PGDSpider

PGDSpider version 2.0.1.0 (October 2011)

An automated data conversion tool for connecting population genetics and genomics programs

Author: Heidi Lischer

Computational and Molecular Population Genetics lab (CMPG)
Institute of Ecology and Evolution (IEE)
University of Berne
3012 Bern
Switzerland

Member of the Swiss Institute of Bioinformatics (SIB)

e-mail: heidi.lischer@iee.unibe.ch

Contents

1 Introduction

PGDSpider is a powerful automated data conversion tool for population genetic and genomics programs. It facilitates the data exchange possibilities between programs (Fig. 1) for a vast range of data types (e.g. DNA, RNA, NGS, microsatellite, SNP, RFLP, AFLP, multi-allelic data, allele frequency or genetic distances). Besides the conventional population genetics formats, PGDSpider integrates population genomics data formats commonly used to store and handle next-generation sequencing (NGS) data. Currently, PGDSpider is not meant to convert very large NGS files as it loads into memory the whole input file, whose size may exceed available RAM. However, since PGDSpider allows one to convert specific subsets of these NGS files into any other format, one could use this feature to calculate parameters or statistics for specific regions, and thus perform sliding window analysis over large genomic regions.

PGDSpider uses a newly developed PGD (Population Genetics Data) format as an intermediate step in the conversion process. PGD is a file format designed to store various kinds of population genetics data, including different data types (e.g. DNA sequences, microsatellites, AFLP or SNPs) and ploidy levels. PGD is based on the XML format and is therefore independent of any particular computer system and extensible for future needs. PGDSpider uses PGD to connect population genetics and genomics programs like a spider knits a web.

PGDSpider is written in Java and is therefore platform independent. It is user friendly due to its intuitive graphical user interface. PGDSpider allows the user to store his preferred conversion settings for repeated conversions of similar input formats. A command line version of PGDSpider is also provided, making it possible to embed PGDSpider in data analysis pipelines.

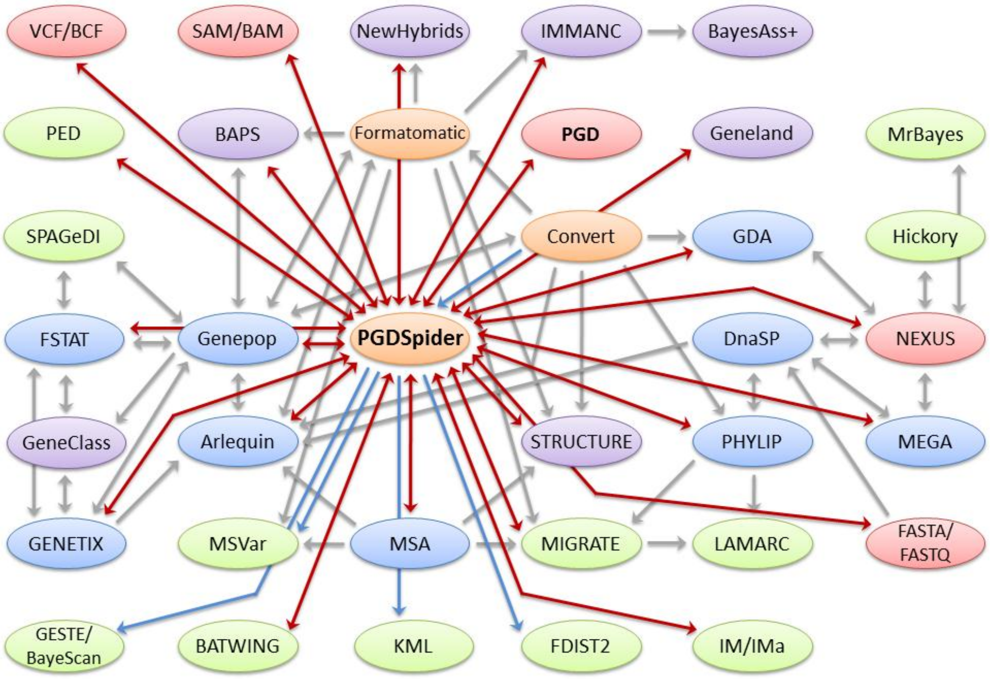


Fig. 1: Connectivity between population genetics programs and format. Red (reading and writing) and blue (reading or writing) arrows indicate direct connections between PGDSpider and other programs. Grey arrows show connections between the programs themselves that are not mediated by PGDSpider. Blue ellipses represent multi-purpose generalist packages and violet ellipses show individual-centred programs. Conversion programs are shown in orange, specialized programs in green and general data formats in red.

2 Formats supported by PGDSpider

PGDSpider is able to parse 27 and to write 29 different file formats:

Data format	Version	References and Links	Input format	Output format
PGD	1.0		x	x
Arlequin	3.5 (24.2.2010)	http://cmpg.unibe.ch/software/arlequin35/, http://cmpg.unibe.ch/software/arlequin35/man/arlequin35.pdf, (Excoffier and Lischer, 2010)	x	x
BAM	(17.4.2011)	http://samtools.sourceforge.net, http://samtools.sourceforge.net/SAM1.pdf, (Li, et al., 2009)	x	x
BAPS	5.4 (29.4. 2010)	http://web.abo.fi/fak/mnf//mate/jc/software/baps.html, http://web.abo.fi/fak/mnf//mate/jc/software/BAPS5manual.pdf, (Tang, et al., 2009)	x	x
BATWING	(2003)	http://www.mas.ncl.ac.uk/~nijw/, http://www.mas.ncl.ac.uk/~nijw/batwing/batguide.pdf, (Wilson, et al., 2003)	x	x
BCF	(14.5.2011)	http://samtools.sourceforge.net/mpileup.shtml	x	x
CONVERT	1.31 (March 2005)	http://www.agriculture.purdue.edu/fnr/html/faculty/Rhodes/Students%20and%20Staff/glaubitz/software.htm, (Glaubitz, 2004)	x	
FASTA		http://en.wikipedia.org/wiki/FASTA_format, http://www.ncbi.nlm.nih.gov/blast/fasta.shtml (Pearson, 1990)	x	x
FASTQ		http://en.wikipedia.org/wiki/FASTQ_format, (Cock, et al., 2010)	x	x
FDist2 (datacal)		http://www.rubic.rdg.ac.uk/~mab/software.html, (Beaumont and Nichols, 1996; Flint, et al., 1999)		x
FSTAT	2.9.3.2 (February 2002)	http://www2.unil.ch/popgen/softwares/fstat.htm, (Goudet, 2001)	x	x
GDA	1.1 (7.1.2002)	http://hydrodictyon.eeb.uconn.edu/people/plewis/software.php, (Lewis, 2001)	x	x
GENELAND	12.4.2011	http://www2.imm.dtu.dk/~gigu/Geneland/ http://www2.imm.dtu.dk/~gigu/Geneland/Geneland-Doc.pdf (Guillot, 2008; Guillot, et al., 2005; Guillot, et al., 2005; Guillot and Santos, 2009; Guillot and Santos, 2010; Guillot, et al., 2008)	x	x
GENEPOP	4.1 (24.3.2011)	http://kimura.univ-montp2.fr/~rousset/Genepop.htm, http://kimura.univ-montp2.fr/~rousset/Genepop.pdf, http://genepop.curtin.edu.au/help_input.html, (Rousset, 2008)	x	x

GENETIX	4.05 (5.5.2004)	http://www.genetix.univ-montp2.fr/genetix/genetix.htm, (Belkhir, 1996-2004)	x	x
GESTE/ BayeScan	GESTE: 2.0/ BayeScan 2.01 (December 2010)	http://www-leca.ujf-grenoble.fr/logiciels.htm http://cmpg.unibe.ch/software/bayescan/index.html GESTE: (Foll and Gaggiotti, 2006) BayeScan: (Fischer, et al., 2011; Foll, et al., 2010; Foll and Gaggiotti, 2008)		x
HGDP-CEPH (Arlequin + log file)	3.0	http://www.cephb.fr/en/hgdp/	x	
Immanc (BayesAss)	5.0 (8.10.1998)	http://www.rannala.org/?page_id=13, http://www.rannala.org/docs/immanc.html, (Rannala and Mountain, 1997) (Wilson and Rannala, 2003)	x	x
IM (IMa)	Updated 17.12.2009	http://genfaculty.rutgers.edu/hey/software, http://lifesci.rutgers.edu/~heylab/ProgramsandData/Programs/IM/Introduction_to_IM_and_IMa_3_5_2007.pdf, (Hey and Nielsen, 2004; Nielsen and Wakeley, 2001), (Hey and Nielsen, 2007)	x	x
KML	2.2	http://code.google.com/intl/de-CH/apis/kml/documentation/kml_tut.html		x
MEGA	5 (26.4.2011)	http://www.megasoftware.net/, http://www.megasoftware.net/manual.pdf, (Tamura, et al., 2011)	x	x
MIGRATE	3.2.6 (13.10.2010)	http://popgen.sc.fsu.edu/Migrate/Migrate-n.html, http://popgen.sc.fsu.edu/migratedoc.pdf, (Beerli, 2009)	x	x
MSA	4.05	http://i122server.vu-wien.ac.at/MSA/info.html/MSA_info.html, (Dieringer and Schlotterer, 2003)	x	x
MSVar	0.4.1.b (7.4.1999)	http://www.rubic.rdg.ac.uk/~mab/software.html, (Beaumont, 1999)		x
NewHybrids	1.1 beta (7.4.2003)	http://ib.berkeley.edu/labs/slatkin/eriq/software/software.htm, http://ib.berkeley.edu/labs/slatkin/eriq/software/new_hybs_doc1_1Beta3.pdf (Anderson and Thompson, 2002)	x	x
NEXUS (DNA/ RNA/ NUCLEOTIDE)		(Maddison, et al., 1997)	x	x
PED		http://pngu.mgh.harvard.edu/~purcell/plink/data.shtml#ped (Anderson and Thompson, 2002)	x	x
PHYLIP (DNA/ RNA /distance matrix)	3.69 (September 2009)	http://evolution.genetics.washington.edu/phylip/doc/main.html, (Felsenstein, 1989; Felsenstein, 2004)	x	x

SAM	1.4 (17.4.2011)	http://samtools.sourceforge.net, http://samtools.sourceforge.net/SAM1.pdf, (Li, et al., 2009)	x	x
STRUCTURE	2.3.3 (January 2010)	http://pritch.bsd.uchicago.edu/structure.html, http://pritch.bsd.uchicago.edu/structure_software/release_versions/v2.3.3/structure_doc.pdf, (Falush, et al., 2003; Falush, et al., 2007; Hubisz, et al., 2009; Pritchard, et al., 2000)	x	x
VCF	4.4 (14.5.2011)	http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-40	x	x

Tab. 1: Data formats supported by PGDSpider including the version number, references and links to webpages and format descriptions, and if the format is supported as input and/or output format.

3 How to cite PGDSpider and License

Lischer HEL and Excoffier L (submitted) PGDSpider: An automated data conversion tool for connecting population genetics and genomics programs

PGDSpider is distributed under the BSD 3-Clause License. For the full text of the license, see the file LICENSE.txt.

By using, modifying or distributing this software you agree to be bound by the terms of this license.

4 System requirements

PGDSpider is written in Java and therefore platform independent, but SUN Java 1.6 RE (or a newer version) has to be installed. Java6 RE can be downloaded under following link: http://www.oracle.com/technetwork/java/javase/downloads/index.html

5 Installing PGDSpider

All necessary links and an installation instruction are also available on http://cmpg.unibe.ch/software/PGDSpider/ or in the readme file.

5.1 Installation Instructions

1st step:

Install the Java6 RE

- Windows:
  download and install Java6 RE with following link:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Linux:
  - Ubuntu / Debian:
    Execute the following command as root user:
    `"apt-get install openjdk-6-jre"`
  - Other Linux distributions:
    http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Mac:
  Apple Computer supplies their own version of Java. Use the Software Update feature (available on the Apple menu) to check that you have the most up-to-date version of Java for your Mac.
  If you have problems with downloading, installing or using Java on Mac, please contact Apple Computer Technical Support.

2nd step:

Download the PGDSpider application from http://cmpg.unibe.ch/software/PGDSpider/ and unzip it on the local drive.

- Execute PGDSpider GUI:
  - Windows: execute the file `"PGDSpider2.exe"` to start the program.
  - Linux: execute the command `"./PGDSpider2.sh"` to start the program.
  - Mac and others: execute the command
    `"java -Xmx1024m -Xms512m -jar PGDSpider2.jar"`
    to start the program.

- Execute PGDSpider-cli (command line)
    - Windows: execute the command `"PGDSpider2-cli.exe"`
    - Linux: execute the command `"java -Xmx1024m -Xms512M -jar PGDSpider2-cli.jar"`
    - Mac and others: execute the command `"java -Xmx1024m -Xms512M -jar PGDSpider2-cli.jar"`

5.2 Java Web Start

Additionally we provide the possibility to download and run PGDSpider from the web by the Java Web Start software. Java Web Start provides an easy, one-click activation of PGDSpider and it guarantees that you are always running the latest version.

5.2.1 Launch PGDSpider

Java Web Start is included in the Java Runtime Environment. Have a look at the 1st step of the 5.1 Installation Instructions to get information on how to get Java6 RE (or a newer version).

Launch PGDSpider using Java Web Start from

- Browser:
  Click on the PGDSpider icon on the web page:

- Java Cache Viewer:
  To launch the PGDSpider Web Start a second time, you do not need to return to the web page where you first launched it; instead you can launch it from the Java Cache Viewer. To open the Java Cache Viewer execute following command in the console: `javaws -viewer`
  To run PGDSpider Web Start, select it and click the Run button or double click the PGDSpider application.

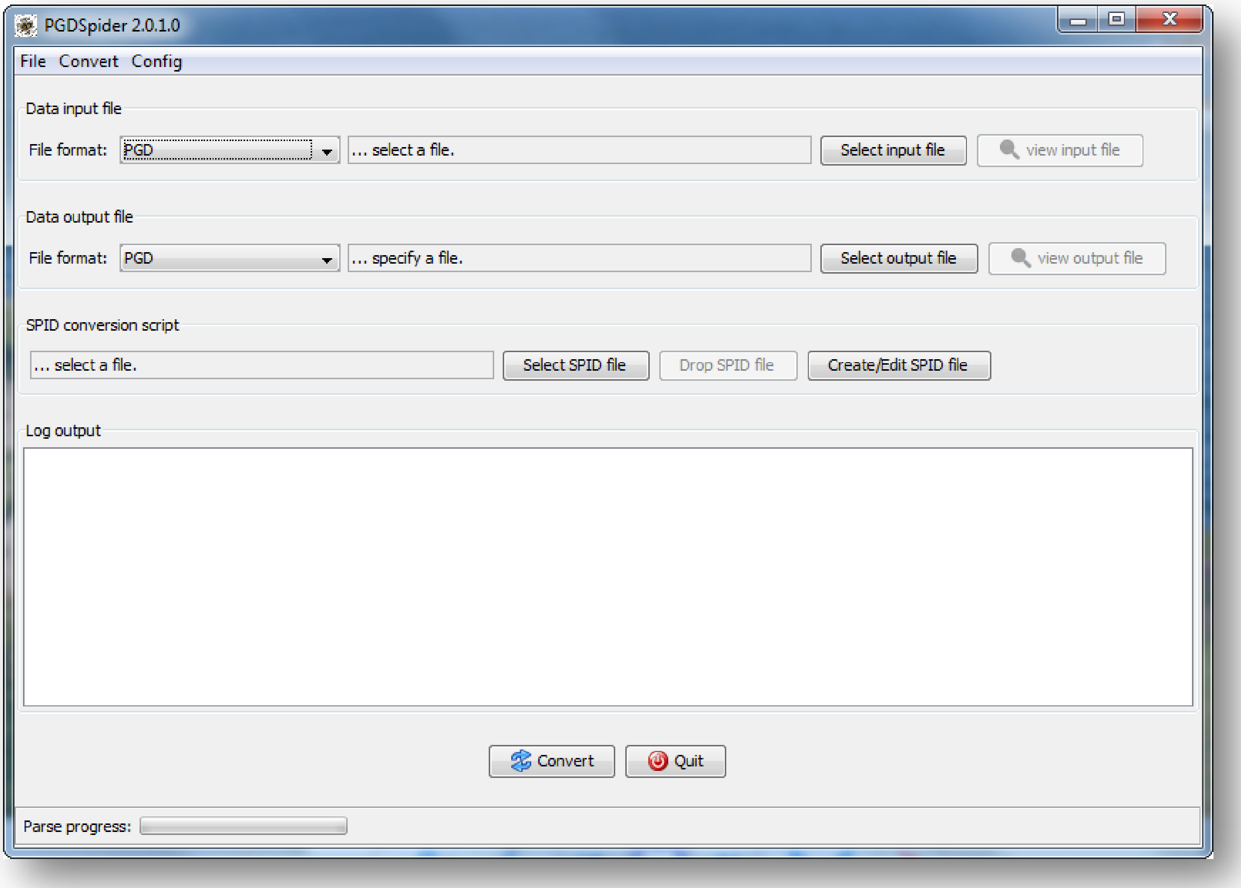
- Desktop:
  You can add a desktop shortcut to the PGDSpider Web Start application. Select the application in the Java Cache Viewer (see above how to open it), then right-click and select “Install Shortcuts” or click the Install button. A shortcut is added to the desktop and you can launch the PGDSpider Web Start application just as you would launch any native application.

5.2.2 Limitations

Starting PGDSpider from Java Web Start it is not possible to change the amount of memory PGDSpider is allowed to use (by default it is set to 1 GB). If you need to change the amount of memory (e.g.: if you have large files to convert), download the PGDSpider application as described in the 2nd step of the 5.1 Installation Instructions.

