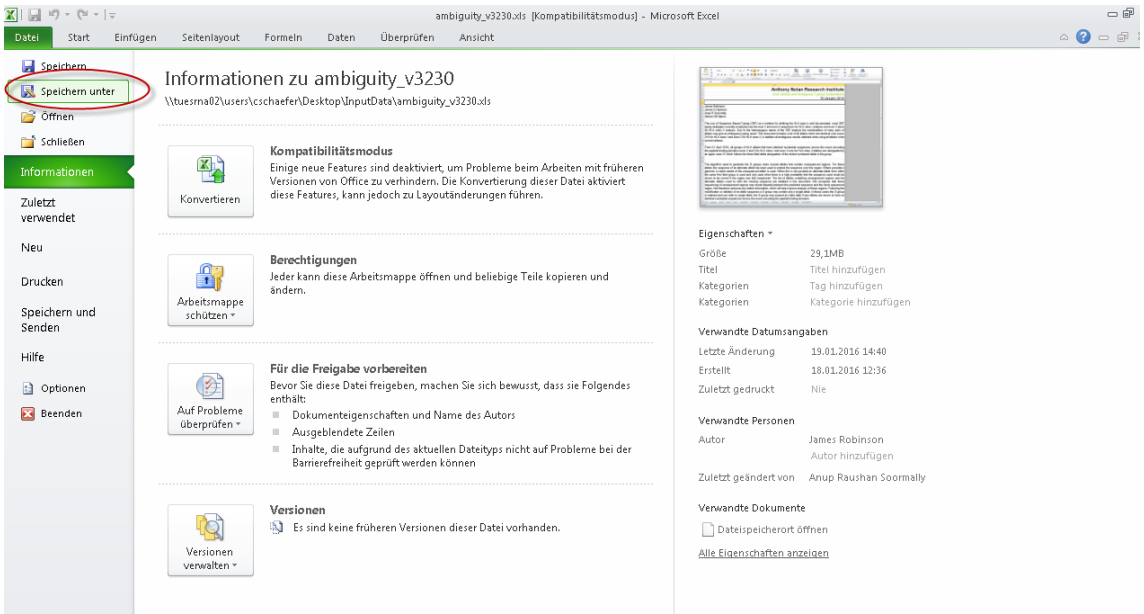


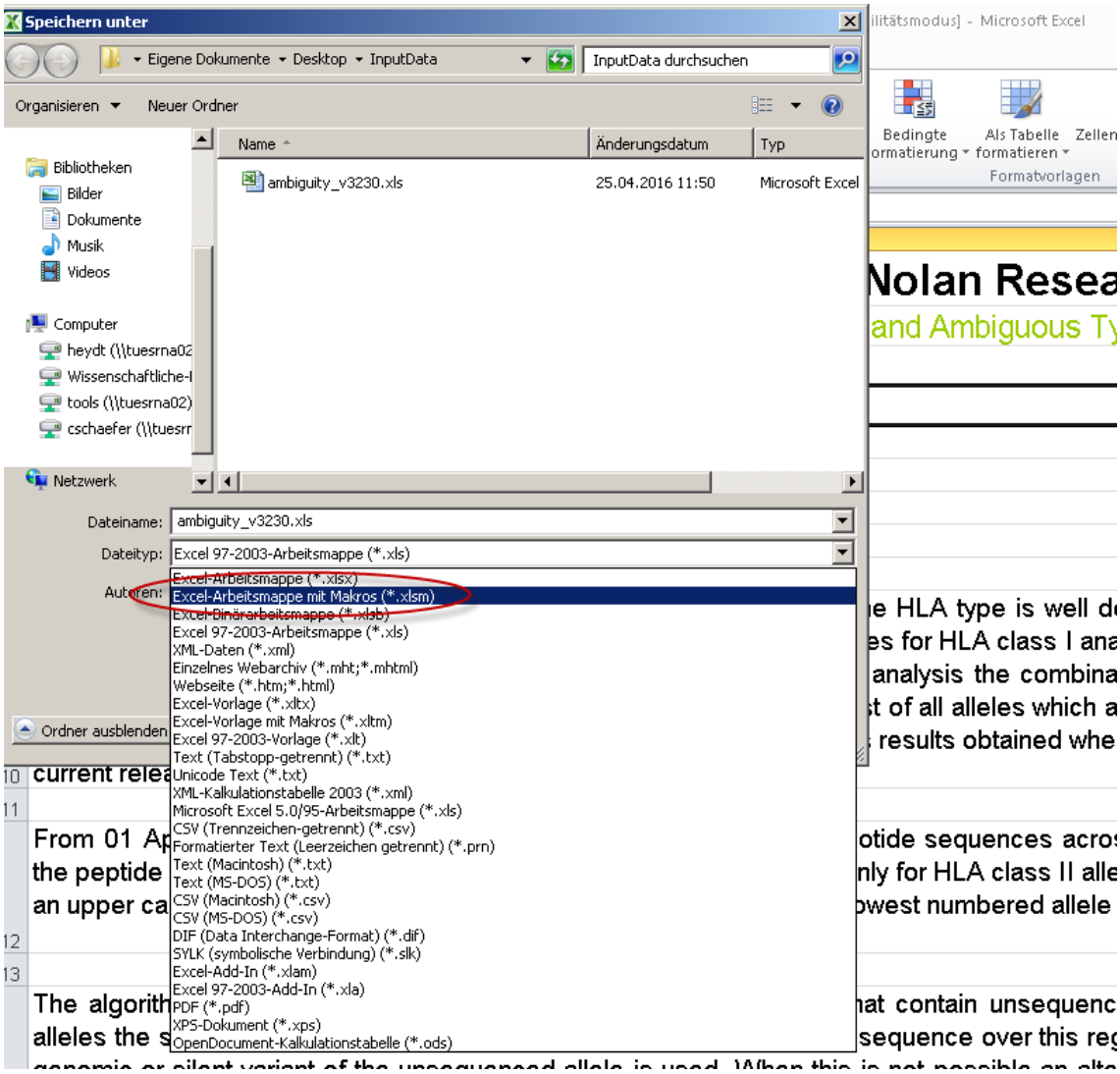

Extract the archive alpha.v3.zip. Afterwards, you can remove the archive.

4) Go to the website https://www.ebi.ac.uk/ipd/imgt/hla/ambig.html. Click on "Download Excel" for the wanted release (usually the latest) and save ambiguity_v<>.xls (replace <> by version).



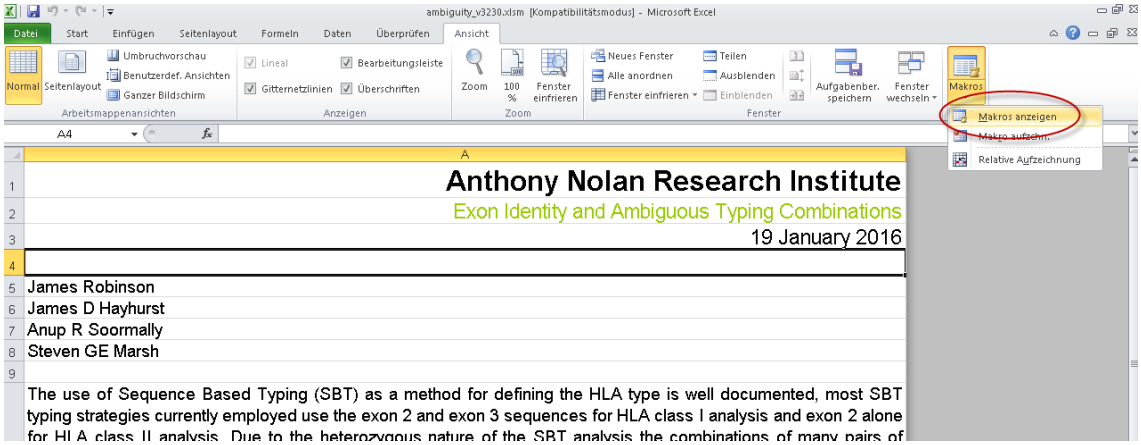

Next we extract the information from the Excel sheet. Open ambiguity_v<>.xls in Excel and save as ambiguity_v<>.xlsm to run macros.