6 Execute PGDSpider GUI

The graphical user interface of the PGDSpider program is available in four different languages (English, French, German and Italian) and looks like:



File format: PGD

... select a file.

File format: PGD

... specify a file.

... select a file.

Fig. 2: Screenshot of the English version of the graphical user interface of PGDSpider

Execute PGDSpider GUI:

- Windows: execute the file "`PGDSpider2.exe`" to start the program.
- Linux: execute the command "`./PGDSpider2.sh`" to start the program.
- Mac and others: execute the command
  "`java -Xmx1024m -Xms512M -jar PGDSpider2.jar`"
  to start the program.

Increase memory:
To increase the memory PGDSpider is allowed to use start the program by executing the command "`java -Xmx1024m -Xms512M -jar PGDSpider2.jar`" and adapt the `–Xmx` parameter to your needs (`–Xmx1024m` means: maximum memory of 1’024 MB).

6.1 How to use the PGDSpider GUI

- Select the input file format to be translated:
  First the format of the input file has to be selected. Use the putt down menu (to the left of the text “File format” in the Input File area) to select the input file format.

- Select the input file:
  Click on the button “Select input file” and choose the file to be translated with the specified file format. Note that PGDSpider does not check if the selected file is of the right format.

- View the input file:
  To have a look at the selected input file click on the “view file” button in the Input File area

- Select the output format:
  Choose the desired file format of the output file. To do this select the output file format in the drop down menu to the left of the text “File format” in the Output File area

- Select the output file:
  Click on the button “Select output file” and choose the place where the output file should be saved.

- Select a SPID file:
  Click on the button “Select SPID file” to select a SPID file which contains the answers for the Parser and the Writer Questions.

- Drop SPID file:
  Click on the button “Drop SPID file” to remove the selected SPID file.

- Create/Edit SPID file:
  Click on the button “Create/Edit SPID file” to open a window with the SPID editor to create or edit a SPID file.

- Convert file format:
  To convert the specified input file to the desired output format, press the “Convert” button. If no SPID file was selected a window appear with Parser and Writer questions (SPID editor). When the questions are answered, the user has the possibility to save the answers in a SPID file. Afterwards the answers are applied in the conversion process. A progress bar at the bottom of the graphical user interface shows the progress of the parsing action.
  After conversion, the user should control the output file (mistakes in the input file could lead to mistakes in the output file)!

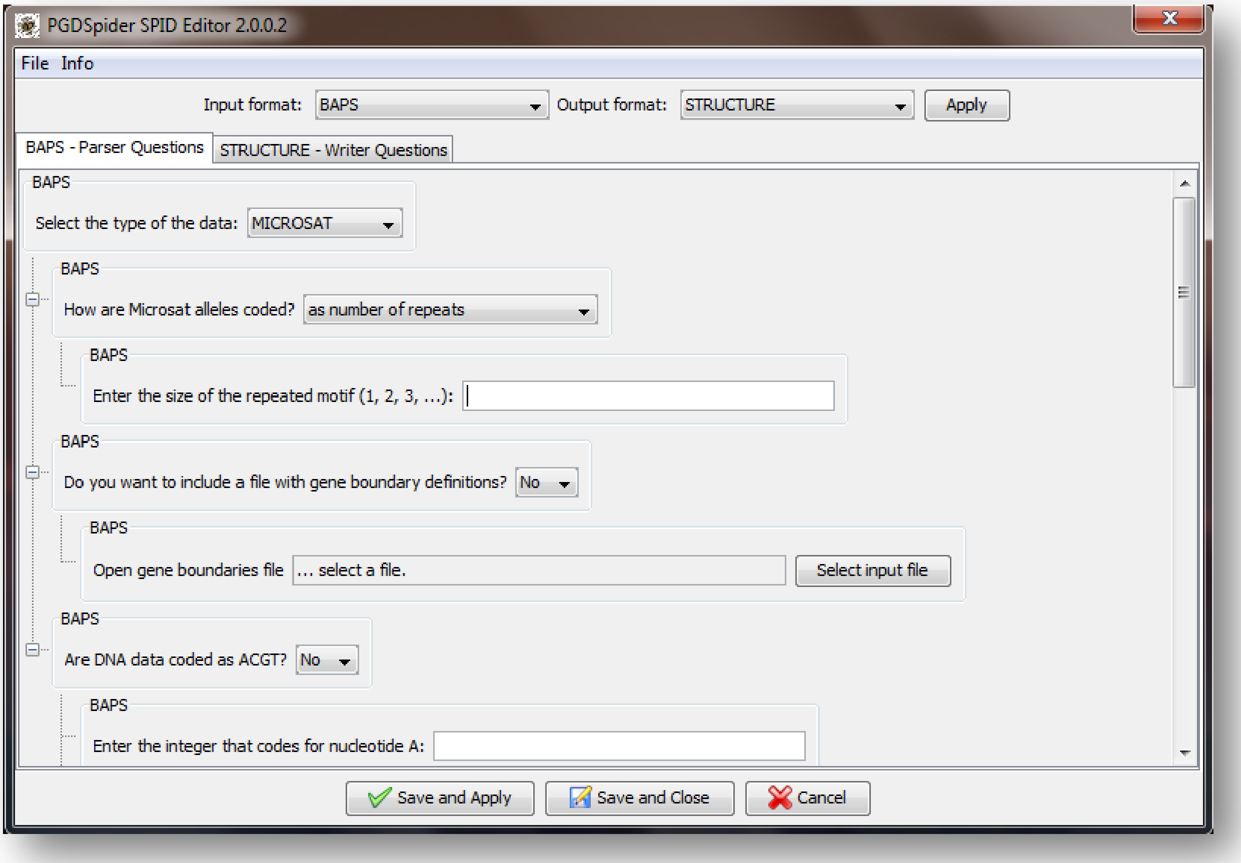
- View the output file:
  If conversion is over, one can have a look at the generated output file if the button “view file” in the Output File area is clicked.

- Create/Edit SPID file:
  Click in the “Convert Menu” on “Create/Edit SPID file” to create a new SPID file or to edit an existing one (selected SPID file in the GUI). A window is opened with the SPID editor, where the user can specify the input and output format and answer the corresponding questions. Afterwards the “SPID file” can be saved and applied (it is inserted in the “SPID conversion script” area in the PGDSpider GUI).

- Quit program:
  To quit the program push the “Quit” button or the red button with the cross in the top right edge of the window

6.2 SPID Editor

The SPID Editor is a tool to answer to the Writer and Parser questions. It also allows one to save these answers in a SPID file, which can then be reused to convert other files with the same format (use the same answers). The SPID Editor can be opened by clicking in the PGDSpider "Convert" menu on "Create/Edit SPID file".



Input format: BAPS

Output format: STRUCTURE

Select the type of the data: MICROSAT

How are Microsat alleles coded? as number of repeats

Enter the size of the repeated motif (1, 2, 3, ...):

Do you want to include a file with gene boundary definitions? No

Open gene boundaries file ... select a file.

Are DNA data coded as ACGT? No

Enter the integer that codes for nucleotide A:

Fig. 3: Screenshot of the SPID Editor

6.2.1 How to use the SPID Editor

- Select/ change input format:
  Use the put down menu to the right of the text "Input format to select or change the input format. Afterwards press the "Apply" button to apply the input format (Parser questions will change).

- Select/ change output format:
  Use the put down menu to the right of the text "Output format to select or change the output format. Afterwards press the "Apply" button to apply the output format (Writer questions will change).

- Answer Parser Questions:
  Click on the "Parser Question" tab. Afterwards the Parser questions appearing below the tab can be answered. Some of the questions do not need to be answered in every situation, because they are questions of special cases (all possible questions are listed. For more detail have a look at the corresponding data format description part).

- Answer Writer Questions:
  Click on the "Writer Question" tab. Afterwards the Writer questions appearing below the tab can be answered. Some of the questions do not need to be answered in every situation, because they are questions of special cases (all possible questions are listed. For more detail have a look at the corresponding data format description part).

- Save and Apply:
  Click on the "Save and Apply" button to save the answers in a SPID file and to apply the answers in the actual conversion process.

- Save and Close:
  Click on the "Save and Apply" button to save the answers in a SPID file and to close the SPID Editor.

- Cancel:
  Click on the "Cancel" button to close the SPID Editor without saving.

6.2.2 SPID file

The SPID file contains the Parser and Writer format and the answers to the corresponding questions. It is a plain text file encoded with UTF_8 and the ".spid" file extension.

6.3 Menus

6.3.1 PGDSpider

File Menu:

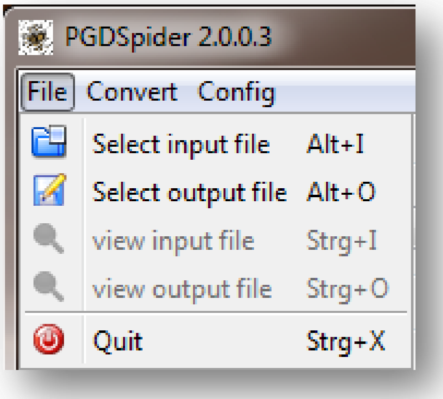


Fig. 4: Screenshot of the file menu

- Select input file: Opens a dialog box to select an input file.
- Select output file: Opens a dialog box to select the place where the output file should be saved.
- View input file: Opens a window with the input file.
- View output file: Opens a window with the output file.
- Quit: Quit the PGDSpider program

Convert Menu:

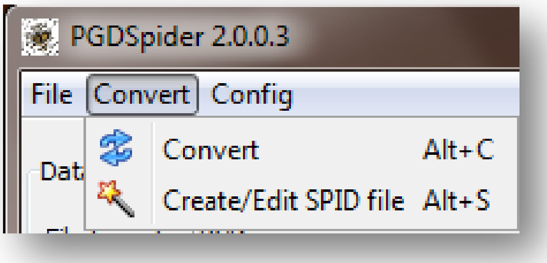


Fig. 5: Screenshot of the convert menu

- Convert: convert the specified input file into the chosen output file format and saves it.
- Create/Edit SPID file: opens a window with the SPID editor to create or edit a SPID file.

Config Menu:

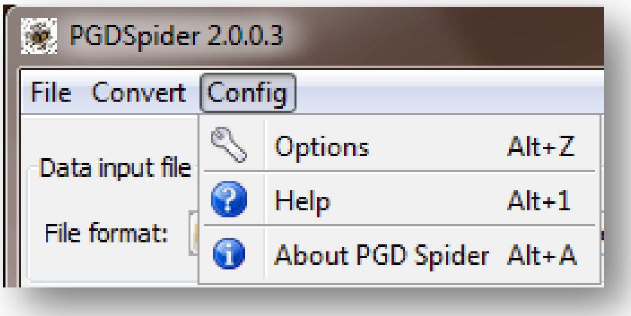
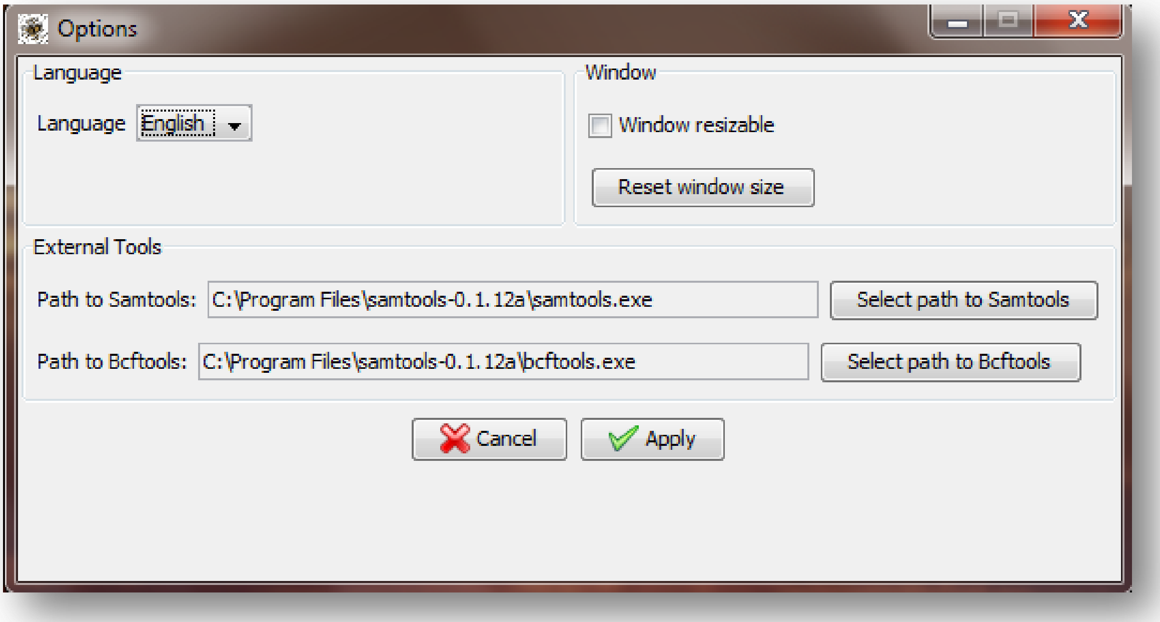


Fig. 6: Screenshot of the config menu

- Options: opens a window with option settings:



Options

Language

Language: English

Window

☐ Window resizable

External Tools

Path to Samtools: C:\Program Files\samtools-0.1.12a\samtools.exe

Path to Bcftools: C:\Program Files\samtools-0.1.12a\bcftools.exe

Fig. 7: Screenshot of the option window

- Language option:
  In the drop down menu one can select the language of the graphical user interface and the menus. One can choose between four languages: English, French, German and Italian.

- Window option:
  If "Window resizable" box is checked, the PGDSpider window can be resized. In order to reset the window size to the default, press the "Reset window size" button.

- External Tools:
  - Select Path to Samtools:
    Click on the button "Select path to Samtools" and give the path to the samtools.exe program (The Samtools distribution can be downloaded from http://samtools.sourceforge.net). Samtools (version 0.1.12/0.1.06) is needed in the conversion process of the formats SAM, BAM, VCF and BCF.
  - Select Path to Bcftools:
    Click on the button "Select path to Bcftools" and give the path to the bcftools.exe program (The Samtools distribution can be downloaded from http://samtools.sourceforge.net). Bcftools is needed in the conversion process of the formats SAM, BAM, VCF and BCF.

- Cancel/ Apply button:
  Apply or cancel the changed options

- Help: opens a window with a help file
- About PGD Spider: opens a window with short information about the PGDSpider program

6.3.2 SPID Editor

File Menu:

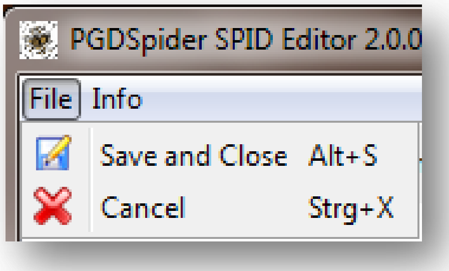


Fig. 8: Screenshot of the file menu in the SPID Editor

- Save and Close: Saves the SPID file and close the SPID Editor.
- Cancel: Cancel the SPID editor without saving.

Info Menu:

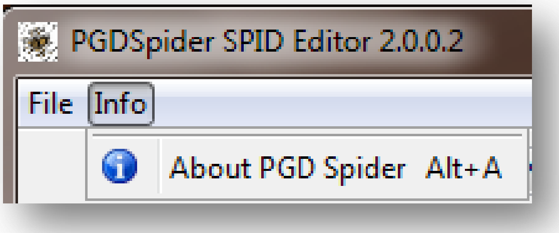


Fig. 9: Screenshot of the info menu in the SPID Editor

- About PGD Spider:
  Opens a window with short information about the PGDSpider program

6.4 Shortcuts

6.4.1 PGDSpider

Menu – Shortcuts:

Shortcut	Action
Alt + F	Open 'File' menu.
Alt + V	Open 'Convert' menu.
Alt + N	Open 'Config' menu.

Tab. 2: Menu shortcuts

File menu – Shortcuts:

Shortcut	Action
Alt + I	Select an existing input file.
Alt + O	Select an output file.
Ctrl + I	View the input file
Ctrl + O	View the output file
Ctrl + X	Quit PGDSpider application.

Tab. 3: File menu shortcuts

Convert menu – Shortcuts:

Shortcut	Action
Alt + C	Convert selected input file.
Alt + S	Create or edit the SPID file.

Tab. 4: Convert menu shortcut

Config menu – Shortcuts:

Shortcut	Action
Alt + Z	Show PGDSpider options panel.
Alt + 1	Show PGDSpider help.
Alt + A	Show some information about PGDSpider.

Tab. 5: Config menu shortcuts

6.4.2 SPID Editor

Menu – Shortcuts:

Shortcut	Action
Alt + F	Open 'File' menu.
Alt + I	Open 'Info' menu.

Tab. 6: Menu shortcuts

File menu – Shortcuts:

Shortcut	Action
Alt + S	Save and Close SPID editor.
Ctrl + X	Cancel SPID editor.

Tab. 7: File menu shortcuts

Info menu – Shortcuts:

Shortcut	Action
Alt + A	Show some information about PGDSpider.

Tab. 8: Info menu shortcuts

6.5 Log Output

The “Log Output” is an area of the graphical user interface which is used to print program messages for the user. These messages consist of 3 types:

- INFO:
  These are normal program messages with the actions the user performed (e.g.: “Opening input file”, “convert...”, etc.)
- WARN (yellow marked):
  Warning messages are written if something is missing or small error occurs but the program is able to deal with it.
- ERROR (red marked):
  If a severe error occurs during the parsing or writing of a file, the program stops and an error message is written (none or an incomplete output file is written).