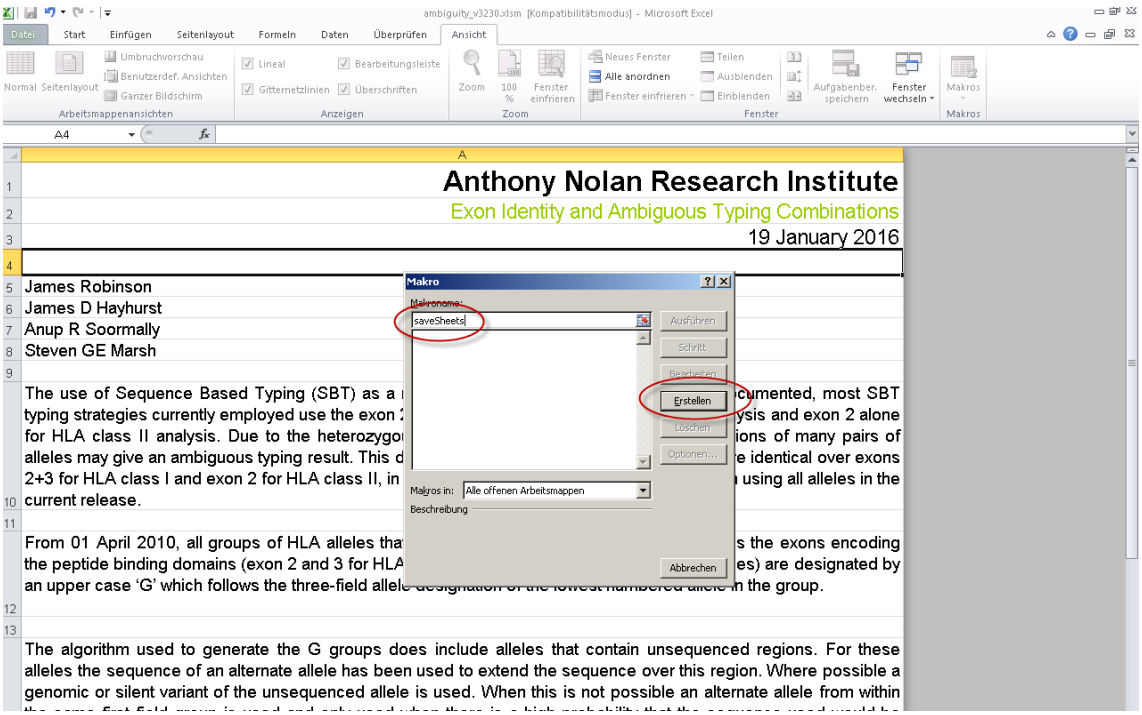


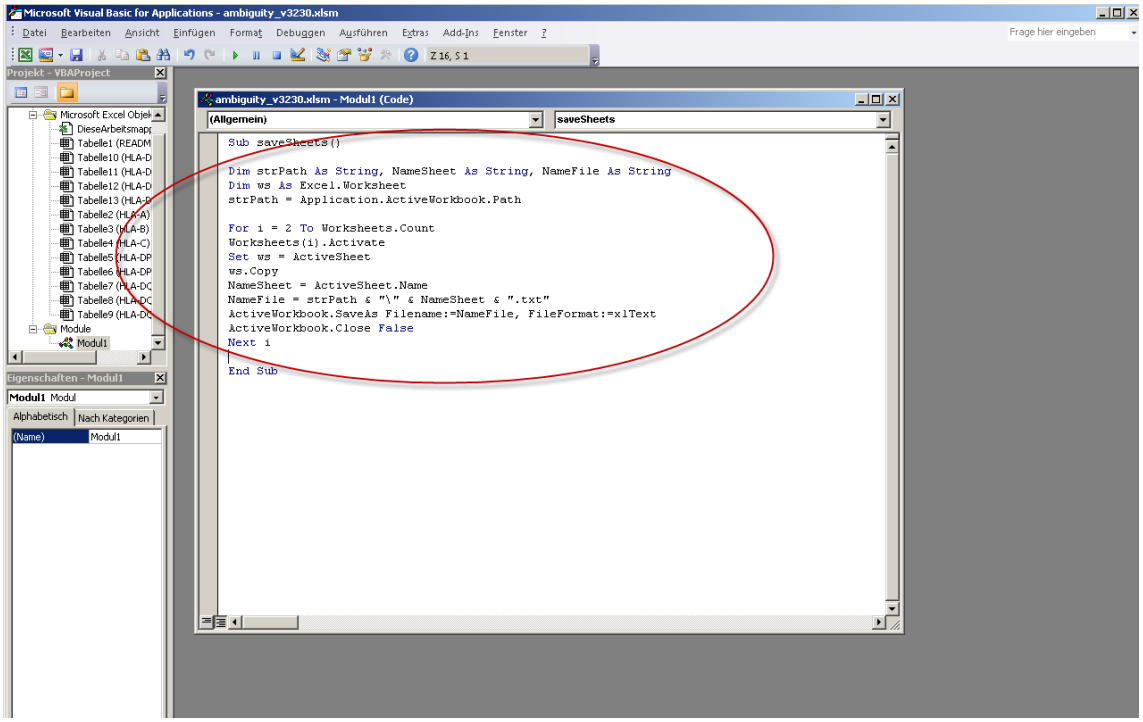



Now insert the following macro, which saves relevant information from the Excelsheets as text files:

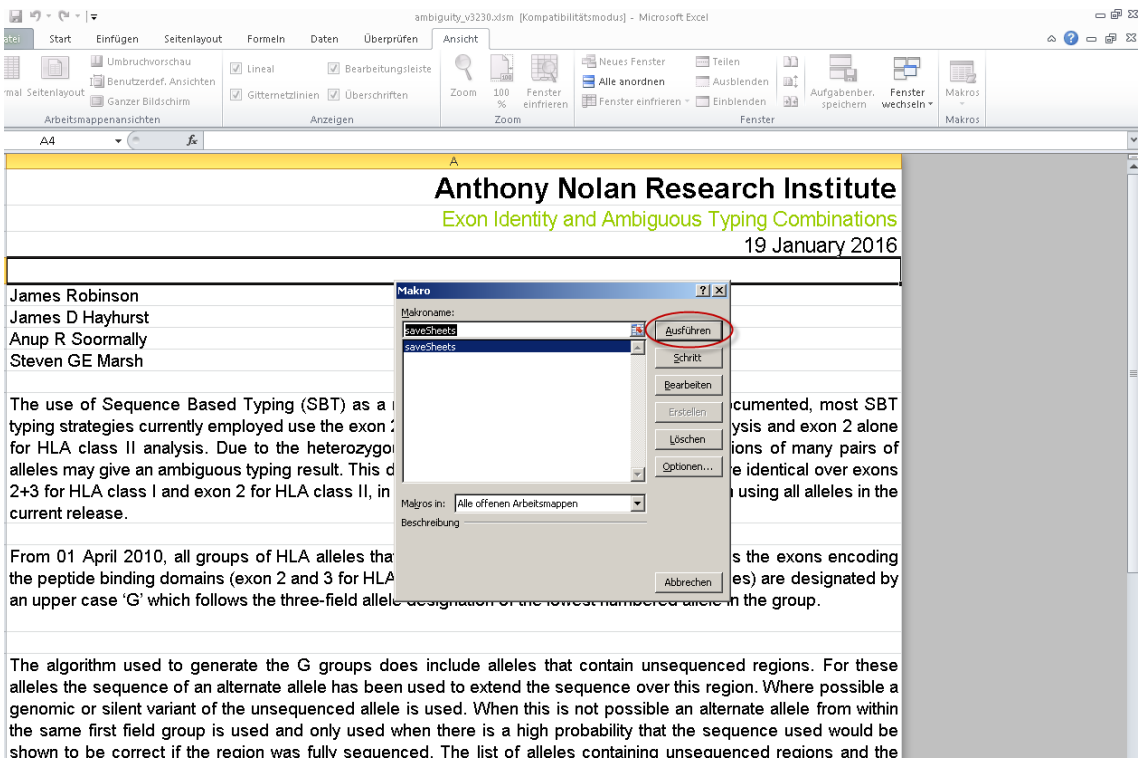
```
Sub saveSheets()
Dim strPath As String, NameSheet As String, NameFile As String
Dim ws As Excel.Worksheet
strPath = Application.ActiveWorkbook.Path

For i = 2 To Worksheets.Count
Worksheets(i).Activate
Set ws = ActiveSheet
ws.Copy
NameSheet = ActiveSheet.name
NameFile = strPath & "\" & NameSheet & ".txt"
ActiveWorkbook.SaveAs Filename:=NameFile, FileFormat:=xlText
ActiveWorkbook.Close False
Next i
End Sub
```


ambiguity_v3230.xlsm [Kompatibilitätsmodus] - Microsoft Excel
Datei
Start
Einfügen
Seitenlayout
Formeln
Daten
Überprüfen
Ansicht
Makros
Makros anzeigen
Makro aufzeichn.
Relative Aufzeichnung
A4
Anthony Nolan Research Institute
Exon Identity and Ambiguous Typing Combinations
19 January 2016
James Robinson
James D Hayhurst
Anup R Soormally
Steven GE Marsh
The use of Sequence Based Typing (SBT) as a method for defining the HLA type is well documented, most SBT typing strategies currently employed use the exon 2 and exon 3 sequences for HLA class I analysis and exon 2 alone for HLA class II analysis. Due to the heterozygous nature of the SBT analysis the combinations of many pairs of


ambiguity_v3230.xlsm [Kompatibilitätsmodus] - Microsoft Excel
Anthony Nolan Research Institute
Exon Identity and Ambiguous Typing Combinations
19 January 2016
James Robinson
James D Hayhurst
Anup R Soormally
Steven GE Marsh
Makro
Makroname:
saveSheets
Ausführen
Schritt...
Bearbeiten
Erstellen
Löschen
Optionen...
Makros in:
Alle offenen Arbeitsmappen
Beschreibung
Abbrechen




Close the window and execute the macro:




Some new text files should have appeared in your folder „inputData“. Afterwards you can remove the Excel file.

## Build Data for Hapl-o-Mat

Enter the folder InputData and copy all files to the folder “prepareData”. Then enter folder “prepareData”. Next, create the data required for Hapl-o-Mat by running the python scripts in the following order (you can easily get Python here https://www.python.org/downloads/windows/):

TransferAlphaToMultipleAlleleCodes.py
BuildAllAllelesFrom_hla_nom_g.py
BuildAllAllelesExpanded.py
BuildP.py
BuildLargeG.py
BuildSmallg.py
BuildAmbiguity.py
AddGToAmbiguity.py
AddAllelesMissingIngCode.py

Then create the folder Hapl-o-Mat/data and copy the files

AllAllelesExpanded.txt
Ambiguity.txt
LargeG.txt
MultipleAlleleCodes.txt
P.txt
Smallg.txt

to the folder data.

## Input Genotype Data

Hapl-o-Mat infers haplotypes from population genotype data. It supports different formats of recording genotype data. To use Hapl-o-Mat your data should be in one of the following data formats:

| **Data format** | **Description** |
|---|---|
| MA | Ambiguities are encoded by multiple allele (MA) codes. Except for the first line, input files hold an individual's identification number and genotype per line. Genotypes are saved allele by allele without locus name. Identification number and alleles are TAB-separated. The first line of the file is a header file indicating the name of the first column and the loci of the other columns. Same loci must be placed next to each other. For an example refer to "examplePopulations/populationData_a.dat". |
| GLC | Genotypes with or without ambiguities are saved by genotype list strings. Input files hold an individual's identification number and genotype per line. Identification number and single-locus genotypes are TAB-separated. For an example refer to "examplePopulations/populationData_b.dat" |
| GL | Genotypes with or without ambiguities are saved by genotype list (GL) strings. Population data is saved in two files. The pull-file contains an individual's identification number and a list of integer numbers, GL-ids, referring to its single-locus genotype. The GL-ids are separated from the identification number via ";" and from each other via ":". The second file, the glid-file, contains a translation from GL-ids starting with "1" to actual single-locus genotypes. GL-id and genotype are separated via ";". A GL-id of "0" is interpreted as a missing typing at the corresponding locus and does not require a translation in the glid-file. For an example refer to "examplePopulations/populationData_c.pull" and "examplePopulations/populationData_c.glid". |

| | |
|---|---|
| READ | Ambiguities are completely resolved and alleles are already translated to the wanted typing resolutions. The input data is of the format as Hapl-o-Mat records processed genotype data. This allows for easily repeating a run without the need to resolve genotype data again. |

When compiling Hapl-o-Mat we assigned MA as argument to the executable. If you want to run another format, you have to change the argument to the appropriate format abbreviation. Right-click in the Project Explorer and select Run As -> Run Configurations…. Then, click on the tab Arguments and change MA to some abbreviation from above.

# Parameters

Each input format for genotype data requires a different set of parameters. The parameters are saved in the corresponding files "parametersMA", "parametersGLC", "parametersGL", and "parametersREAD". All input formats have the following parameters in common:

| **Parameter** | **Description** |
|---|---|
| FILENAME_HAPLOTYPES | Name of the file which temporarily saves haplotype names |
| FILENAME_GENOTYPES | Name of the file which saves resolved genotypes. |
| FILENAME_HAPLOTYPEFREQUENCIES | Name of the file which saves haplotypes and estimated haplotype frequencies. |
| FILENAME_EPSILON_LOGL | Name of the file which saves stopping criterion and log-likelihood per iteration. |
| INITIALIZATION_HAPLOTYPEFREQUENCIES | Initialization routine for haplotype frequencies. It takes the following values:<br>• "equal": All haplotype frequencies are initialized with the same frequency<br>• "numberOccurrence": Haplotype frequencies are initialized according to the initial number of occurrence of the haplotype<br>• "random": Haplotype frequencies are initialized randomly<br>• "perturbation": Haplotype frequencies are initialized as in numberOccurrence and then randomly modified by a small (<10%) positive or negative offset |
| EPSILON | Value for the stopping criterion, i.e. the maximal change between consecutive haplotype frequency estimations is smaller than the assigned value. |
| CUT_HAPLOTYPEFREQUENCIES | Estimated haplotype frequencies smaller than this value are removed from the output |
| RENORMALIZE_HAPLOTYPEFREQUENCIES | Takes values "true" and "false". If "true", normalize estimated haplotype frequencies to sum to one. Within machine precision, this becomes necessary, if estimated haplotypes are removed, e.g. via the option CUT_HAPLOTYPEFREQUENCIES |

| | |
|---|---|
| SEED | Set the seed of the used pseudo random number generator. If set to "0", the seed is initialized by the system time. |