7 Execute PGDSpider-cli

Execute PGDSpider-cli (command line)

- Windows: execute the command `"PGDSpider2-cli.exe"`
- Linux: execute the command
  `"java -Xmx1024m -Xms512m -jar PGDSpider2-cli.jar"`
- Mac and others: execute the command
  `"java -Xmx1024m -Xms512m -jar PGDSpider2-cli.jar"`

Increase memory:
To increase the memory PGDSpider is allowed to use start the program by executing the command `"java -Xmx1024m -Xms512M -jar PGDSpider2.jar"` and adapt the `-Xmx` parameter to your needs (`–Xmx1024m` means: maximum memory of 1’024 MB).

The command line version of the PGDSpider program can be executed with the following options (the order does not matter):

- **-? or -h:**
  To show a help text with the different options

- **-inputfile <file>** (mandatory):
  Specify the input file for the conversion process.

- -**inputformat <format>:**
  - Specify the format of the input file. This option is mandatory if the input format is not defined in the answer (SPID) file.
  - Possible input formats are:
    PGD, ARLEQUIN, BAM, BAPS, BATWING, BCF, CONVERT, FASTA, FASTQ, FSTAT, GDA, GENELAND, GENEPOP, GENETIX, HGDP_CEPH, IMMANC, IM, MEGA, MIGRATE, MSA, NEWHYBRIDS, NEXUS, PED, PHYLIP, SAM, STRUCTURE, VCF

- **-outputfile <file>** (mandatory):
  Specify the output file for the conversion process.

- **-outputformat <format>:**
  - Specify the format of the output file. This option is mandatory if the output format is not defined in the answer (SPID) file.
  - Possible output formats are:
    PGD, ARLEQUIN, BAM, BAPS, BATWING, BCF, FASTA, FASTQ, FDIST2, FSTAT, GDA, GENELAND, GENEPOP, GENETIX, GESTE_BAYE_SCAN, IMMANC, IM, KML, MEGA, MIGRATE, MSA, MSVAR, NEWHYBRIDS, NEXUS, PED, PHYLIP, SAM, STRUCTURE, VCF

- **-spid <file>** (mandatory):
  Specify the SPID file containing the pre-answered conversion questions. The SPID file can be generated with the help of the SPID Editor (see Section 6.2 SPID Editor) integrated in the PGDSpider GUI.

7.1 Examples

- call help:
  `PGDSpider_cli -?` or `PGDSpider_cli -h`

- convert a STRUCTURE file to an Arlequin file:
  ```
  PGDSpider2-cli -inputfile examples\example_Structure.txt
  -inputformat STRUCTURE -outputfile examples\output_Arlequin.arp
  -outputformat ARLEQUIN -spid examples\Structure_Arlequin.spid
  ```

- Execute the jar file itself and convert a STRUCTURE file to an Arlequin file:
  ```
  java -Xmx1024m -Xms512m -jar PGDSpider2-cli.jar -inputfile
  examples\example_Structure.txt -inputformat STRUCTURE
  -outputfile examples\output_Arlequin.arp -outputformat ARLEQUIN
  -spid examples\Structure_Arlequin.spid
  ```

8 Conversion examples

The PGDSpider distribution contains following simple example files (in the “examples” folder) to do some trial format conversion with PGDSpider:

- “example_Arlequin.arp”:
    - Data: DNA, haploid
    - Convert to:
      BAM, BAPS, BCF, FASTA, FDist2, GENELAND, MEGA, MIGRATE, NEXUS, PGD, PHYLIP, SAM, VCF

- “example_Genepop.txt”:
    - Data: Microsat, diploid
    - Convert to:
      Arlequin, BAPS, BATWING, FDist2, FSTAT, GDA, GENELAND, GENETIX, GESTE/BayeScan, Immanc, IM/IMa, MIGRATE, MSA, NewHybrids, MSVar, PGD, STRUCTURE

- “example_MEGA.meg”:
    - Data: DNA, haploid
    - Convert to:
      Arlequin, BAM, BAPS, BCF, FASTA, FDist2, GENELAND, MIGRATE, NEXUS, PGD, PHYLIP, SAM, VCF

- “example_PGD.xml” (can be displayed in a nice way with any browser by using the stylesheet_PGD.xsl):
    - Data: standard (multi-allelic), diploid, with distance matrix
    - Convert to:
      Arlequin, BAPS, FDist2, FSTAT, GDA, GENELAND, GENEPOP, GENETIX, GESTE/BayeScan, Immanc, MIGRATE , NewHybrids

- “example_SAM.sam” and its reference file “example_SAM_references.fasta”:
    - Data: NGS, diploid
    - Convert to:
      Arlequin, BAM, BAPS, BCF, FASTA, FASTQ, FDist2, GENELAND, MEGA, MIGRATE, NEXUS, PGD, PHYLIP, SAM, VCF

- “example_Structure.txt”:
    - Data: Microsat (as number of repeats), diploid (on two consecutive rows), “Phase Information” row is not present, Missing value code = -9, “Locus names” are present, “individual labels” are present, “PopData” column is present, “Recessive Alleles/Inter-Marker Distance” rows are not present.

- Convert to:
  Arlequin, BAPS, BATWING, FDist2, FSTAT, GDA, GENELAND, GENEPOP, GENETIX, GESTE/BayeScan, Immanc, IM/IMa, MIGRATE, MSA, NewHybrids, MSVar, PGD
- The spid file "Structure_Arlequin.spid" can be used for the conversion to the Arlequin format.

9 Reporting bugs and comments

If there are any bugs, send me an e-mail. Please give me a short description of the bug and tell me the input and output file format. If it is possible also attach the input file which caused the problem.

PGDSpider is an on-going project. For any comments or suggestions of further file formats, please send me an e-mail.

e-mail address: heidi.lischer@iee.unibe.ch

10 File format descriptions

In the next sections, a short description of every supported file format is provided. The table below shows the file extensions and handled data types of the different file formats:

		handled data types									
format	**File extension**	**NGS**	**DNA**	**RNA**	**Microsat**	**SNP**	**RFLP**	**AFLP**	**Standard**	**Frequency**	**distance**
Arlequin	.arp		x		x	x	x		x	x	
BAM	.bam	x	x	x							
BAPS	.txt		x		x	x		x	x		
BATWING	.txt				x	x					
BCF	.bcf	x	x	x		x					
CONVERT	.txt				x	x	x	x	x		
FASTA	no standard, .fa, .mpfa, .fna, .fsa, .fas, .fasta, .txt		x	x		x					
FASTQ	no standard, .fastq, .fq, .txt	x									
FDist2 (datacal)	no standard		x		x				x		
FSTAT	.dat				x				x		
GDA	.nex				x	x	x	x	x		
GENELAND	.txt		x		x	x		x	x		
GENEPOP	.txt				x				x		
GENETIX	.gtx				x	x	x	x	x		
GESTE / BayeScan	no standard				x	x		x	x		
HGDP-CEPH	.arp		(x)		(x)	x	(x)		(x)	(x)	
Immanc	.inp or .txt				x	x	x		x		
IM/IMa	.u or .txt		x		x						
KML	.kml										
MEGA	.meg		x	x							x
MIGRATE	no standard, .txt		x		x	x			x		
MSA	.dat, .txt				x						
NewHybrids	.dat, .txt				x			x	x		
MSVar	no standard				x						
NEXUS	.nex		x	x							
PED	.ped					x					
PGD	.xml		x	x	x	x	x	x	x	x	x
PHYLIP	.txt		x	x							x
SAM	.sam	x	x	x							
STRUCTURE	no standard				x	x	x	x			
VCF	.vcf	x	x	x		x					

Tab. 9: Table of the different file formats and their handled data types and file extensions

10.1 PGD

PGD version 1.0

PGD (Population Genetics Data) is a file format designed to contain population genetics data. The aim of this format is to facilitate the transfer among several population genetics software packages. PGD plays an important role in the new data format converter PGDSpider.

PGD is written in XML and is therefore independent of any particular computer system and extensible for future needs (W3Schools, 2008). The XML structure can easily be processed by computer programs. An additional XSLT style sheet makes it possible to display the data in an understandable and comprehensive way. This XSLT style sheet is delivered within the PGDSpider download (stylesheet_PGD.xsl).

The PGDSpider distribution also includes an XML Schema (PGD_schema.xsd), which defines the structure of the PGD file. The purpose of an XML Schema is to define the legal building blocks of an XML document and the allowable contents (W3Schools, 2008). The provided XML Schema can be used to validate a PGD file.

10.1.1 Data type handled

PGD is able to handle the following data types:

- DNA
- NGS (Next-Generation Sequencing data)
- Microsat (coded as number of repeats!)
- RFLP
- SNP
- AFLP
- Standard
- Frequency (Allele Frequency)
- etc.

10.1.2 PGD format

The PGD format is written in XML (eXtensible Markup Language) and can be created and edited in any text editor (file extension *.xml). An XML document has an ordered, labelled tree structure with following rules:

- An XML declaration needs to be included at the beginning of the file:
  `<?xml version="1.0" encoding="iso-8859-1"?>`
- If a style sheet exists, the name of an XSL style sheet reference must be mentioned with the absolute or relative file path to the style sheet after the declaration:
  `<?xml-stylesheet type="text/xsl" href="stylesheet_PGD.xsl"?>`
- A root element is needed. This element is "the parent" of all other elements and includes all other elements. In the PGD file format the root element is named: `<PGD>`

- All XML elements need to have a start and a closing tag and have to be properly nested
- XML tags are case sensitive
- Attribute values have to be within quotes
- The characters “<” and “&” are strictly illegal within text tags. They can be replaced with the expression “&lt;“ (for “<”) and “&amp;” (for “&”).
- Comments have to be written within “<!--“ and “-->”:
  `<!-- This is a comment -->`

The PGD file format has a block structure and the information’s are saved in a hierarchical way. Therefore the format is very modular and general information can be saved at a higher level than information specific for one individual. This is very convenient because general information’s need to written only once.

A short description of the different blocks can be found below:

Root Element:

The root element named “PGD” encapsulates all other elements of the XML file.

Header block:

The header block contains the general information’s about the data. The tag is named “header” and can contain an attribute named “title=” that defines the title of the data. The header block has the following sub tags:

- <organism> (optional):
  - Value: String
  - Indicates from which organism the data come from
- <numPop> (mandatory):
  - Value: Integer
  - gives the number of populations listed in the file
- <ploidy> (mandatory):
  - Value: “mixed” or any Integer
  - Specify the ploidy level of the data
  - It contains the value “mixed” if the ploidy level is not the same in every population or individual.
- <missing> (mandatory):
  - Value: Character
  - Character which codes missing values
- <gap> (optional):
  - Value: Character
  - Character which codes gaps

- <gameticPhase> (optional):
    - Value: “known” or “unknown”
    - Define if the gametic Phase of the genotypes is known or not
- <recessiveData> (optional):
    - Value: “no” or “yes”
    - Define if genotypic data present a recessive allele

DataDescription block:

The dataDescription block contains specifications about the different loci. The tag is named “dataDescription” and contains following sub tags:

- <numLoci> (mandatory):
    - Value: Integer
    - Gives the number of loci studied
- <dataType> (mandatory):
    - Value: “mixed”, “DNA”, “NGS”, “Microsat”, “RFLP”, “SNP”, “AFLP”, “Standard”, “Frequency” or etc.
    - Defines the data type of the data
    - It has the value “mixed” if the data contain different data types
- <locus> with attribute “id=” (optional):
    - Describes the different loci contained in the file
    - Could be repeated for multiple times (as many times as there are different loci)
    - The “id” attribute gives the name of the locus

The <locus> tag has following sub tags:

- <locusDataType> (optional):
    - Value: “DNA”, “NGS”, “Microsat”, “RFLP”, “SNP”, “AFLP”, “Standard”, “Frequency” or etc.
    - Only required if the <dataType> tag contains the value “mixed”
    - Defines the data type of the locus
- <locusChromosome> (optional):
    - Value: Integer, “X”, “Y”, “W”, “Z”, “mtDNA” or etc.
    - Gives the chromosome the locus come from
- <locusLocation> (optional):
    - Value: Integer
    - Gives the location/position on a chromosome the locus come from
- <locusGenic> (optional):
    - Value: “coding” or “noncoding”
    - Defines if the locus codes for a gene or not
- <locusLength> (optional):
    - Value: Integer

    - Gives the length of the locus in number of base pairs (for DNA)
- <locusLinks> (optional):
    - Value: String
    - Here one can specify internet links (URL) to locus information
- <locusComments> (optional):
    - Value: String
    - Here one can put comments about the locus

Population block:

The population block contains information about the population and their individuals with the data. This block could be repeated for multiple times (as many times as there are different populations in the sample). This block is structured differently if the data are aligned or not, and if the data are of the same data type or not. The tag is named “population” and can contain an attribute named “name=” which defines the name of the population. The population block has the following sub tags:

- <popSize> (mandatory):
    - Value: Integer
    - Defines the number of individuals in the population
- <popGeogCoord> (optional):
    - Value: longitude, latitude
    - Defines the geographic coordinate of the population
- <popLingGroup> (optional):
    - Value: String
    - Defines the linguistic group to which the population belongs
- <popPloidy> (optional):
    - Value: “mixed” or any Integer
    - Only required if the <ploidy> tag in the header block contains the value “mixed”
    - Specify the ploidy level of the data in this population
    - It contains the value “mixed” if the ploidy level is different between different individuals.
- <popLoci> (optional):
    - Value: String, String, ...
    - Only if all individuals in this population have the same loci
    - Defines the names of the loci in the data for this population, separated by comma
    - The loci have to be of the same type
    - If the data are nucleotide sequence data, just include one locus in this tag (if more than one locus exist one has to repeat the whole <ind> tag for every locus and specify the locus name in the <indLoci> tag). For example:

```
<ind name="1">
 <indLoci> loci one </indLoci>
 <data> GACTCTCTACGTAGCATCCGATGACGATA </data>
 <data> GACTCTCTACGTAGCATCCGATGACGATA </data>
</ind>

<ind name="1">
 <indLoci> loci two </indLoci>
 <data> GACTCTCTACGTAGCATCCGATGACGATA </data>
 <data> GACTCTCTACGTAGCATCCGATGACGATA </data>
</ind>
```

- <ind> with attribute “name=” (mandatory):
  - Defines the different individuals in this population
  - Could be repeated for multiple times (as many as there are different individuals in this population)
  - The “name” attribute gives the name of the individual

The <ind> tag has following sub tags:

- <indGeogCoord> (optional):
  - Value: longitude, latitude
  - Defines the geographic coordination of the individual
- <indLingGroup> (optional):
  - Value: String
  - Defines the linguistic group which the individual belongs to
- <indLoci> (mandatory, if aligned data with different data types)
  - Value: String, String, ...
  - Only if the data are of different data types in this population
  - Defines the loci names of the data with the same data type in this individual separated by “,”
  - The loci must be of the same data type and all loci of the same data type have to be included (only one loci tag allowed per data type, except for nucleotide sequence data)
  - If the data are nucleotide sequence data, just include one loci in this tag (if more than one loci exist repeat the whole ind tag for every loci)
- <indPloidy> (optional):
  - Value: Integer
  - Only required if the <popPloidy> tag in the population block contains the value “mixed”
  - Specify the ploidy level of the data in this individual
- <indFreq> (optional, but obligatory if “Frequency” data type)
  - Value: Integer
  - Defines the absolute frequency of this genotype in the population
- <data> (mandatory, if non-NGS data):
  - Value: locus data, locus data, ...