Depending on the input format (indicated in brackets), additional parameters are:

| **Parameter** | **Input format** | **Description** |
|---|---|---|
| FILENAME_INPUT | MA, GLC, READ | The file name of the input population data |
| FILENAME_PULL | GL | The file name of the pull-file |
| FILENAME_GLID | GL | The file name of the glid-file |
| LOCI_AND_ RESOLUTIONS | MA, GL, GLC | Loci included into analysis and desired typing resolution per locus. The list is separated by "," and contains the locus name followed by ":" and the desired typing resolution, e.g. A:g,B:4d,C:g. Supported typing resolutions and their abbreviations are g-groups (g), P-groups (P), G-groups (G), 2-digit fields (2d), 4-digit fields (4d), 6-digit fields (6d), and 8-digit fields (8d). Alleles are not translated via the option asItIs (applying the ambiguity filter includes an intrinsic translation to G-groups) |
| LOCIORDER | GL | Specify the order of loci the individual's GL-ids correspond to. Loci are separated via ",". |
| RESOLVE_MISSING_ GENOTYPES | GL | Takes values "true" and "false". If set to true, a missing typing is replaced by a combination of all alleles from AlleleList.txt at the locus. Else, individuals with a missing typing are discarded from analysis |
| MINIMAL_FREQUENCY_ GENOTYPES | MA, GL, GLC | Genotypes which split into more genotypes than the inverse of this number are discarded from analysis |
| DO_AMBIGUITYFILTER | MA, GL, GLC | Takes values "true" and "false". The option "true" activates the ambiguity filter |
| EXPAND_LINES_ AMBIGUITYFILTER | MA, GL, GLC | Takes values "true" and "false". If set to "true", matching lines with additional genotype pairs in the ambiguity filter are considered |

Whenever specifying a file name including folders, you have to create the folders before running Hapl-o-Mat.

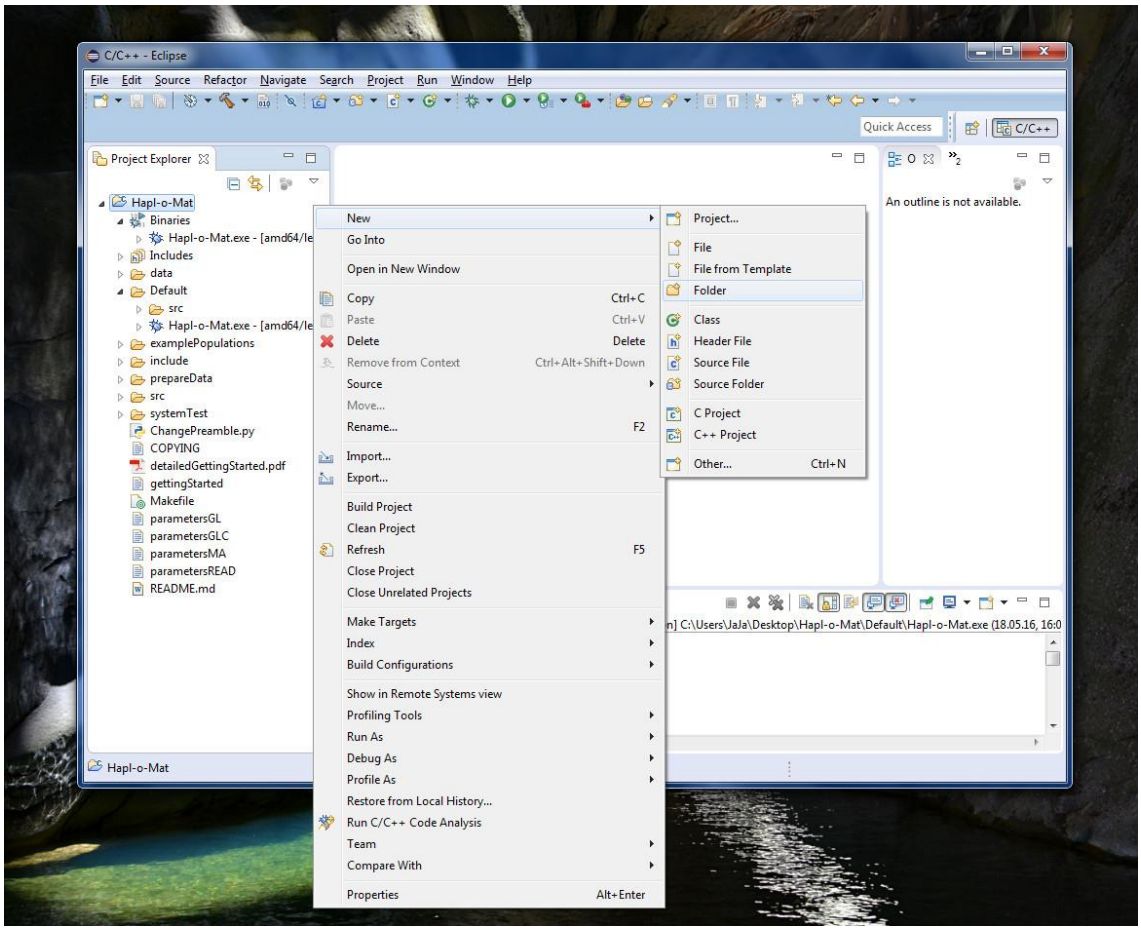
# Tutorials

We have everything ready to use Hapl-o-Mat. In the following we estimate haplotype frequencies from some included genotype data recorded in the input format MA.