    - Can be repeated for multiple times (as many as there are different reads in this individual)
    - Contains the data of one read of each specified locus (same order as the locus names) separated by a comma
- <read> with attribute “id=” (mandatory, if NGS data (Next Generation Sequencing)):
    - Defines the different reads in this individual
    - Could be repeated for multiple times (as many as there are different reads in this individual)

The <read> tag has the following sub tags:

- <start> (mandatory):
    - Value: Integer
    - Defines the starting point of the sequence
- <length> (optional):
    - Value: Integer
    - Gives the length of the sequence
- <data> (mandatory):
    - Value: locus data
    - Contains the data of one read for the specified locus
- <quality> (optional):
    - Value: white space separated Integers
    - Contains the quality scores of the read

Structure block:

The structure block is optional. It contains the information about the genetic structure of the population (grouping). The tag is named “structure” and can contain an attribute named “name=” which defines the name of the structure. The structure block has following sub tags:

- <numGroups> (mandatory):
    - Value: Integer
    - Defines the number of groups of populations
- <group> with attribute “name=” (mandatory):
    - Value: String, String, ...
    - Defines which populations belong to this group. The population names are separated by a comma
    - Could be repeated for multiple times (as many as there are different groups)
    - The “name” attribute is the name of the group

DistanceMatrix block:

The distanceMatrix block is optional. It contains information about the genetic distances between haplotypes. The tag is named “distanceMatrix” and can contain an attribute named “name=” which defines the name of the distance matrix. The distanceMatrix block has following sub tags:

- <matrixSize> (mandatory):
    - Value: Integer
    - Defines the size of the distance matrix
- <matrixLabels> (mandatory):
    - Value: String, String, ...
    - Defines the labels of the distance matrix separated by a comma
- <matrix> (mandatory):
    - Value: Integer (line break) Integer, Integer (line break) ...
    - Gives the genetic distances of each specified individual to each other (same order as in the <matrixLabels> tag
    - Data have to be in the lower triangle with diagonals. Lines are separated by a line break and values by a comma

10.1.3 Schema of the PGD format

Specifications

- Root element: <PGD>
- Header/loci block: obligatory and only one per file
- Population block: obligatory and can exist multiple times
- Structure/ distanceMatrix block: optional and only one per file
- Microsat data must be coded as number of repeats
- Only one loci tag is allowed per data type, i.e. all loci of the same data type have to be in the same tag
- Nucleotide sequence data: just one locus in one tag (popLoci/ indLoci tag). If more than one loci exist repeat the whole ind tag for every loci
- Distance Matrix: lower triangle with diagonals

Block	**sub tags**			**data type**		
				Microsat, SNP, AFLP, RFLP, standard	**DNA**	**Freq**
header (attribute: title)	organism			o	o	o
	numPop			x	x	x
	ploidy * *(-> mixed / 1 / 2 /...)*			x (a 4)	x (a 4)	x
	missing			x	x	
	gap			o	o	
	gameticPhase *(->known/ unknown)*			o	o	
	recessiveData *(-> no / yes)*			o	o	
dataDescription	numLoci			x	x	
	dataType * *(-> mixed / DNA /NGS / Microsat / SNP / AFLP / RFLP/ Standard / Frequency /...)*			x (a 1)	x (a 1)	X
	locus (attribute: id)	locusDataType *(-> DNA / NGS / Microsat / SNP / AFLP / standard /...)*		a 1	a 1	
		locusChromosome *(->number/ X/ Y/ W/ Z/ mtDNA/ ...)*		o	o	
		locusLocation		o	o	
		locusGenic *(-> coding/ noncoding)*		o	o	
		locusLength		o	o	
		locusLinks *(->URL)*		o	o	
		locusComments		o	o	
population (attribute: name)	popSize			x	x	x
	popGeogCoord * *(lon, lat)*			o (a 2)	o (a 2)	o
	popLingGroup *			o (a 3)	o (a 3)	o
	popPloidy * *(-> mixed / 1 / 2 /...)*			a 4	a 4	
	popLoci *(locus name, locus name,...)* -> all locus of same data type			o	o	
	ind (attribute: name)	indGeogCoord *(lon, lat)*		o (a 2)	o (a 2)	
		indLingGroup		o (a 3)	o (a 3)	
		indLoci *(locus name, locus name,...) -> all locus of same data type*		o	o	
		indPloidy *(-> 1 / 2 /...)*		a 4	a 4	
		indFreq *(absolute Freq)*		o	o	x
		data *(locus data, locus data, ...)*		x	x	
		read (attribute: id)	**start**		x	
			length		o	
			data		x	
			quality		o	
structure (o) (attribute: name)	numGroups			x	x	x
	group (attribute: name) *(pop name, pop name, ...)*			x	x	x
distanceMatrix (attribute: name) (o)	matrixSize			x	x	x
	matrixLabels *(name, name,...)*			x	x	x
	matrix *(number (line break) number, number (line break)...)*			x	x	x

Tab. 10: Schema of the PGD file format

Legend: Non NGS data NGS data x: obligatory

same data type (loci) for all individuals (genotypes) o: optional

different data types (aligned within each locus) a: alternative to

* if all populations or individuals are identical for a given tag

10.1.4 PGD file examples

- Data of two loci with Standard data type from two diploid populations:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="stylesheet_PGD.xsl"?>

<PGD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PGD_schema.xsd">
  <header title="Fake HLA data">
    <numPop> 2 </numPop>
    <ploidy> 2 </ploidy>
    <missing> ? </missing>
    <gap> - </gap>
    <gameticPhase> known </gameticPhase>
  </header>

  <dataDescription>
    <numLoci> 2 </numLoci>
    <dataType> Standard </dataType>
    <locus id="loci one">
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23892 </locusLocation>
    </locus>
    <locus id="loci two">
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23992 </locusLocation>
    </locus>
  </dataDescription>

  <population name="A sample of  Algerians">
    <popSize> 2 </popSize>
    <popLoci> loci one, loci two </popLoci>
    <ind name="1">
      <indGeogCoord> 35, 4 </indGeogCoord>
      <indLingGroup> African </indLingGroup>
      <data> 1104, 0200 </data>
      <data> 0700, 0301 </data>
    </ind>
    <ind name="3">
      <indGeogCoord> 36, 4 </indGeogCoord>
      <indLingGroup> Africanic </indLingGroup>
      <data> 0302, 0200 </data>
      <data> 1310, 0402 </data>
    </ind>
  </population>

  <population name="A sample of Bulgarians">
    <popSize>1</popSize>
    <popGeogCoord> 35, 4 </popGeogCoord>
    <popLingGroup> African </popLingGroup>
    <popLoci> loci one, loci two </popLoci>
    <ind name="2">
      <data> 1101, 0301 </data>
      <data> 0700, 0200 </data>
    </ind>
  </population>

  <structure name="My population structure">
    <numGroups> 2 </numGroups>
    <group> A sample of Algerians </group>
    <group> A sample of Bulgarians </group>
  </structure>

  <distanceMatrix name="Faked distance matrix">
    <matrixSize> 3 </matrixSize>
    <matrixLabels> 1, 2, 3</matrixLabels>
    <matrix> 0
             1, 0
             3, 4, 0
    </matrix>
  </distanceMatrix>
</PGD>
```

- Data of two loci with different data types (Standard and DNA) from two diploid populations:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="stylesheet_PGD.xsl"?>

<PGD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PGD_schema.xsd">
  <header title="Fake HLA data">
    <numPop> 2 </numPop>
    <ploidy> 1 </ploidy>
    <missing> ? </missing>
    <gap> - </gap>
    <gameticPhase> known </gameticPhase>
  </header>

  <dataDescription>
    <numLoci> 2 </numLoci>
    <dataType> mixed </dataType>
    <locus id="loci one">
      <locusDataType> Standard </locusDataType>
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23892 </locusLocation>
    </locus>
    <locus id="loci two">
      <locusDataType> DNA </locusDataType>
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23992 </locusLocation>
      <locusLength> 29 </locusLength>
    </locus>

 </dataDescription>
  <population name="A sample of Algerians">
    <popSize> 4 </popSize>
    <ind name="1">
      <indLoci> loci one </indLoci>
      <data> 1104 </data>
    </ind>
    <ind name="2">
      <indLoci> loci one </indLoci>
      <data> 0302 </data>
    </ind>
    <ind name="1">
      <indLoci> loci two   </indLoci>
      <data> GACTCTCTACGTAGCATCCGATGACGATA </data>
    </ind>
    <ind name="2">
      <indLoci> loci two </indLoci>
      <data> GACTGTCTGCGTAGCATACGACGACGATA </data>
    </ind>
  </population>

  <population name="A sample of Bulgarians">
    <popSize>2</popSize>
    <ind name="5">
      <indLoci> loci one </indLoci>
      <data> 1103</data>
    </ind>
    <ind name="5">
      <indLoci> loci two </indLoci>
      <data> GCCTGTCTGCGTAGCATAGGATGACGATA </data>
    </ind>
  </population>

  <structure name="My population structure">
    <numGroups> 2 </numGroups>
    <group>A sample of Algerians </group>
    <group>A sample of Bulgarians </group>
  </structure>
</PGD>
```

- NGS data of two loci from two haploid populations:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<?xml-stylesheet type="text/xsl" href="stylesheet_PGD.xsl"?>

<PGD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="PGD_schema.xsd">
  <header title="faked example data">
    <organism> homo sapiens sapiens </organism>
    <numPop> 2 </numPop>
    <ploidy> 1 </ploidy>
    <missing> ? </missing>
    <gap> - </gap>
  </header>

  <dataDescription>
    <numLoci> 2 </numLoci>
    <dataType> NGS </dataType>
    <locus id="loci one">
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23892 </locusLocation>
      <locusGenic> coding </locusGenic>
      <locusLinks> www.loci.ch/faked_adress/ </locusLinks>
      <locusComments> faked </locusComments>
    </locus>
    <locus id="loci two">
      <locusChromosome> 3 </locusChromosome>
      <locusLocation> Hs8_23992 </locusLocation>
      <locusGenic> noncoding </locusGenic>
    </locus>
  </dataDescription>

  <population name="pop 1">
    <popSize> 2 </popSize>
    <ind name="1">
      <indGeogCoord> 35, 4 </indGeogCoord>
      <indLingGroup> African </indLingGroup>
      <indLoci> loci one </indLoci>
      <indFreq> 10 </indFreq>
      <read>
        <start> 230 </start>
        <length> 70 </length>
        <data> ATTAGCACCCAAAGCTAAGATTCTAATTTAAACTATTCTCTGTTCTTTCATGGGGAAGCAGATTTGGGTA </data>
        <quality> IIIIIIHHHHHHHHHBIIIIIIIIIIIIIIIIIIIIIIIIIHHHHHHHHHHHHHHHHHHHHHHHHHHHHA </quality>
      </read>
      <read>
        <start> 240 </start>
        <length>71 </length>
        <data> AAAGCTAAGATTCTAATTTAAACTATTCTCTGTTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTA    </data>
        <quality> IIIIIIHHHHHHHHHBIIIIIIIIIIIIIIIIIIIIIIIIIHHHHHHHHHHHHHHHHHBBBBBHHHHHA,, </quality>
      </read>
    </ind>

    <ind name="2">
      <indGeogCoord> 36, 4 </indGeogCoord>
      <indLingGroup> Africanic </indLingGroup>
      <indLoi> loci one </indLoci>
      <indFreq> 11 </indFreq>
      <read>
        <start> 273 </start>
        <length> 57 </length>
        <data> TCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTCACCCATCAACAT </data>
        <quality> IIIIIIHHHHHHHHHBIIIIIIIIIIIIIIIIIIIIIIIIIHHHHHHHHHHHHHHHH </quality>
      </read>
    </ind>
  </population>

  <population name="pop 2">
    <popSize> 1 </popSize>
    <popGeogCoord> 8, 48 </popGeogCoord>
    <popLingGroup> European </popLingGroup>
    <popLocus> loci two </popLocus>
    <ind name="4">
      <read>
        <start> 265 </start>
        <length> 65 </length>
        <data> TTCTCTGTTCTTTCATGGGGAAGCAGATTTGGGTACCACCCAAGTATTGACTCACCCATCAACAT </data>
        <quality> IHHHHHHHHHBIIIIIIIIIIIIIIIIIIIIIIIIIIHHHHHHHHHHHHHHHHHHHHHHHHHHHA </quality>
      </read>
  </population>
</PGD>
```

10.2 ARLEQUIN

ARLEQUIN version 3.5 (released 24 February 2010)

ARLEQUIN provide a large set of basic methods and statistical tests, in order to extract information on genetic and demographic features of a collection of population samples. It is able to compute standard genetic diversity indices, to estimate allele and haplotype frequencies, to test departure from linkage equilibrium, departure from selective neutrality and demographic equilibrium, to estimate parameters from past population expansion, and to analyse population subdivision under the AMOVA framework (Excoffier and Lischer, 2010).

10.2.1 Data type handled

ARLEQUIN can handle haploid and diploid data of following data types:

- DNA
- RFLP
- SNP
- Microsatellite
- Standard data
- Allele frequency data

10.2.2 ARLEQUIN format

The input files should have an “*.arp” extension (for ARLEQUIN Project). They are structured into two main sections:

- Profile section (mandatory)
- Data section (mandatory):
  - Haplotype list (optional)
  - Distance matrices (optional)
  - Samples (mandatory)
  - Genetic structure (optional)
  - Mantel tests (optional)

Profile section:

The profile section contains the properties of the data. The beginning is indicated by [Profile].

Specify:

- Title (string within “”): `Title=”title xy”`
- Number of samples (integer 1-1000): `NbSamples =3`
- Type of data (DNA, RFLP, MICROSAT, STANDARD, FREQUENCY): `DataType = DNA`
- Haplotypic/genotypic data (0/1): `GenotypicData = 0`

Optionally (default value):

- locus separator (WHITESPACE, TAB, NONE, …): `LocusSeparator = TAB`
- gametic phase known/unknown (1/0): `GameticPhase = 1`
- recessive/ co-dominant allele (1/0): `RecessiveData = 1`
- code for recessive allele (string within “null”): `RecessiveAllel =”xxx”`
- code for missing data (character within “?” or ‘?’): `MissingData = ‘$’`
- frequencies as absolute/relative values (ABS/REL): `Frequency = ABS`
- significant digits for haplotype frequency outputs (real number 1e-2 – 1e-7(1e-5)): `FrequencyThreshold = 0.00001`
- convergence criterion for the EM algorithm (real number 1e-7 – 1e-12): `EpsilonValue = 1e-10`

Data section:

The data section contains the raw data to be analysed. The beginning is indicated by [Data]. It contains several subsections:

- Haplotype list (optional):
  One can define a list of haplotypes, which are used for all samples. It is possible to define the list in an external file.
    - intern:

```
[[HaplotypeDefinition]]   #start the section of Haplotype definition
  HaplListName="list1"    #give any name you wish to this list
  HaplList={
   h1 A T                  #on each line, the name of the haplotype
   h2 G C                  #is followed by its definition.
   h3 A G
   h4 A A
   h5 G G
  }
```

    - extern:

```
[[HaplotypeDefinition]]    #start the section of Haplotype definition
  HaplListName="list1"     #give any name you wish to this list
  HaplList = EXTERN "hapl_file.hap"
```

- Distance matrix (optional):
  This subsection contains a matrix of genetic distances between haplotypes. It is also possible to define the matrix in an external file.

- intern:

```
[[HaplotypeDefinition]]    #start the section of Haplotype definition
  HaplListName="list1"     #give any name you wish to this list
  HaplList={
   h1  A  T                 #on each line, the name of the haplotype
   h2  G  C                 #is followed by its definition.
   h3  A  G
   h4  A  A
   h5  G  G
  }
```

- extern:

```
[[DistanceMatrix]]       #start the distance matrix definition section
   MatrixName= "none"    #name of the distance matrix
   MatrixSize= 4         #size = number of lines of the distance matrix
  MatrixData= EXTERN "mat_file.dis"
```

- Samples (obligatory):
  This subsection defines the haplotypic/genotypic content of the different samples:
  - start of the subsection: `[[Samples]]`
  - name for each sample (string within “ ”): `SampleName = “name xy”`
  - size of sample (integer value): `SampleSize = 732`
  - data itself (list of haplotypes or genotypes and their frequencies, entered with braces “{ }”):

```
[[Samples]]                #start the samples definition section
  SampleData={
   id1  1   ACGGTGTCGA
   id2  2   ACGGTGTCAG
   id3  8   ACGGTGCCAA
   id4  10  ACAGTGTCAA
   id5  1   GCGGTGTCAA
  }
```

Frequency data:

```
SampleData={
  id1  1
  id2  2
  id3  8
  id4  10
  id5  1
}
```

Haplotypic data: Define for each haplotype its identifier and sample frequency (if no haplotype list has been defined: specify also allelic content of the haplotype)

Genotypic data: for each genotype its identifier, sample frequency and allelic content (on two separate lines) is needed.

```
Id1   2   ACTCGGGTTCGCGCGC        # the first pseudo-haplotype
          ACTCGGGCTCACGCGC        # the second pseudo-haplotype
```

- Genetic structure (only required for AMOVA):
  The genetic structure specifies the hierarchical genetic structure of the samples. It is possible to define groups of populations.
  - start of the subsection: `[[Structure]]`
  - name for the genetic structure (string within ” ”): `StructureName = “name”`
  - number of groups defined in the structure (int value): `NbGroups = 5`
  - group definitions (list containing the names of the samples belonging to the group, entered within braces “{ }”):

```
NbGroups=2
Group ={
  population1
}
Group ={
  population2
  population3
}
```

- Mantel test settings (optional):
  This subsection specifying some distance matrices. The goal is to compute a correlation between the Ymatrix and X1 or a partial correlation between the Ymatrix, X1 and X2. The Ymatrix can be either a pairwise population FST matrix or a custom matrix entered into the project by the user. X1 (and X2) have to be defined in the project.
  - start of the subsection: `[[Mantel]]`
  - size of the matrices (pos. integer value): `MatrixSize= 5`
  - number of matrices among which we compute the correlations (2/3): `MatrixNumber = 2`
  - matrix that is used as genetic distance (“fst” (→Y=Fst)/ “log_fst” (→Y=log(Fst))/ “slatkinlinearfst” (→Y=Fst/(1-Fst))/ “log_slatkinlinearfst” (→Y=log(Fst/(1-Fst)))/ “nm” (→Y=(1-Fst)/(2 Fst))/ “custom” (→Y= user-specified in the project)): `YMatrix = “fst”`
  - labels that identify the columns of the YMatrix (list containing the names of the label name belonging to the group, entered within braces “{ }”):

```
YMatrixLabels = {
  "Population1 " "Population4" "Population2"
  "Population8" "Population5"
}
```

- keyword that allows to define a matrix with witch the correlation with the YMatrix is computed:

```
DistMatMantel={
  0.00
  3.20 0.00
  0.47 0.76 0.00
  0.00 1.23 0.37 0.00
  0.22 0.37 0.21 0.38 0.00
}
```

- Labels defining the sub-matrix on which the correlation is computed:

```
UsedYMatrixLabels={
  "Population1 "
  "Population5"
  "Population8"
}
```

Example input file:

The following small example is a project file containing four populations. The data type is STANDARD genotypic data with unknown gametic phase:

```
[Profile]
Title="Fake HLA data"
  NbSamples=2
  GenotypicData=1
  GameticPhase=0
  DataType=STANDARD
  LocusSeparator=WHITESPACE
  MissingData='?'
[Data]

[[Samples]]
  SampleName="A sample of 6 Algerians"
  SampleSize=6
  SampleData={
   1 1 1104 0200
       0700 0301
   3 3 0302 0200
       1310 0402
   4 2 0402 0602
       1502 0602
  }
  SampleName="A sample of 11 Bulgarians"
  SampleSize=5
  SampleData={
   1 1 1103 0301
       0301 0200
   2 4 1101 0301
       0700 0200
  }
```