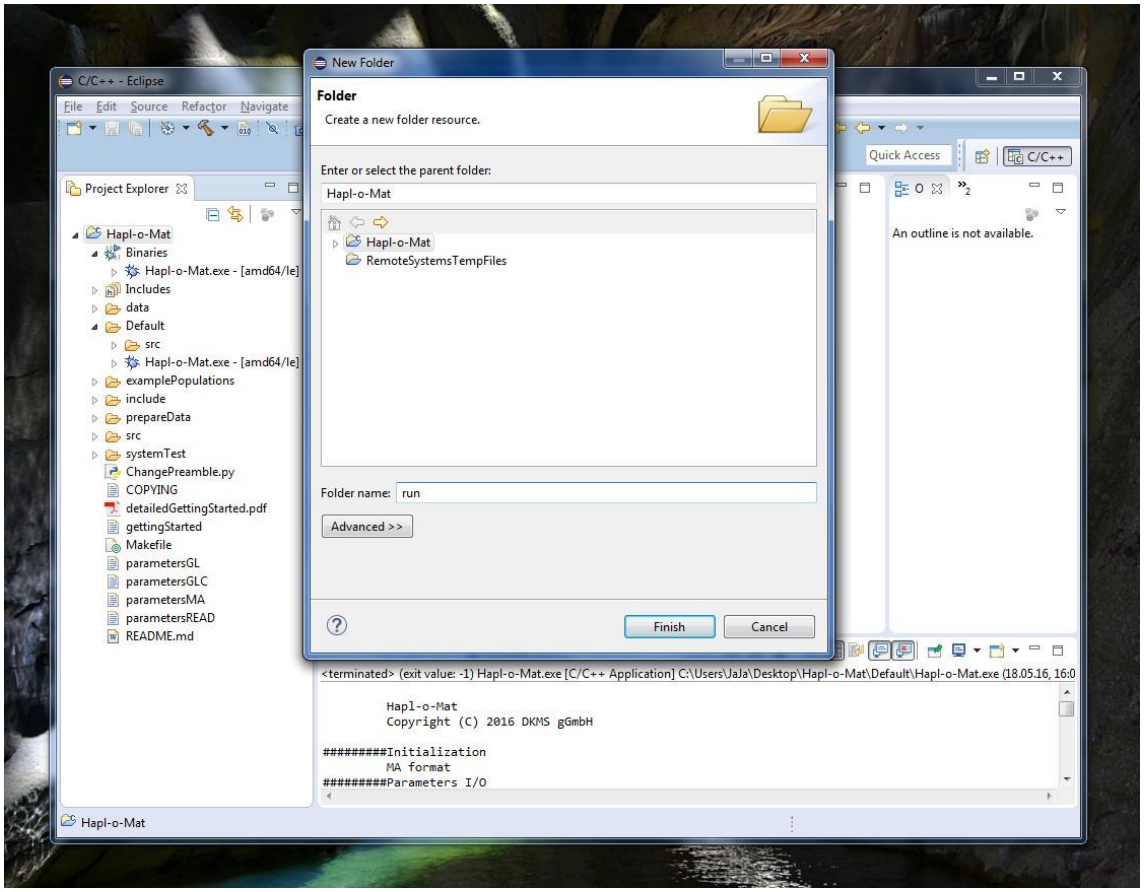
## Input Format MA

You find the relevant population data in "examplePopulations/populationData_a.dat". As ambiguities are recorded as multiple allele codes, the input format is MA. We are going to infer three locus (A, B, DRB1) haplotypes from this data. Alleles at loci A and B shall be translated to typing resolution g and alleles at locus DRB1 to 4-digits typing resolution.

Create a folder by right-clicking in the Project Explorer and choosing New -> Folder.

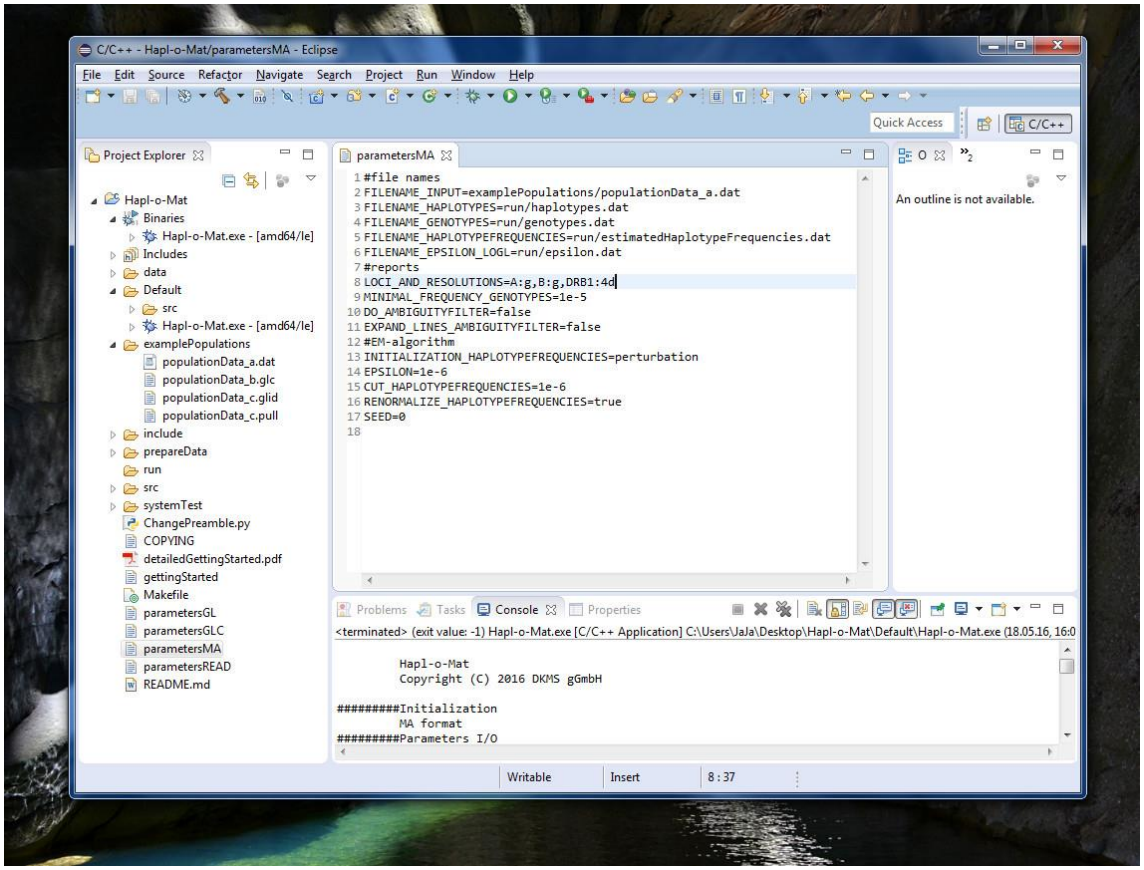



Name the folder run and then click Finish.