```
[[Structure]]
  StructureName="My population structure"
  NbGroups=2
  Group={
   "A sample of 6 Algerians"
  }
  Group={
   "A sample of 11 Bulgarians"
  }
```

10.2.3 Links and References

Website: http://cmpg.unibe.ch/software/arlequin35/,

Manual: http://cmpg.unibe.ch/software/arlequin35/man/arlequin35.pdf

(Excoffier and Lischer, 2010)

10.2.4 Special PGDSpider input/output questions

- Input: none

- Output:
  - Specify which data type should be included (optional):
    If more than one allowed data type exists, one have to select the data type which should be included in the output file (only one data type can be analysed per file).
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.3 BAM

BAM (17. April 2011)

BAM is a generic format for storing large nucleotide sequence alignments. It is the binary equivalent to SAM for intensive data processing.

The program SAMtools provide various utilities for manipulating alignments in the BAM/SAM format, including sorting, merging, indexing and generating alignments in a per-position format (Li, et al., 2009).

The conversion process of the format BAM needs the programs Samtools (version 0.1.12/0.1.06) and Bcftools, which can be downloaded from http://samtools.sourceforge.net. The paths to the program files (samtools.exe and bcftools.exe) have to be specified in the “Config” menu under “Options” (see section 5.3.1 PGDSpider menus).

10.3.1 Data type handled

BAM can handle data of following type:

- DNA
- NGS (Next-Generation Sequencing data)

10.3.2 BAM format

BAM is the binary file format of SAM with following file extension: *.bam
For a detailed description of the format see http://samtools.sourceforge.net/SAM1.pdf.

10.3.3 Links and References

Website: http://samtools.sourceforge.net,
Manual: http://samtools.sourceforge.net/SAM1.pdf

(Li, et al., 2009)

10.3.4 Special PGDSpider input/output questions

- Input:
    - Reference file:
      Choose the file with the reference sequences
    - Select what should be imported:
      Defines if all reads, the consensus sequences or only the variant sites (SNP) should be imported

  - Concatenate consensus sequences from different reference data (only works if you choose to import consensus sequences):
    Specify if consensus sequences coming from different reference sequences should be concatenated or not
  - What is the ploidy of the data:
    Define if the data are haploid or diploid

  - Only import following regions (optional):
    Defines which regions should be imported. Regions should be defined in following format: refSeqName:start:end, multiple regions: separate it with white spaces

- Output:
  - Save an additional file with reference sequences:
    Saves a file with the reference sequences
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.4 BAPS

BAPS version 5.4 (29 April 2010)

BAPS (Bayesian Analysis of Population Structure) is a program for Bayesian inference of the genetic structure in a population. One can cluster molecular data (assign the data to different groups) and perform admixture analysis at the group or at the individual level (Tang, et al., 2009).

10.4.1 Data type handled

BAPS can handle haploid or diploid data of the following data types:

- DNA (also tetraploid data possible)
- SNP (sequence/numeric)
- AFLP
- Microsatellite
- Standard (multi-allelic markers)

10.4.2 BAPS format

Clustering of individuals:

The molecular data are assigned to the corresponding individual. The individuals again can be assigned to groups (populations).

- The data file contains a data matrix:
    - columns: loci at which the individuals were observed
    - rows: individuals
    - additional column at the right end of the matrix (last column):
      Each row contains an index of the individual whose alleles are reported. There can be more than one row per individual (e.g. in diploids)
- alleles are indexed with any non-negative integer
- individuals: indices start with 1 for the first individual and end with the value that corresponds to the total number of individuals
- missing alleles: coded by a negative integer (e.g. -999 or -9)
- If the populations of the individuals are known, one can input two additional files:
  The first file contains the names of the populations and the second file contains the indices of the first individual of each sampling populations.

- Example (cluster 5 diploid individuals (two rows per individual). The first individual has alleles 5 and 7 at the first locus and so on. Individuals 1, 2 and 3 were sampled in America and individuals 4 and 5 in Europe):

- data file:

```
5         2         1
7         2         1
5         8         2
3         9         2
2         5         3
-999      5         3
5         -999      4
2         3         4
3         8         5
2         5         5
```

- name file:

```
American
European
```

- Index file:

```
1
4
```

Clustering groups of individuals:

The molecular data are assigned to the corresponding group of individuals. Data of individuals coming from the same group (population) cannot be separated at the individual level.

- The data file contains a data matrix:
    - columns: loci at which the individuals were observed
    - rows: individuals
    - Last column contains the index of the group that is the origin of the alleles on the particular row
- the names of the groups can be given in a separate file

- example (data from four distinct groups)
    - data file:

```
5      2      1
7      2      1
5      8      1
3      9      2
2      5      2
-999   5      3
5      -999   4
2      3      4
3      8      4
2      5      4
```

- name file:

```
American
European
African
Asian
```

Trained clustering

Find the best clustering for individuals with unknown origin with the help of individuals whose origin is known.

- Not included, because not enough information is available from a PGD file to generate this kind of data file.

Spatial clustering

The spatial clustering is a genetic mixture analysis using a spatial model. The spatial clustering model requires the coordinate data for the clustered units (groups or individuals).

- Uses the same files as: “Clustering of individuals” or “Clustering groups of individuals”. But an additional file with coordinate values have to be given:
  The coordinate file contains as many rows as there are individuals (spatial clustering of individual’s → sample coordinates of each individual) or groups (spatial clustering of groups → sample coordinates of each group) in the data file.
- Missing geographic coordinates are coded as two consecutive zeros

- Example (individual 1 has the coordinates: 172, 88 and individual 4 has missing geographic coordinates):
  - Data file: see first example
  - Coordinate file:

```
172  88
155  96
180  78
0    0
-18  81
```

Clustering of linked molecular data (nucleotide sequence data)

Clustering of linked molecular data is genetic mixture analysis done either for haploid sequence data, phased diploid/tetraploid sequence data or for linked marker data for which a single allele is recorded per locus.

Haploid data need a single data row per individual, diploid two and tetraploid four rows. There are a numeric and a sequence input format:

- numeric format:
  - replace each nucleotide (A,C,G,T) with a unique positive integer and missing values with a negative integer (e.g.: -999)
  - Individual indices are located after the sequence and separated by a space
  - example: a single data row for individual 110 with sequence AACCG-T could look like this:

```
65 65 67 67 71 -999 84 110
```

- sequence format:
    - Individual Indexes are located after the sequence and separated by a space
    - example (diploid):

```
ATTTGCCTACGTAGCCAATT 1
TTACCGACCTTAAAAACCTT 1
ATTTCCCAAAGGGTTTAAAA 2
TAACCGGACATAGCCAATAA 2
```

- Need to concatenate the sequences from all considered genes into a single one and tell the program about the gene boundaries in a separate file:
    - The number of rows is equal to the number of genes
    - at each row, the integers refer to columns of the data matrix that correspond to the specific gene
    - Additional zeros are used to fill the rows to have an equal number of columns
    - Example ("linkage map" of 3 genes. The first gene corresponds to the columns 1-10 in the data matrix, the second gene to the columns 11-19 and so on):

```
1  2  3  4  5  6  7  8  9  10
11 12 13 14 15 16 17 18 19 0
20 21 22 23 24 25 26 27 0  0
```

10.4.3 Links and References

Website: http://web.abo.fi/fak/mnf//mate/jc/software/baps.html

Manual: http://web.abo.fi/fak/mnf//mate/jc/software/BAPS5manual.pdf

(Tang, et al., 2009)

10.4.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      One has to define the type of the data (e.g.: DNA, SNP, AFLP, Microsat or standard)
    - How are Microsat alleles coded?
      Need to define if the Microsat data are coded as number of repeats, as the length of the PCR fragments or as an arbitrary number.

    - Enter the size of the repeated motif:
      Need to define the size of the repeated motif, so that the number of repeats can be calculated (Microsat alleles have to be saved as number of repeats in the PGD format)

  - Missing value code:
    Specify the code for the missing values (e.g.: -9, -999, ?, etc.)
  - The last column of the input file contains the index of the individual or group:
    Select if the last column contains the indices of the individuals (when clustering of individuals) or of the groups (when clustering of groups of individuals)
  - Include file with coordinates:
    Possibility to add a file with geographic coordinates
  - Are individuals assigned to populations:
    If yes, one can add a file with population names (in individual clustering the index file is also needed)
  - DNA coded as ACGT:
    Need to define the format of the molecular data file (coded as ACGT or as integers)
  - Gene boundaries file:
    Possibility to add a file with gene boundary definitions
  - Enter the integer that codes for the nucleotide:
    Define the integer that codes for a specific nucleotide

- Output:
  - Clustering of individuals or groups of individuals:
    Choose between these two options
  - Save additional file with population/group names:
    Saves a file with the population/group names
  - Save additional file with geographic coordinates:
    Saves a file with the geographic coordinates of individuals or groups (used for spatial clustering analysis)
  - Specify which data type should be included (optional):
    If more than one allowed data type exists, select the data type which should be included in the output file (only one can be analysed per file).
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.5 BATWING

BATWING (updated 2003)

BATWING uses multi-locus haplotype data, model (stepwise mutation model, mutation models for unique event polymorphism, standard coalescent model), distribution specifications and a Markov chain Monte Carlo (MCMC) method based on coalescent theory to generate approximate random samples from the posterior distributions of parameters such as mutation rates, effective population sizes and growth rates, and times of population splitting events. It also generates approximate posterior samples of the entire genealogical tree underlying the sample, including the tree height, which corresponds to the time since the most recent common ancestor. BATWING is intended for within-species data, and not between-species data (Wilson, et al., 2003).

10.5.1 Data type handled

Batwing can deal with haploid SNP and Microsatellite data

10.5.2 BATWING format

- The input file contains one line per haplotype, with one or more spaces separating the alleles at distinct loci
- Everything after a # is ignored (the whole line is ignored if # is the first non-space character)
- The UEP (unique event polymorphism) alleles (SNP) which may be coded by any two single alphanumeric characters (e.g. “0” and “1”, or “A” and “T”) comes first in the line, followed by the microsatellite or STR (Short Tandem Repeat) data
- The data are coded by integer values. Microsatellite data are coded as the number of tandem repeats at that locus
- Missing STR data can be specified using −1
- If the data are drawn from several distinct populations:
  - One can store the population assignments in a location file.
  - The rows of the location file should correspond to the rows of the data file
  - Subpopulation are coded by any positive integers
  - missing location information can be specified using −1

- example (these input file specifies 10 STR loci, no UEP (SNP) loci and a sample size of 6 haplotypes):

```
3 3 2 1 7 8 2 3 10 11
5 5 4 7 9 1 2 3 4 3
2 5 1 3 1 5 6 2 4 4
3 3 1 5 7 8 2 3 11 13
5 5 4 7 9 1 2 3 4 3
1 7 6 2 3 3 2 3 2 1
```

Location file:

```
1
1
2
2
3
-1
```

10.5.3 Links and References

Website: http://www.mas.ncl.ac.uk/~nijw/

Manual: http://www.mas.ncl.ac.uk/~nijw/batwing/batguide.pdf

(Wilson, et al., 2003)

10.5.4 Special PGDSpider input/output questions

- Input:
  - Enter how many SNP loci are defined in the data file:
    The parser needs to know how many columns of the data file contains SNP data
  - Include a file with locations:
    Possibility to add a file with the definition of populations (individuals assigned to populations).

- Output:
  - Save an additional file with population definitions:
    Allows saving a location file with the population definitions
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.6 BCF

BCF (14. May 2011)

BCF, or the binary variant call format, is the binary version of VCF. It keeps the same information in VCF, while much more efficient to process especially for many samples. The relationship between BCF and VCF is similar to that between BAM and SAM.

The conversion process of the format BCF needs the programs Samtools (version 0.1.12/0.1.06) and Bcftools, which can be downloaded from http://samtools.sourceforge.net. The paths to the program files (samtools.exe and bcftools.exe) have to be specified in the “Config” menu under “Options” (see section 5.3.1 PGDSpider menus).

10.6.1 Data type handled

BCF can handle data of following type:

- SNP
- DNA
- UHTS (Ultra High-Throughput Sequencing data)

10.6.2 BCF format

BCF is the binary format of VCF with following file extension: *.bcf

The detailed format description of the BCF format can be found in bcf.tex included in the samtools source code package.

10.6.3 Links and References

Website: http://samtools.sourceforge.net/mpileup.shtml

10.6.4 Special PGDSpider input/output questions

- Input:
    - What is the ploidy of the data:
      Define if the data are haploid or diploid
    - Only import following regions (optional):
      Defines which regions should be imported. Regions should be defined in following format: refSeqName:start:end, multiple regions: separate it with white spaces

- Output:
  - Save an additional file with reference sequences:
    Saves a file with the reference sequences
  - Specify which data type should be included (optional):
    If the input file contains sequence and SNP data, one has to select which should be included in the output file (only sequence or SNP can be analysed per file).

10.7 CONVERT

CONVERT version 1.31 (March 2005)

CONVERT is a user friendly program that facilitates the transfer of co-dominant, diploid genotypic data among commonly used population genetic software packages. CONVERT reads input files in its own “standard” data format and in GENEPOP format. It can convert these formats into the input formats of the following programs: GDA, GENEPOP, ARLEQUIN, POPGENE, MICROSAT, PHYLIP, and STRUCTURE. In addition, CONVERT can produce a summary table of allele frequencies in which private alleles and the sample sizes at each locus are indicated (Glaubitz, 2004).

10.7.1 Data type handled

CONVERT is able to deal with co-dominant, diploid genotypic data of following data type:

- Microsat
- RFLP
- SNP (numeric)
- Standard
- AFLP

10.7.2 CONVERT format

The CONVERT format is defined as follow:

- It is an EXCEL file saved as a tab delimited Text file (*.txt)
- The first line (cell A1) contains a brief description of the data file (title)
- The second line (cell A2) gives the number of populations present in the data file (e.g.: `npops = 2`)
- The third line (cell A3) contains the number of loci (e.g.: `nloci = 7`)
- fourth line: the names of the loci (in order, without spaces, underscores are allowed)
- Each population starts with the line `pop = pop_name`. There must be at least one space between 'pop' and '='. Spaces within population name are not allowed, but an underscores can be used.
- Each individual within a population begins with an individual name. It can be a number, characters like: '/', '=', '?', ':', ';' or ',' are not allowed and there must be no spaces within names.
- After an individual’s name, the diploid genotypes are given:
  - on the same line or wrapped to the next line
  - alleles must be given in numeric form (two alleles for each locus)
  - alleles can be coded as any integer between 1 and 9999
  - Typically the numbers will indicate the size of the allele in base pairs (e.g., for microsats)
  - Missing data are coded as: '?'

- Example:

```
Canucks vs Yanks for the Ice Hockey gold - Salt Lake City 2002
npops = 2
nloci = 4
       SSR01        SSR02         SSR03         SSR04
pop = Canadian_Team
Lemieux      239    245    204    222    169    185    180    206
Sakic        241    247    216    238    195    195    174    198
Fleury       ?      ?      224    224    175    189    160    218
Kariya       229    239    226    240    181    203    182    194
Yzerman      223    239    224    226    191    195    174    180
Lindros      237    245    222    226    179    193    194    222
pop = American_Team

Chelios      235    243    216    226    179    183    172    218
Leetch       235    239    208    216    179    191    198    208
Suter        237    241    222    224    185    197    166    192
Housley      ?      ?      212    228    183    191    184    218
Hatcher      237    241    222    222    173    187    160    198
Amonte       235    245    226    230    181    181    178    192
```

10.7.3 Links and References

Website:
http://www.agriculture.purdue.edu/fnr/html/faculty/Rhodes/Students%20and%20Staff/glaubitz/software.htm

(Glaubitz, 2004)

10.7.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      Allows selecting the type of the data
    - How are Microsat alleles coded:
      Needs to define if the Microsat data are coded as number of repeats or as length of the PCR fragments
    - Enter the size of the repeated motif:
      Needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsatellite data)

10.8 FASTA

FASTA format is a text based format for representing either nucleic acid sequences or peptide sequences, in which base pairs or amino acids are represented using single-letter codes. Sequence names and comments can also be included before the sequences (Pearson, 1990).

10.8.1 Data type handled

The FASTA format can contain nucleic acid or peptide sequences.

10.8.2 FASTA format

- FASTA has no standard file extension. The following extensions are often used: .fa, .mpfa, .fna, .fsa, .fas or .fasta
- The FASTA format begins with a single line description, followed by lines of sequence data. It is recommended that all lines of text be shorter than 80 characters.
- The sequence ends if another line starting with a ”>” appears (this indicates the start of another sequence)

- The header line is arranged as follows:
  - It begins with a ”>”
  - The following word following is the identifier and/or name of the sequence (optional)
  - The rest of the line is the description (optional)
  - There should be no space between the ”>” and the first letter of the identifier
  - The header line may contain more than one header separated by a ^A (Control-A) character
  - Possible sequence identifiers: Many different sequence databases use standardized headers, which helps to automatically extract information from the header:

GenBank	`“gi”|gi-number|”gb”|accession|locus`
EMBL Data Library	`“gi”|gi-number|”emb”|accession|locus`
DDBJ, DNA Database of Japan	`“gi”|gi-number|”dbj”|accession|locus`
General database identifier	`“gnl”|database|identifier`
“simply”	`identifier`

- Sequence representation:
  - The sequences comes after the header line and comments
  - each line of a sequence should have fewer than 80 characters
  - Sequences may be protein sequences or nucleic acid sequences
  - Sequences can contain gaps or alignment characters

  - Sequences are expected to be represented in the standard IUB/IUPAC amino acid and nucleic acid codes, with these exceptions: lower-case letters are accepted and are mapped into upper-case, a single hyphen or dash can be used to represent a gap character and in amino acid sequences: U and * are acceptable letters
  - Numerical digits are not allowed but are used in some databases to indicate the position in the sequence

- simple example of a cytochrome b protein sequence:

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

10.8.3 Links

Wikipedia: http://en.wikipedia.org/wiki/FASTA_format

NCBI's FASTA format description: http://www.ncbi.nlm.nih.gov/blast/fasta.shtml

10.8.4 Special PGDSpider input/output questions

- Input:
  - Select the data type:
    Allows the user to specify if the data are Microsat or SNP data

- Output:
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type that should be included in the output file (only one data type can be analysed per file).
  - If numeric SNP data: Enter the integer that codes for the nucleotide:
    Define the integer that codes for a specific nucleotide
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file. Type “All loci” if you want to write all loci to the FASTA file.

10.9 FASTQ

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are encoded with a single ASCII character. It was originally developed at the Wellcome Trust Sanger Institute to bundle a FASTA sequence and its quality data, but has recently become the de facto standard for storing the output of high throughput sequencing instruments (Cock, et al., 2010).

10.9.1 Data type handled

The FASTQ format contains sequences and their quality scores.

10.9.2 FASTQ format

- FASTQ has no standard file extension. The following extensions are often used: .fastq, .fq or .txt
- A FASTQ file normally uses four lines per sequence:
    - Line 1: begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line)
    - Line 2: is the raw sequence letters (IUPAC ambiguity codes: ACTGNURYSWKMBDHV)
    - Line 3: begins with a '+' character and is optionally followed by the same sequence identifier (and any description).
    - Line 4: encodes the quality values for the sequence in Line 2 and must contain the same number of symbols as letters in the sequence.
- The original Sanger FASTQ files also allowed the sequence and quality strings to be wrapped (split over multiple lines), but this is generally discouraged as it can make parsing complicated due to the unfortunate choice of ”@” and ”+” as markers (these characters can also occur in the quality string).

- simple example of a fastq file:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```

10.9.3 Links

Wikipedia: http://en.wikipedia.org/wiki/FASTQ_format

(Cock, et al., 2010)

10.9.4 Special PGDSpider input/output questions

- Input:
    - Select the quality score encoding:
      Allows the user to specify how the quality scores are encoded (Phred Score + 33 or Phred Score +64)
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file. Type “All loci” if you want to write all loci to the FASTQ file.

10.10 FDist2

FDist2 is a program to detect loci that might be under selection in samples drawn from structured populations (Beaumont and Nichols, 1996; Flint, et al., 1999).

10.10.1 Data type handled

FDist2 can handle Microsat, DNA and Standard (multi-allelic marker) data

10.10.2 FDist2 format

The datacal program (in the FDist2 distribution) can read the following input file:

- In the first line, a 1 or 0 indicate the format of the data matrix: alleles by rows (1) or populations by rows (0).
- The second line gives the number of populations
- Third line: number of loci
- Fourth line: number of alleles at locus 1
- Then the matrix of data at locus 1 follows either with each row corresponding to the same allele or to the same population
- The number of alleles at locus 2 is listed followed by the next data matrix, etc.
- The data matrices can also contain populations for which a locus was not genotyped, these missing data should be indicated by zero entries

- Example:

```
0
5
2

3
4 76 0
45 115 0
109 11 0
120 0 0
0 0 0

3
4 23 53
0 38 122
0 72 48
0 57 63
0 0 0
```

Locus 1

Locus 2

10.10.3 Links and References

Website: http://www.rubic.rdg.ac.uk/~mab/software.html

(Beaumont and Nichols, 1996; Flint, et al., 1999)

10.10.4 Special PGDSpider input/output questions

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.11 FSTAT

FSTAT version 2.9.3.2 (February 2002)

FSTAT estimates and tests gene diversities and differentiation statistics from codominant genetic markers. It computes both Nei and Weir & Cockerham families of estimators of gene diversities and F-statistics, and tests them using randomization methods. Jackknife and bootstrap confidence intervals are also provided (Goudet, 2001).

10.11.1 Data type handled

FSTAT can handle Microsatellite and standard (multi-allelic marker) data

10.11.2 FSTAT format

- FSTAT files have the extension *.dat
- The total number of individuals in the data set needs to be less than 20’000
- The first line contains 4 numbers separated by any number of spaces:
  - The number of samples, np (‹=200)
  - The number of loci, nl (‹=100)
  - The highest number used to label an allele, nu (‹=999)
  - And a 1 if the code for alleles is a one digit number (1-9), a 2 if the code for alleles is a 2 digit number (01-99) or a 3 if the code for alleles is a 3 digit number (001-999)
- The first line is followed by nl (number of loci) lines, each containing the name of a locus, in the order they will appear in the rest of the file
- The line nl+2 contains a series of numbers like: `1 0102 0103 0101 0203 0 0303`
  - The first number identifies the sample to which the individual belongs
  - The second number is the genotype of the individual at the first locus
  - And the third number is the genotype at the second locus and so on
- Missing genotypes are encoded with zeros (0001 or 0100 are not valid formats, because both alleles at a locus have to be known, otherwise the genotype is considered as missing)
- No empty lines are needed between samples
- The number of spaces between genotypes can be anything
- The numbering of the samples need not be sequential
- Samples need not to be ordered
- nu needs to be equal to the largest code given to an allele (even if there are less than nu alleles)

- example (code for alleles is a two digit number):

```
3  5  4  2
loc-1
loc-2
loc-3
loc-4
loc-5
1       0404 0403 0403 0303 0404
1       0404 0404 0403 0303 0404
1       0404 0404 0403 0403 0404
2       0404 0404 0303 0302 0404
2       0404 0303 0404 0403 0404
2       0404 0403 0404 0403 0404
3       0404 0404 0404 0403 0404
3       0404 0404 0404 0404 0404
3       0404 0404 0403 0201 0404
```

- To label the populations an additional label file can be given:
    - It is a text file with the extension *.lab and contains the names (labels) of the populations
    - Each line should contain the name (label) of one sample
    - The samples should appear in the same order as in the *.dat file
    - The labels can be of any length but they will be truncated to six characters in the output files
    - example:

```
Stade de France
Twickenham
Arms Park
```

10.11.3 Links and References

Website: http://www2.unil.ch/popgen/softwares/fstat.htm

(Goudet, 2001)

10.11.4 Special PGDSpider input/output questions

- Input:
    - Include file with labels:
      Possibility to add a file with labels (name the populations)
    - How are the Microsat alleles coded:
      Need to define if the Microsat data are coded as number of repeats, as length of the PCR fragments or as an arbitrary number
    - Enter the size of the repeated motif:
      Information needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsatellite data)

- Output:
    - Safe additional file with labels:
      Allows saving an additional file with the population names
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.12 GDA

GDA version 1.1 (7. January 2002)

GDA allows one to compute linkage and Hardy-Weinberg disequilibrium, some genetic distances, and provides method-of-moments estimators for hierarchical F-statistics (Lewis, 2001).

10.12.1 Data type handled

GDA deals with Microsatellite, RFLP, AFLP, SNP and Standard (multi-allelic marker) data

10.12.2 GDA format

GDA uses the NEXUS format (also have a look at the NEXUS file format description) and allows the definition of a GDADATA block:

If a token (a word or a name) begins with a single or double quote character, then every character until the next, matching quote character will be treated as a single token. This is useful for putting blank spaces inside population or locus labels. The commands are not case-sensitive, except in the matrix command (allele named A is treated as being distinct form a). The following commands exist:

- begin
- dimensions:
    - number of populations: `npops=2`
    - number of loci: `nloci=5`
- format:
    - `tokens / notokens`
    - `labels / nolables`
    - `interleaved`
    - `haploid`
    - `missing = ?`
    - `separator=/`
    - `datapoint=standard`
- locusallelelabels (optional):
    - contains the loci names
    - allele names can be provided
    - loci will be numbered beginning with 1 if this command is absent
- matrix:
    - end of the data for one population is signed by either a comma or the semicolon indicating the end of the matrix command
- end

Haploid data can be described in two ways: First, if all loci are haploid one can include the keyword “haploid” in the format command. And second, if a mixture of haploid and diploid data exists one can use the command “hapset” to specify which loci are haploid.

Examples:

- diploid data:

```
#nexus
begin gdadata; [comments are surrounded by square brackets]
  dimensions npops=2 nloci=3;
  format missing=? seperator=/;
  locusallelelabels
    1 'pgi 1',
    2 'pgi 2',
    3 adh / slow fast
  ;
  matrix
    Embudo:
      indiv_1 A/A 100/100 slow/fast
      indiv_2 A/A 75 / 90 slow/slow
      indiv_3 A/a 75/100  fast/Slow
      indiv_4 A/A 100/100 fast/fast,
    Black_Mesa:
      1 a/a 110/100 fast/slow
      2 a/A  75/100 slow/slow
      3 a/a 100/100 fast/fast
  ;
end;
```

- haploid data:

```
#nexus
begin gdadata;
       dimensions nloci=5 npops=2;
       format haploid tokens missing=? datapoint=standard;
       locusallelelabels
             1 locus_1 [ / 1 2 3 4],
             2 locus_2 [ / 1 2 3 4],
             3 locus_3 [ / 1 2 3 4],
             4 locus_4 [ / 1 2 3 4],
             5 locus_5 [ / 1 2 3 4];
       matrix
       Pop_1: _1_ 4 3 3 3 4
             _2_ 4 4 3 3 4
             _3_ 4 4 3 3 4
             _4_ 4 4 ? 3 4
             _5_ 4 4 2 4 4,
       Pop_2: _1_ 4 4 2 2 4
             _2_ 4 3 4 3 4
             _3_ 4 4 3 3 4
             _4_ 4 4 4 4 4
             _5_ 4 3 4 4 4
;
end;
```

- mixed haploid/diploid data:

```
#NEXUS
[Note: first 2 loci are diploid and last 3 are haploid]
begin gdadata;
  dimensions nloci=5 npops=6;
  format tokens labels missing=? datapoint=standard;
  hapset 3-5;
  locusallelelabels
    1 'dip 1',
    2 'dip 2',
    3 'hap 1',
    4 'hap 2',
    5 'hap 3'
  ;
  matrix
    Pop1:
      indiv1 4/4 4/3 3 3 4
      indiv2 4/4 4/4 3 3 4
      ...
```

10.12.3 Links and References

Website: http://hydrodictyon.eeb.uconn.edu/people/plewis/software.php

(Lewis, 2001)

10.12.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      Needs to specify the type of the data
    - How are Microsat alleles coded:
      Needs to define if the Microsat data are coded as number of repeats or as length of the PCR fragments
    - Enter the size of the repeated motif:
      Information needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsatellite data)

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.13 GENELAND

GENELAND (12. April 2011)

GENELAND is a computer program whose main goal is to process individual multilocus genetic data to detect population structure, i.e sub-populations at (or close to) Hardy-Weinberg and linkage equilibrium. Although the concept of population refers here to genetic structure only, it is often realistic to assume that populations are spatially organised. Toward this aim, GENELAND is based on a spatially explicit model that can make use of both geographic and genetic informations to estimate the number of populations in a dataset and delineate their spatial organisation. Important areas of application include landscape genetics, conservation genetics, human genetics and epidemiology (Guillot, 2008; Guillot, et al., 2005; Guillot, et al., 2005; Guillot and Santos, 2009; Guillot and Santos, 2010; Guillot, et al., 2008).

GENELAND is released as an add-on to the free statistical program R and is currently available for Linux, Mac-OS and Windows. It includes a fully clickable user interface requiring no particular knowledge of R.

10.13.1 Data type handled

GENEPOP handles Microsatellite, SNP, AFLP, Standard (multi-allelic markers) and DNA data types.

10.13.2 GENELAND format

- Genotypes file: contains the genotypes of n haploid or diploid individuals at L co-dominant markers
- Coordinates file (optional): contains the spatial coordinates representative of each individual.

Genotypes file

Assuming that you have data for n individuals genotyped at L loci, the data must be arranged in:

- a plain ascii file
- without any extra invisible characters (like in MS-Word .doc files)
- with one line per individual
- each allele must be coded by an integer
- the number of digits of each field is arbitrary and can vary across loci
- extra header lines are not allowed
- missing data are allowed and can be coded by any arbitrary character string (e.g. 000, 00, NA or -999). By default, it is assumed that missing data are coded as NA.
- for haploid organisms with L loci, the genotype file must have L columns.

Diploid data:

Codominant data (SNP or Microsat):

- one line and 2x L columns per individual
- Example (2 individuals with 10 loci, missing data are coded as 000):

```
198 000 358 362 141 141 179 000 208 224 243 243 278 284 86 88 120 124 238 244
200 202 000 358 141 141 183 183 218 224 237 243 276 278 88 88 120 124 240 244
```

Dominant data (AFLP):

- one line and L columns per individual
- absence/presence of the allele is coded as 0/1
- Example (2 individuals with 10 loci, missing data are coded as 000):

```
0 1 1 1 0 1 0 0 0
1 0 0 1 1 0 0 1 000
```

Haploid data:

- for Microsat, SNP or mtDNA data
- one line and L columns per individual
- Example (2 individuals with 10 loci, missing data are coded as 000):

```
198 000 358 362 141 141 179 000 208 224
200 202 000 358 141 141 183 183 218 224
```

Coordinates file

- one line per individual and two columns (x-axis and y-axis coordinate)
- the units do not matter
- coordinates are planar coordinates such as UTM coordinates. Coordinates given as spherical coordinates will be interpreted as planar coordinates.
- extra header lines are not allowed
- missing data are not allowed. If some coordinates are missing, you can either substitute an estimated value or do the analysis without spatial coordinates at all using the non-spatial model.
- Example (2 individuals):

```
25.6 745.2
54.1 827.8
```

10.13.3 Links and References

Website: http://www2.imm.dtu.dk/~gigu/Geneland/,

Manual: http://www2.imm.dtu.dk/~gigu/Geneland/Geneland-Doc.pdf

(Guillot, 2008; Guillot, et al., 2005; Guillot, et al., 2005; Guillot and Santos, 2009; Guillot and Santos, 2010; Guillot, et al., 2008)

10.13.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      One has to define the type of the data (e.g.: SNP, AFLP, Microsat , Standard or DNA)
    - How are Microsat alleles coded?
      Need to define if the Microsat data are coded as number of repeats, as the length of the PCR fragments or as an arbitrary number.
    - Enter the size of the repeated motif:
      Need to define the size of the repeated motif, so that the number of repeats can be calculated (Microsat alleles have to be saved as number of repeats in the PGD format)
    - Missing value code:
      Specify the code for the missing values (e.g.: -9, -999, ?, etc.)
    - Include file with coordinates:
      Possibility to add a file with geographic coordinates
    - Enter the integer that codes for the nucleotide:
      Define the integer that codes for a specific nucleotide

- Output:
    - Save additional file with geographic coordinates:
      Saves a file with the geographic coordinates of individuals or groups (used for spatial clustering analysis)
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.14 GENEPOP

GENEPOP version 4.1 (24. March 2011)

GENEPOP allows one to compute exact tests for Hardy-Weinberg equilibrium, for population differentiation and for genotypic disequilibrium among pairs of loci. It also computes estimations of F-statistics, null allele frequencies, allele size based statistics for microsatellites, etc. It performs analyses of isolation by distance from pairwise comparisons of individuals or population samples, including confidence intervals for “neighbourhood size” (Raymond and Rousset, 1995). This format is also used by many other population genetics programs (BAPS, FSTAT, ARLEQUIN, GENETIX, etc).

10.14.1 Data type handled

GENEPOP handles haploid and diploid data of Microsatellite and Standard (multi-allelic markers) data type.

10.14.2 GENEPOP format

- GENEPOP accepts input file names either with the extension *.txt or without any extension, but the input files have to be ASCII text files
- The first line can contain anything. It can be used to store information about the data
- The locus names may be given next, one per line or on the same line but separated by commas
- Then the population sample indicator “Pop” follows (capitalization does not matter). Each sample from a different geographical origin is declared by a line with a pop statement.
- Information for the first individual:
    - `ind#001 fem ,0101 0202 0000 0410`
    - Here “ind#001 fem” is an identifier. It is possible to use any character (except a comma!). The last identifier of every sub-population is used as the sample name in the output files. The comma between the identifier and the list of genotypes is required.
    - “0101” indicates that this individual is homozygous for the 01 allele at the first locus.
    - The third locus (0000) contains missing data
    - At the fourth locus, the genotype is 0410, which indicates the presence of alleles 04 and 10.
    - Alleles are numbered from 01 to 99 or 001 to 999 if needed. 2-digits and 3-digits coding of alleles can be intermixed (among loci, not within loci).
    - Haploid and diploid data can be intermixed. (6-digits genotypes are recognized as 3-digits diploid genotypes; 4-digits genotypes are recognized as 2-digits diploid genotypes; 2- and 3-digits genotypes are recognized as haploid genotypes. The same coding should be used consistently within each locus (for haplo-diploid data haploid data should be coded as diploid data with one unknown allele).)
    - Genotypes can extend on more than one line

- Each additional individual information starts on a new line, and may extend over several lines (but it is not allowed to start a new line in the middle of a locus genotype)
- Additional samples begin with a “Pop” statement on a new line
- There is no constraint on the number of blanks separating the various fields, but blank lines at the end of the file are not allowed
- Missing data should be indicated with 00 (or 000 for 3-digits coding) and not with blanks
- The number of locus names should correspond to the number of genotypes in each individual

- Example:

```
Title line: "Grape populations in southern France"
ADH Locus 1
ADH #2
ADH three
Pop
Grange des Peres , 0201 003003 01
Grange des Peres , 0202 003001 01
Grange des Peres , 0102 004001 01
Grange des Peres , 0103 002002 01
Grange des Peres , 0203 002004 01
POP
Tertre Roteboeuf , 0102 002002 01
Tertre Roteboeuf , 0102 002001 01
Tertre Roteboeuf , 0201 002003 01
Tertre Roteboeuf , 0201 003003 01
Tertre Roteboeuf , 0101 002001 01
Pop
, 0000 002001 01
, 0200 002001 01
, 0010 002001
01
last pop, 0101 002001 02
```

10.14.3 Links and References

Website: http://kimura.univ-montp2.fr/~rousset/Genepop.htm,

Manual: http://kimura.univ-montp2.fr/~rousset/Genepop.pdf

Input file: http://genepop.curtin.edu.au/help_input.html

(Rousset, 2008)

10.14.4 Special PGDSpider input/output questions

- Input:
    - How are the Microsat alleles coded:
      Need to define if the Microsat data are coded as number of repeats, as length of the PCR fragments, or as an arbitrary number

  - Enter the size of the repeated motif:
    Information needed to convert the Microsatellite data (length of the PCR fragments) into number of repeat data (PGD can only save number of repeat Microsatellite data)
- Output:
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.15 GENETIX

GENETIX version 4.05 (5. May 2004)

This set of programs computes several basic parameters of population genetics such as Nei's D and H, Wright's F-statistics and linkage disequilibrium D. For each of them, the distribution of the parameter values under the null hypothesis (for instance Hardy-Weinberg equilibrium for Fstats) is generated by the appropriate resampling scheme of the relevant objects (e.g. alleles between individuals in the case of Fis) using permutations. The permutation-based statistical inference procedures implemented in GENETIX represent an alternative to bootstrapping and jack-knifing, or to exact probability tests when available (Raymond and Rousset, 1995). In addition, an adaptation of Mantel's test for the correlation between distance matrices is available. A correspondence analysis program adapted to handle individual diploid genotypes, with tridimensional graphics, is also implemented (Belkhir, 1996-2004).

10.15.1 Data type handled

GENETIX deals with following diploid data types:

- Microsat
- RFLP
- AFLP
- SNP (numeric)
- Standard

10.15.2 GENETIX format

- The GENETIX file format has the extension *.gtx and must be an ASCII file
- The text separators can be blanks, tabulators, or other characters that need to be specified
- The first line contains the number of loci
- Second line: the number of populations
- Third line: the name of the first locus with maximal 5 characters length
- Fourth line: the number of alleles followed by a list of alleles coded with 3 numbers
- Fifth line: name of second locus
- …
- n. line: name of the first population (only 15 characters are taken)
- n+1. line: number of samples (individuals)
- n+2. line: identifier of the individual with a length of 10 characters followed by its genotype (The loci have the same order as in the list above (6 numbers per locus, because data have to be diploid)
- …
- m. line: name of the second population
- etc.

- Genotypes are coded with 6 numbers. The first 3 numbers stands for the first allele and the rest of the numbers for the second allele. The smaller allele has to come first!
- Haploid data have to be coded as homozygous diploids
- Missing values are coded as 000000. If one allele is unknown then the whole genotype must be coded as missing

- example:

```
4
2
aat1
1 120
aat2
3 100 132 146
adh
2 100 123
est1
4 100 107 110 115
Population "i"
3
i001       120120 132132 100100 107110
i002       120120 132132 100100 107110
i003       120120 100132 100100 110110
Population "j"
2
j001      120120 132132 100100 107107
j002      120120 132132 100123 107107
```

10.15.3 Links and References

Website: http://www.genetix.univ-montp2.fr/genetix/genetix.htm

(Belkhir, 1996-2004)

10.15.4 Special PGDSpider input/output questions

- Input:
  - Select the data type:
    Allows specifying the type of the data
  - How are Microsat alleles coded:
    Need to define if the Microsat data are coded as number of repeats or as length of the PCR fragments
  - Enter the size of the repeated motif:
    Needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsatellite data)

- Output:
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

- Specify the locus/locus combination which should be written to the output file (optional):
  If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.16 GESTE / BayeScan

BayeScan version 2.01 (December 2010)**:**
This program identifies candidate loci under natural selection. It's applicable to both, dominant and codominant data.

GESTE version 2.0:
"GEnetic STructure inference based on genetic and Environmental data" is a Bayesian method to evaluate the effect that biotic and abiotic environmental factors (geographic distance, language, temperature, altitude, local population sizes, etc.) have on the genetic structure of populations. It can also be used to study spatial population processes, such as range expansions, by simply introducing longitude and latitude as the explanatory variables.
GESTE estimates FST values for each local population and relates them to environmental factors using a generalized linear model. The method requires genetic data from codominant markers (e.g. allozymes, microsatellites, or SNPs) and environmental data specific to each local population. The software is written in C++ and integrates a tool to draw posterior density functions (histogram, running mean, traces, etc.) and to estimate parameters from them (mean, mode, variance, HPDI etc.).

10.16.1 Data type handled

GESTE / BayeScan is able to deal with following data types:

- AFLP
- SNP
- Microsatellites
- Allozymes

10.16.2 GESTE / BayeScan format

- The program recognizes keywords in […].
- The number of loci (keyword: [loci]) and populations (keyword: [populaitons]) must be indicated before the main data.
- For each population (keyword: [pop]), there is one line per locus numbered from 1 to the number of loci.
- Population must be numbered from 1 to the number of populations.
- Then there is the number of alleles measured for this population at this locus (50 individuals make 100 alleles for diploids) and the number of possible alleles found at this locus (for all populations).
- After, there is the corresponding allele count. This part must sum to the number of alleles measured.
- Number of individuals can be different at every locus (missing data).
- Comments can be written between sections.

- There is no particular file extension needed.

- Example:

```
[loci]=5

[populations]=2

[pop]=1
  1  100   7   34    0    4    0  13  20  29
  2  100   7    3   26    2    8  56   2   3
  3  100   7   46    0   17    1  10  25   1
  4  100   7    4    7    2   52  23  12   0
  5  100   7   23   28    0    1   2  38   8

[pop]=2
  1  100   7   11    6   17    2   8  36  20
  2  100   7    8    6    3   26  36   9  12
  3  100   7   11    7   35   13  26   8   0
  4  100   7   14    2    0   24  36  24   0
  5  100   7   20    6   19   36   6  10   3
```

10.16.3 Links and References

GESTE:

- Website: http://www-leca.ujf-grenoble.fr/logiciels.htm
- (Foll and Gaggiotti, 2006)

BayeScan:

- Website: http://cmpg.unibe.ch/software/bayescan/index.html
- (Fischer, et al., 2011; Foll, et al., 2010; Foll and Gaggiotti, 2008)

10.16.4 Special PGDSpider input/output questions

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.17 HGDP-CEPH

The HGDP-CEPH Human Genome Diversity Cell Line Panel is a widely used resource for studies of human genetic variation. The HGDP-CEPH Human Genome Diversity Cell Line Panel (henceforth the “HGDP panel”) is a collection of 1064 DNA samples from individuals distributed around the world (Cann, et al., 2002).The DNA samples in the HGDP panel are publicly available for studies of genetic variation, and they now form the basis for a sizeable body of human genetics research (Cavalli-Sforza, 2005).

The HGDP Database is designed to receive and store the polymorphic marker genotypes generated by users of the DNAs of the HGDP-CEPH Diversity Panel. The data are accessible publically via a web interface (database V2.0 only) and/or as flat files. In addition to genotypes, the database includes information on the geographic and population origin, and on the gender of each of the participating volunteers, who are identified by code numbers only (HGDP identifiers).

10.17.1 HGDP-CEPH export formats

The HGDP data can be exported into two different formats: The LINKAGE-like and the ARLEQUIN format (see the ARLEQUIN format description) with an additional log file. The PGDSpider can only read in the ARLEQUIN format with the additional log file.

The log file looks like the following example:

```
#HGDP database V2.0 ; 2008/04/09 12:04:24
#Dump format : Arlequin ; Filename : 20080409_120424_report.log
#Selected populations : Karitiana, Surui, Colombians, Maya, Pima, Cambodians
identifier   dbsnp_id     chrom  physical_pos MAF_Europe   HetZ_Europe
rs6696404    rs6696404    1      3015090       0.0268       0.0537
rs760567     rs760567     1      3023622       0.1946       0.2819
rs2993491    rs2993491    1      3034767       0.2349       0.2953
rs2817172    rs2817172    1      3064676       0.3926       0.4631
```

10.17.2 Links

Website: http://www.cephb.fr/en/hgdp/

10.17.3 Special PGDSpider input/output questions

- None

10.18 Immanc and BayesAss

Immanc version 5.0 (8 October 1998)

Detecting Immigrants Using Multilocus Genotypes (Rannala and Mountain, 1997).

BayesAss+ version 1.3 (4 May 2005)

Bayesian Estimation of Recent Migration Rates Using Multilocus Genotypes (Wilson and Rannala, 2003).

10.18.1 Data type handled

Immanc/ BayesAss can handle diploid data of following data types:

- microsatellites
- RFLPs
- SNPs
- Standard
- allozymes

10.18.2 Immanc format

- The Immanc file should have the suffix of *.inp or *.txt
- The Columns can be separated by any whitespace
- Missing genotypes should be represented by 0, 00, or 000
- Blanks are not allowed
- The first column contains the individual labels
- The second column contains the population labels
- The third column contains the locus labels
- The remaining 2 columns contain the alleles at each locus that make up a genotype
- Spaces within names are not allowed
- Each individual has a row entry for every locus
- The order of the alleles determines the haplotype phase (this information is not currently used and so the ordering is arbitrary)
- The setup for the data file should be:

```
Individual1     population1     locus1     allele1     allele2
Individual2     population1     locus1     allele1     allele2
```

- Example (data for 15 individuals from two populations, genotyped for two loci)

```
ind1    pop1    locA    194     198
ind2    pop1    locA    198     198
ind3    pop1    locA    192     198
ind4    pop2    locA    184     194
ind5    pop2    locA    190     194
ind6    pop2    locA    184     194
ind7    pop2    locA    192     194
ind8    pop2    locA    184     194
ind1    pop1    locB    158     162
ind2    pop1    locB    148     162
ind3    pop1    locB    150     158
ind5    pop2    locB    150     162
ind6    pop2    locB    158     162
ind7    pop2    locB    152     156
ind8    pop2    locB    156     158
```

10.18.3 Links and References

Website: http://www.rannala.org/?page_id=13

Manual: http://www.rannala.org/docs/immanc.html

(Rannala and Mountain, 1997)

(Wilson and Rannala, 2003).

10.18.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      Allows specifying the type of the data
    - How are Microsat alleles coded:
      Need to define if the Microsat data are coded as number of repeats, as length of the PCR fragments or as an arbitrary number
    - Enter the size of the repeated motif:
      Needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsatellite data)

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.19 IM/IMa

IM/IMa (17. December 2009)

IM is a program estimating the parameters of an isolation model with migration from haplotype data drawn from two closely related species or populations. A relatively large numbers of loci can be studied simultaneously, and different mutation models can be used. IM estimates the divergence time and the migrations having occurred in the ancestry of two populations, which might have grown exponentially since they split (Hey and Nielsen, 2004; Nielsen and Wakeley, 2001).

IMa allows log likelihood ratio tests of nested demographic models to be performed. IMa is faster and better than IM (i.e. by virtue of providing access to the joint posterior density function), and it can be used for most (but not all) of the situations and options that IM can be used for (Hey and Nielsen, 2007).

10.19.1 Data type handled

IM can handle DNA and Microsatellite (STR) data.

10.19.2 IM/IMa format

- line 1 contains arbitrary text, usually explaining the content of the file
- After line 1, comments can be included to provide explanatory information. Each line of comment must begin with a '#'.
- Line 2 (or line after comments): two population names, for populations 1 and 2 respectively, separated by one or more spaces
- Line 3: the number of loci in the data set (integer)
- Line 4: basic information for locus 1. This line contains at least five items separated by one or more spaces
  1. The locus name (no spaces within the name)
  2. The sample size for population 1 (n1)
  3. The sample size for population 2 (n2)
  4. The length of the sequence
  5. A letter indicating the mutation model
     - I: Infinite Sites (IS) model (Kimura, 1969). Under this model every mutation that has occurred in the history of a sample of sequences occurs at a different place in a DNA sequence.
     - H: Hasegawa-Kishino-Yano (HKY) model (Hasegawa, et al., 1985) was applied to the Isolation with Migration model by Palsboll et al. (Palsboll, et al., 2004). It is a general model that allows for multiple substitution, with different rates of transitions and transversions as well as unequal frequencies of the four nucleotides.
     - S: Stepwise Mutation Model (SMM) (Kimura and Ohta, 1978). This model can be applied to allelic variation in which each mutation causes an allele to increase or decrease by one step on whatever scale the alleles are being measured.

    - J: joint SSM and IS model. This model is named HapSTR
    - If this letter is not included on this line, the default is IS. If SSM (S) or HapSTR (J), the letter is followed immediately (no spaces) by the number of linked STR markers within the locus.
  6. Inheritance scalar (e.g.: 1 for autosome, 0.75 for X-linked, 0.25 for Y-linked or mtDNA)
  7. The mutation rate per year for the locus (not per base pair). This can be left blank, but is needed for estimating parameters on demographic scales. If there are multiple STRs in the locus then there can be multiple mutation rates on this line separated by spaces. If the locus is a HapSTR, then the first mutation rate given applies to the sequence portion of the locus with subsequent values corresponding to STR markers included in the locus.
  8. If the mutation rate is given, it can be followed by a range of mutation rates that can be used (with ranges for other loci in the analysis) to set priors on the ratios of mutation rate scalars. The range is entered with open parentheses, the lowest value, a comma, the highest value, and closed parentheses (e.g. '(0.00001, 0.00004)'. The range must bracket the rate. For a locus with multiple mutation rates, and multiple ranges, each range follows its corresponding mutation rate immediately on line.
- Line 5: data for first gene copy from population 1:
  - The first 10 spaces are devoted to the sample name.
  - The sequence or allele length (for SSM model) begins in column 11 of the file. The sequence for a given sample is given all on one line without gaps
  - For SSM or HapSTR data, the allele length assumes a step size of 1. This means that data from STRs that are multiples of lengths greater than 1 must be converted to counts of the number of base repeats.
  - If the data is for an SSM model locus and there are multiple STRs, then there will be one integer on each line for each STR, separated by a space.
  - If the locus is HapSTR (joint IS and SSM) then the STR data is given on the line, beginning at column 11, followed by the sequence data.
  - For SSM data, as for other types of data, only one gene copy is represented on each line of the data file. Diploid genotype data must be broken up and listed, with one data line for each gene copy.
- Line 6 through line (n1+n2 +4): the remainder of the data for locus 1. Each line contains the data for one sample. The data for population 1 samples are given in lines 5 through line (n1 + 4). The data for population 2 begins on line (n1+5) and proceeds to line (n1+n2+4).
- Additional lines for additional loci: If there is more than one locus, then the data for locus 2 begins on line (n1+n2+5) with a line similar to line 4 presenting the basic information for locus 2. The sample names and sample sizes for locus 2 and the inheritance scalars and mutation model for locus 2 does not need to be the same as for locus 1
- The last line should end with a newline so that the file ends on a blank line

- Example (a tiny three locus data set. The mutation rate per year is known and specified for locus 1, but not for loci 2 and 3)

```
Example data for IM
# im test data
population1 population2
3
locus1 1 1 13 I 1 0.0000000008 (0.0000000001, 0.0000000015)
pop1_1    ACTACTGTCATGA
pop2_1    AGTACTATCACGA
hapstrexample 2 1 4 J2 0.75
pop1_1    13 34 GTAC
pop1_2    12 35 GTAT
pop2_1    12 37 GTAT
strexample 2 2 1 S1 1 0.00001 (0.000001, 0.00005)
strpop11a 23
strpop11b 26
strpop21a 25
strpop21b 31
```

10.19.3 Links and References

Website: http://genfaculty.rutgers.edu/hey/software

Manual: http://lifesci.rutgers.edu/~heylab/ProgramsandData/Programs/IM/Using_IM_3_5_2007.pdf

(Hey and Nielsen, 2004; Nielsen and Wakeley, 2001)

(Hey and Nielsen, 2007)

10.19.4 Special PGDSpider input/output questions

- Input: none

- Output:
    - Select first population:
      Need to define the first population, because only two populations can be included
    - Select second population:
      Need to define the second population, because only two populations can be included
    - Are the loci linked:
      Need to define if the specified loci are linked or not
    - Select the inheritance scalar of the loci:
      Need to give the inheritance scalar of the specified loci. One can choose between 1 (autosome), 0.75 (X-linked) and 0.25 (Y-linked or mtDNA)

10.20 KML

KML version 2.2

KML is a file format used to display geographic data in an Earth browser such as Google Earth, Google Maps, and Google Maps for mobile. KML uses a tag-based structure with nested elements and attributes and is based on the XML standard (Google, 2009).

10.20.1 Data type handled

KML is able to handle geographic data (coordinates).

10.20.2 KML format

A Placemark is one of the most commonly used features in Google Earth. It marks a position on the Earth's surface, using a yellow pushpin as the icon. The simplest Placemark includes only a <Point> element, which specifies the location of the Placemark. One can specify a name and a custom icon for the Placemark, and one can also add other geometry elements to it. There exist three different types of placemark: simple, floating, and extruded.

The structure of a KML file breaks down as follows:

- An XML header: This is line 1 in every KML file. No spaces or other characters can appear before this line: `<?xml version=”1.0” encoding=”UTF-8”?>`
- A KML namespace declaration and root element. This is line 2 in every KML 2.2 file: `<kml xmlns=http://www.opengis.net/kml/2.2>`
- A element named “Document” which surrounds all other elements
- A Style element:
    - “id” attribute giving the name of the style
    - “IconStyle” element containing the “Icon” element, which contains the “href” element with the URL to the icon picture
- A Placemark element that contains the following elements:
    - “name” element used as the label for the Placemark
    - “styleUrl” element giving the name of the used style
    - “description” element containing a description that appears in the “balloon” attached to the Placemark
    - “Point” element that contains the “coordinates” element specifying the position of the Placemark on the Earth's surface (longitude, latitude, and optional altitude)

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>

    <Style id="blue">
      <IconStyle>
        <Icon>
          <href>http://maps.google.com/mapfiles/ms/icons/blue.png</href>
        </Icon>
      </IconStyle>
    </Style>

    <Placemark>
      <name>IconStyle.kml</name>
      <styleUrl>#blue</styleUrl>
      <Point>
        <coordinates>-122.36868,37.831145,0</coordinates>
      </Point>
    </Placemark>

    <Style id="blue-dot">
      <IconStyle>
        <Icon>
          <href>http://maps.google.com/mapfiles/ms/icons/blue-dot.png</href>
        </Icon>
      </IconStyle>
    </Style>

    <Placemark>
      <name>IconStyle.kml</name>
      <styleUrl>#blue-dot</styleUrl>
      <Point>
        <coordinates>-123.36868,37.831145,0</coordinates>
      </Point>
    </Placemark>

  </Document>
</kml>
```

10.20.3 Links and References

Website: http://code.google.com/intl/de-CH/apis/kml/documentation/kml_tut.html

10.20.4 Special PGDSpider input/output questions

- Input: none

- Output: none

10.21 MEGA

MEGA version 5 (26. April 2011)

MEGA is an integrated tool for conducting automatic or manual sequence alignment, inferring phylogenetic trees, mining web-based databases, estimating rates of molecular evolution, and testing evolutionary hypotheses (Tamura, et al., 2007).