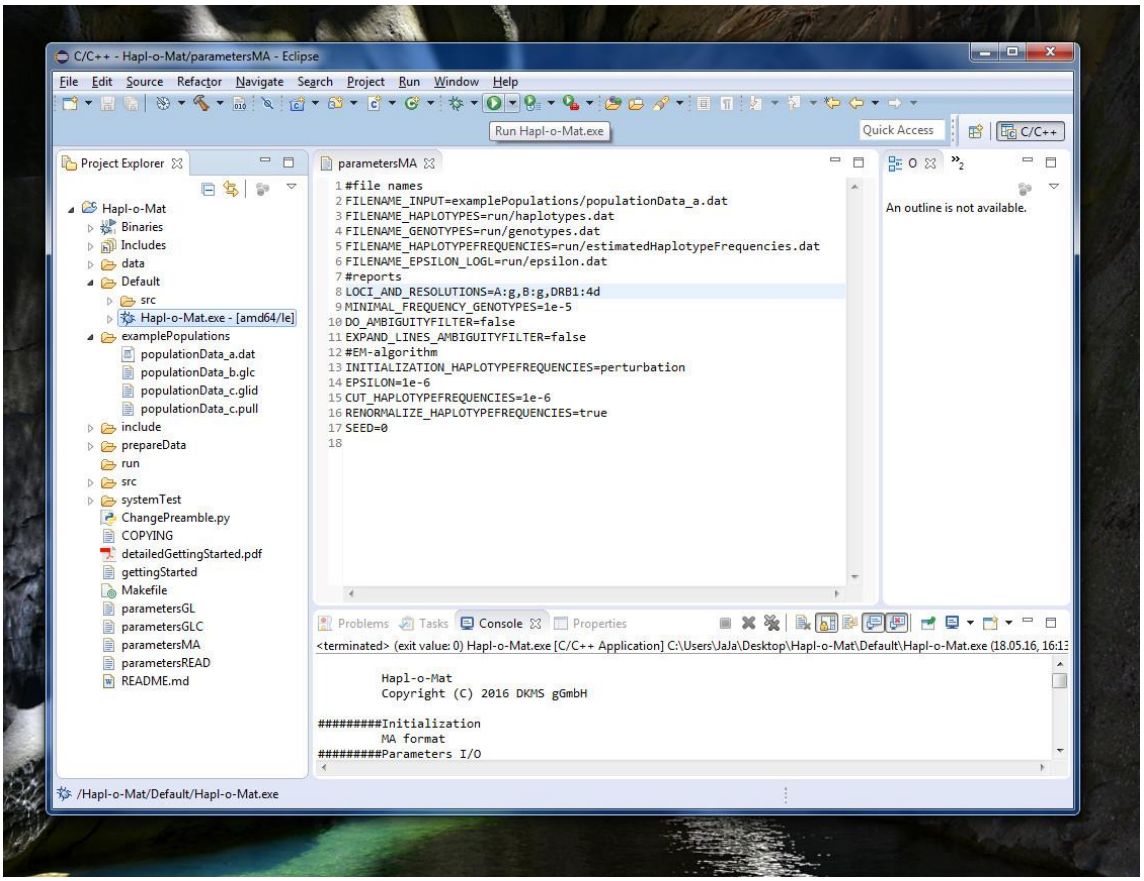



Next, open the parameter file parametersMA by double-clicking on it. Change its parameters according to:

#file names
FILENAME_INPUT=../populationData_a.dat
FILENAME_HAPLOTYPES=run/haplotypes.dat
FILENAME_GENOTYPES=run/genotypes.dat
FILENAME_HAPLOTYPEFREQUENCIES=run/hfs.dat
FILENAME_EPSILON_LOGL=run/epsilon.dat
#reports
LOCI_AND_RESOLUTIONS=A:g,B:g,DRB1:4d
MINIMAL_FREQUENCY_GENOTYPES=1e-5
DO_AMBIGUITYFILTER=false
EXPAND_LINES_AMBIGUITYFILTER=false
#EM-algorithm
INITIALIZATION_HAPLOTYPEFREQUENCIES=perturbation
EPSILON=1e-6
CUT_HAPLOTYPEFREQUENCIES=1e-6
RENORMALIZE_HAPLOTYPEFREQUENCIES=true
SEED=1000

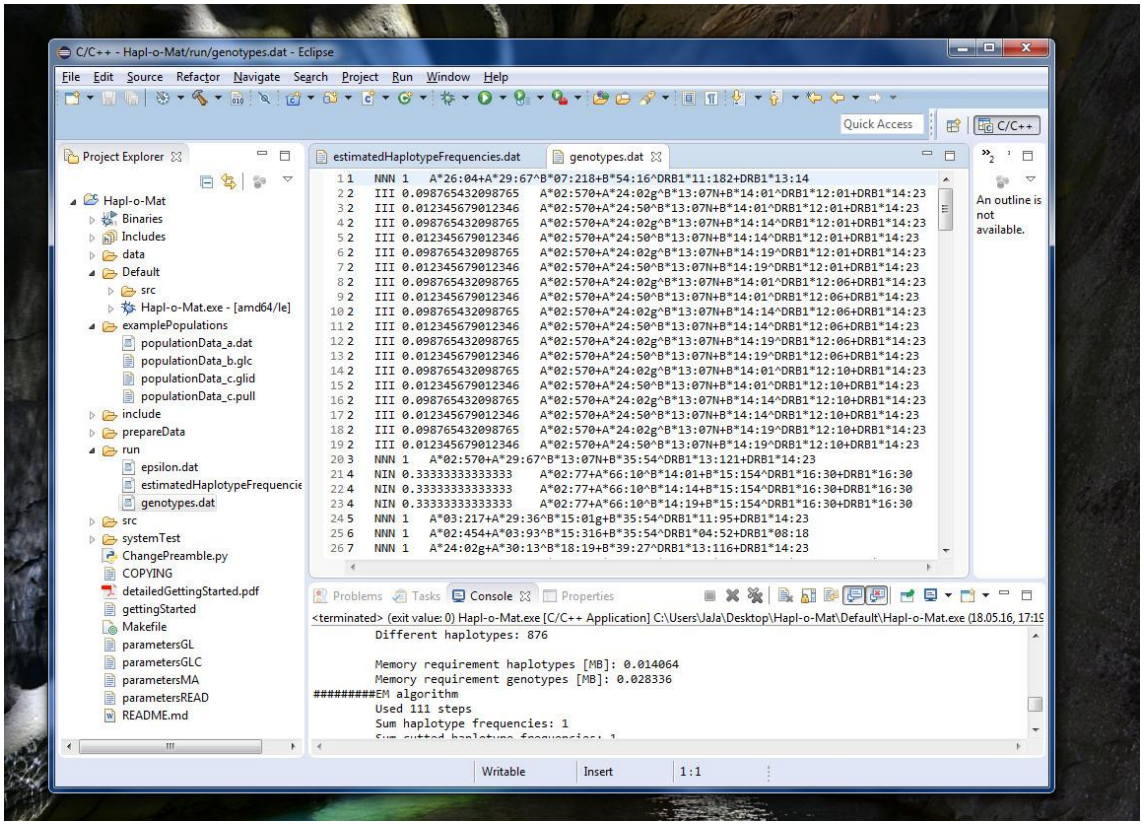



Compute haplotype frequencies from the genotype input data by running Hapl-o-Mat. To this end, click on the green Play button in the upper navigation bar.



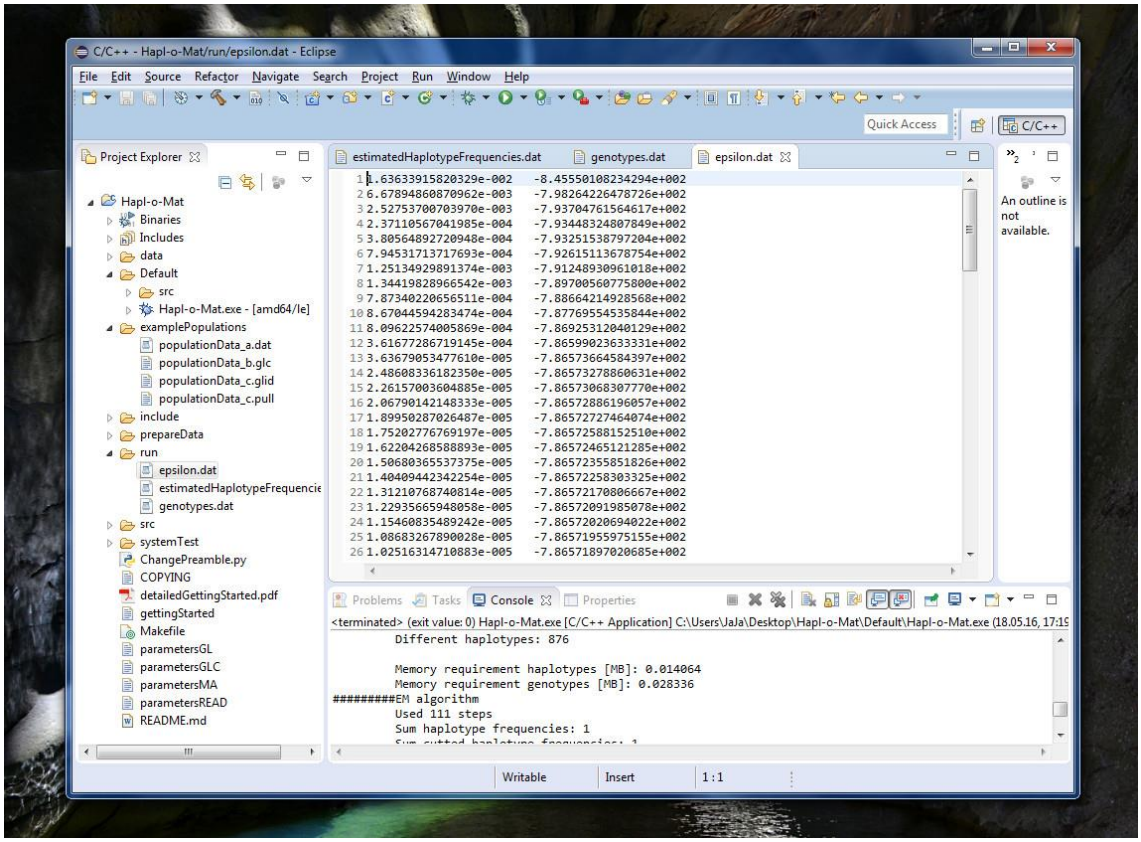
It produces some output on the screen including your chosen parameters, statistics on the resolved genotype data and the expectation-maximization algorithm, and the run time.

Now let's examine the results produced by Hapl-o-Mat. The results are saved in the folder run (press F5 to update, if they do not appear). We first look into the file with the resolved genotypes, "run/genotypes.dat".




The first column corresponds to the individual's identification number. The second column indicates how ambiguities per single-locus genotypes have been resolved. If no ambiguity occurred or no additional genotypes are formed, the type is N. If an ambiguity occurred and was resolved via building all possible allele combinations, the type is I. Activating the ambiguity filter gives additional types: A, if one matching line in the ambiguity file was found, and M if multiple matching lines were found. The third column gives the frequency of the genotype and the fourth column the genotype itself. The genotype is saved in the GL format. If an individual's genotype splits into a set of genotypes, each genotype is written to one line starting with the same identification number. The corresponding frequencies become non-integer and sum to one.

The evolution of the stopping criterion and log-likelihood while iterating expectation and maximization steps is written to "run/epsilon.dat". The first column is the stopping criterion and the second one the not normalized log-likelihood.




The inferred haplotypes including estimated frequencies are listed in "run/hfs.dat". Haplotypes are saved in the GL format. This is the file you were aiming at. It is sorted by descending frequency and already normalized if you activated the corresponding option (we did in this tutorial).