10.21.1 Data type handled

MEGA is able to handle following data types:

- DNA
- RNA
- nucleotide
- distance
- protein sequences

10.21.2 MEGA format

- The MEGA files have the extension *.meg and are ASCII text files
- The first line need to contain the keyword #MEGA
- The second line of the data file may contain a description of the data. The Title statement is written according to a set of rules:
    - always begins with `!Title` and ends with a semicolon (“;”)
    - do not occupy more than one line of text
    - a semicolon inside the statement is not allowed
    - example:

```
#mega
!Title This is an example title;
```

- The third line contains the description statement:
    - Gives detailed information on the data file
    - always begins with `!Description` and ends with a semicolon (“;”)
    - may occupy multiple lines
    - a semicolon inside the statement is not allowed
    - example:

```
#mega
!Title This is an example title;
!Description This is detailed information the data file;
```

- The format statement includes the information on the data type present in the file and some of its attributes:
    - written after the Title and the Description statement
    - contains one or more command statements. This command statements contain a predefined command, followed by an equal sign and a valid setting keyword (`command=keyword`).
- Comments can be anywhere in the data file, can span multiple lines, are enclosed in square brackets ([and]) and can be nested
- Keyword can be written in any combination of lower- and uppercase letters
- Taxa Names:
    - Every label must be written on a new line, and a '#' sign must precede the label
    - There are no restrictions on the length of the labels
    - The labels are not required to be unique (although identical labels may result in ambiguities and should be avoided)
    - Labels must start with alphanumeric characters (0-9, a-z, and A-Z) or a special character: `-`, `+` or `.`
    - After the first character, taxa labels may contain the following additional special characters: `_`, `*`, `:`, `( )`, `|`, `\`, `/`
    - For multiple word labels, an underscore can be used to represent a blank space

Sequence Input Data:

- Need to consist of two or more aligned sequences of equal length
- Sequences are written in the IUPAC single-letter code in any combination of upper- and lowercase letters
- Spaces and tabs are ignored
- Generally used special symbols : “.” for identical sites, “-“ for alignment gaps and “?” for missing data
- Keywords for the Format Statement:

Command	Setting	Remark	Example
DataType	DNA, RNA, nucleotide, protein		DataType=DNA
NSeqs	integer	Number of sequences	NSeqs=85
NTaxa	integer	Synonymous with NSeqs	NTaxa=85
NSites	integer	Number of nucleotides	Nsites=4592
Property	Exon, Intron, Coding, Noncoding, and End	Specifies whether a domain is protein coding. Exon and Coding are synonymous, as are Intron and Noncoding. End specifies that the domain with the given name ends at this point	Property=cyt_b
Indel	single character	dash (-) to identify insertion/deletions	Indel = -

Identical	single character	use period (.) to show identity with the first sequence	Identical = .
MatchChar	single character	Synonymous with the identical keyword	MatchChar = .
Missing	single character	use question mark (?) to indicate missing data	Missing = ?
CodeTable	A name	This instruction gives the name of the code table for the protein coding domains of the data	CodeTable = Standard

Tab. 11: table with the keywords of the format statement

- Defining Genes and Domains:
  Attributes of different sites (and groups of sites, termed domains) are specified within the data "on the spot" rather than in an attributes block before or after the actual data.

Command	Setting	Remark	Example
Domain	A name	defines a domain with the given name	Domain=first_exon
Gene	A name	defines a gene with the given name	Gene=cytb
Property	Exon, Intron, Coding, Noncoding, and End	specifies the protein-coding attribute for a domain	Property=cytb
CodonStart	A number	specifies the site where the next 1st-codon position will be found in a protein-coding domain	CodonStart=2

Tab. 12: table with the keywords of the attributes of the different sites

- Defining Groups:
  - Assign different taxa to groups in a sequence as well as to distance data files.
  - The name of the group is written in a set of curly brackets ({ }) following the taxa name. The group name can be attached to the taxa name using an underscore or just can be appended.
  - There should be no spaces between the taxa name and the group name
- Labelling Individual Sites:
  - The individual sites in nucleotide or amino acid data can be labelled to construct non-contiguous sets of sites.
  - Each site can be associated with only one label
  - A label can be a letter or a number.

- Example:

```
!Gene=FirstGene Domain=Exon1 Property=Coding;
#Human_{Mammal} ATGGTTTCTAGTCAGGTCACCATGATAGGTCTCAAT
#Mouse_{Mammal} ATGGTTTCTAGTCAGGTCACCATGATAGGTCCCAAT
#Chicken_(Waterston, et al.)
ATGGTTTCTAGTCAGCTCACCATGATAGGTCTCAAT

!Gene=SecondGene Domain=Intron Property=Noncoding;
#Human ATTCCCAGGGAATTCCCGGGGGGTTTAAGGCCCCTTTAAAGAAAGAT
#Mouse GTAGCGCGCGTCGTCAGAGCTCCCAAGGGTAGCAGTCACAGAAAGAT
```

Distance Input Data

- Must be a lower-left or an upper-right triangular matrix
- After writing the #mega, !Title, !Description and !Format commands (some of which are optional), one need to write all the taxa names
- Taxa names are followed by the distance matrix
- Keywords for Format Statement are:

Command	Setting	Remark	Example
DataType	Distance	Specifies that the distance data is in the file	DataType=distance
NSeqs	integer	Number of sequences	NSeqs=85
NTaxa	integer	Same as NSeqs	NTaxa=85
DataFormat	Lowerleft, upperright	Specifies whether the data is in lower left triangular matrix or the upper right triangular matrix	DataFormat=lowerleft

Tab. 13: table with the keywords of the format statement

- Defining Groups: see above
- Example:

```
#mega
!Title: Concatenated Files;
!Format DataType=Distance
DataFormat=LowerLeft NTaxa=6;

#Rodent
#Primate
#Lagomorpha
#Artiodactyla
#Carnivora
#Perissodactyla

0.514
0.535 0.436
0.530 0.388 0.418
0.521 0.353 0.417 0.345
0.500 0.331 0.402 0.327 0.349
```

10.21.3 Links and References

Website: http://www.megasoftware.net/

Manual: http://www.megasoftware.net/manual.pdf

(Tamura, et al., 2011)

10.21.4 Special PGDSpider input/output questions

- Input:
    - Select the data type:
      Allows specifying the type of the data
- Output:
    - Select the kind of data to print:
      Need to define to write a file with sequence data or distance matrix
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.22 MIGRATE

MIGRATE version 3.2.6 (13. October 2010)

MIGRATE estimates effective population sizes and past migration rates between populations, assuming a migration matrix model with asymmetric migration rates and different subpopulation sizes. It uses a coalescent theory approach taking into account the history of mutations and the uncertainty of the genealogy. The estimation of the parameter values are done under either a Maximum likelihood or a Bayesian inference framework. The output can contain estimates of all migration rates and all population sizes, assuming constant mutation rates among loci or a gamma distributed mutation rate among loci; profile likelihood tables, percentiles, likelihood-ratio tests, and simple plots of the log-likelihood surfaces for all populations and all loci (Beerli, 2006; Beerli, 2008; Beerli, 2009; Beerli and Felsenstein, 1999; Beerli and Felsenstein, 2001).

10.22.1 Data type handled

MIGRATE can deal with following data types:

- DNA sequence
- SNP
- Microsatellite
- Standard (Electrophoretic marker)

10.22.2 MIGRATE format

Some syntax specifications:

- < token >: the token is obligatory
- [token]: optional
- {token}: obligatory for some
- < token1|token2 >: choose one of the token kind of data
- <individual1 10-10>: means that this token needs to be 10 characters long
- The characters for any word token can normally include special characters, punctuation, and blanks (e.g.:`Ind1 02 @` is legal)

Enzyme electrophoretic data or microsatellite data would look like this:

```
<Number of populations> <number of loci> {delimiter between alleles} [project title 0-79]
<Number of individuals> <title for population 0-79>
<Individual 1 10-10> <data>
<Individual 2 10-10> <data>
....
<Number of individuals> <title for population 0-79>
<Individuum 1 10-10> <data>
<Individuum 2 10-10> <data>
....
```

Enzyme electrophoretic data (infinite allele model):

- The project title is optional
- The individual name is by default 10 characters long
- The “data token” contains the genotypes
- Missing data are coded by ”?”
- One can use multi-character coding when using a delimiter

Microsatellite data:

- The project title is optional
- The individual name is by default 10 characters long
- The third argument on the first line has to be a delimiter character (e.g.: ”.”)
- The data contain the genotypes
- Homozygote individual needs to be coded as e.g.: 6.6 (”.” is the delimiter)
- Missing data are symbolized by ”?’”
- Each individual must have two alleles, which are coded as number of repeats or as fragment length (in this case an extra line with repeat numbers is needed: second line, starting with `#M`)

Sequences data:

- The individual name is followed by the base sequence of that species
- Blanks will be ignored and characters can be either upper or lower case
- characters constitute the IUPAC (IUB) nucleic acid code plus some slight extensions

Non-interleaved data:

```
<Number of populations> <number of loci> [project title 0-79]
<number of sites for locus1> <number of sites for locus 2> ...
<Number of individuals locus1> <#ind locus 2> ... <#ind loc n> <title for population 0-79>
<Individuum 1 10-10> <data locus 1>
<Individuum 2 10-10> <data locus 1>
....
<Individuum 1 10-10> <data locus 2>
<Individuum 2 10-10> <data locus 2>
....
<Number of individuals> <#ind locus 2> ... <#ind loc n> <title for population 0-79>
<Individuum 1 10-10> <data locus 1>
<Individuum 2 10-10> <data locus 1>
....
```

Interleaved data (not anymore supported by MIGRATE):

```
<Number of populations> <number of loci> [project title 0-79]
<number of sites for locus1> <number of sites for locus 2> ...
<Number of individuals locus1> <#ind locus 2> ... <#ind loc n> <title for population 0-79>
<Individual 1 10-10> <data locus 1 part 1>
<Individuum 2 10-10> <data locus 1 part 1>
....
<data ind1 locus 1 part 2>
<data ind2 locus 1 part 2>
....
<Individual 1 10-10> <data locus 2 part 1>
....
<data ind1 locus 2 part 2>
....
```

SNP data:

- The individual name is followed by the base sequence of that species
- Blanks will be ignored and characters can be either upper or lower case
- characters constitute the IUPAC (IUB) nucleic acid code plus some slight extensions
- two different formats: Nucleotide and HapMap

Nucleotide format:

- Same format as sequence data, except that first line starts with an `N`
- Linked SNP: more than one site on one line
- Unlinked SNP: one site per line

```
N <Number of populations> <number of loci> [project title 0-79]
<number of sites for locus1> <number of sites for locus 2> ...
<Number of individuals locus1> <#ind locus 2> ... <#ind loc n> <title for population 0-79>
<Individuum 1 10-10> <data locus 1>
<Individuum 2 10-10> <data locus 1>
....
<Individuum 1 10-10> <data locus 2>
<Individuum 2 10-10> <data locus 2>
....
<Number of individuals> <#ind locus 2> ... <#ind loc n> <title for population 0-79>
<Individuum 1 10-10> <data locus 1>
<Individuum 2 10-10> <data locus 1>
....
```

HapMap format:

- assumes that each SNP is biallelic
- <allele> contains the nucleotide
- <number> contains the number of individuals with the specific allele
- <total> is the sum of both numbers

```
H <Number of populations> <number of loci> [project title 0-79]
<Any Number> <title for population 0-79>
<Position on chr locus1> <TAB><allele><TAB><number><TAB><allele><TAB><number><TAB><total>
<Position on chr locus2> <TAB><allele><TAB><number><TAB><allele><TAB><number><TAB><total>
....
<Position on chr locus1> <TAB><allele><TAB><number><TAB><allele><TAB><number><TAB><total>
<Position on chr locus2> <TAB><allele><TAB><number><TAB><allele><TAB><number><TAB><total>
....
```

Examples:

- Enzyme electrophoretic data
  - 2 populations and 11 loci and with 2 or 1 individuals per population:

```
2 11 Migration rates between two Turkish frog populations
2 Akcapinar (between Marmaris and Adana)
PB1058    ee bb ab bb bb aa aa bb ?? cc ab
PB1059    ee bb ab bb bb aa aa bb bb cc aa
1 Ezine (between Selcuk and Dardanelles)
PB16843   ee bb ab bb aa aa aa cc bb cc aa
```

  - Same data but with / as separator:

```
2 11 / Migration rates between two Turkish frog populations
2 Akcapinar (between Marmaris and Adana)
PB1058    e/e b/b a/b b/b b/b a/a a/a b/b ?/? c/c Rs/Rf
PB1059    e/e b/b a/b b/b b/b a/a a/a b/b b/b c/c Rs/Rs
1 Ezine (between Selcuk and Dardanelles)
PB16843   e/e b/b a/b b/b a/a a/a a/a c/c b/b c/c Rs/Rs
```

- Microsatellite data:
  - Encoded as repeat numbers:

```
2 3 . Rana lessonae: Seeruecken versus Tal
2   Riedtli near Guendelhart-Hoerhausen
0         42.45 37.31 18.18
0         42.45 37.33 18.16
1   Tal near Steckborn
1         43.46 33.? 18.18
```

  - Encoded as fragment length:

```
2 3 . Rana lessonae: Seeruecken versus Tal
#M 2 2 3
2   Riedtli near Guendelhart-Hoerhausen
0         84.90 74.62 54.54
0         84.90 74.66 54.48
1   Tal near Steckborn
1         86.92 66.? 54.54
```

- Sequence data:
    - not interleaved (2 population with 2 loci):

```
   2 2 Make believe data set using simulated data (2 loci)
50 46
2 2   pop1
eis       ACACCCAACACGGCCCGCGGACAGGGGCTCGAGGGATCACTGACTGGCAC
zwo       ACACAAAACACGGCCCGCGGACAGGGGCTCGAGGGGTCACTGAGTGGCAC
eis       ACGCGGCGCGCGAACGAAGACCAAATCTTCTTGATCCCCAAGTGTC
zwo       ACGCGGCGCGAGAACGAAGACCAAATCTTCTTGATCCCCAAGTGTC
2   pop2
vier      CAGCGCGCGTATCGCCCCATGTGGTTCGGCCAAAGAATGGTAGAGCGGAG
fuef      CAGCGCGAGTCTCGCCCCATGGGGTTAGGCCAAATAATGTTAGAGCGGCA
vier      TCGACTAGATCTGCAGCACATACGAGGGTCATGCGTCCCAGATGTG
fuef      TCGACTAGATATGCAGCAAATACGAGGGGCATGCGTCCCAGATGTG
```

    - interleaved (2 populations with 2 loci, not anymore supported by MIGRATE):

```
   2 2 Make believe data set using simulated data (2 loci,interleaved)
50 46
2 1   pop1
zwo       ACACAAAACACGGCCCGCGGACA
drue      ATACCCAGCACGGCCGGCGGACA
          GGGGCTCGAGGGATCACTGACTGGCAC
          GGGGCTCGAGGGGTCACTGAGTGGCAC
          GGGGCTCGAGGGAGCACTGAGTGGAAC
zwo       ACGCGGCGCGAGAACGAAGACCA
          AATCTTCTTGATCCCCAAGTGTC
          AATCTTCTTGATCCCCAAGTGTC
2 2 pop2
vier      CAGCGCGCGTATCGCCCCATGTGGTTCGGCCAAAGAATG
fuef      CAGCGCGAGTCTCGCCCCATGGGGTTAGGCCAAATAATG
          GTAGAGCGGAG
  TTAGAGCGGCA
          TCGACTAGATCTG CAGCACATAC
          TCGACTAGATATG CAGCAAATAC
  GAGGGTCATGCGTCCCAGATGTG
  GAGGGGCATGCGTCCCAGATGTG
```

- SNP data:
    - Nucleotide format (2 populations and 2 loci):

```
N 2 2 Make believe data set using simulated data (2 loci)
1 4
3 3 pop1
ind1      A
ind2      A
ind3      A
ind1      ACAC
ind2      ACAC
ind3      ACGC
2 pop2
ind4      C
ind5      C
ind4      TGGA
ind5      TCGA
```

  - HapMap format:

```
H 2 2 Make believe data set using simulated data (2 loci)
3 pop1
1    A 3 C 0 3
1000 A 3 T 0 3
1010 C 3 G 0 3
1011 A 2 G 1 3
1015 C 3 A 0 3
2 pop2
1    A 0 C 2 2
1000 A 0 T 2 2
1010 C 1 G 1 2
1011 A 0 G 2 2
1015 C 0 A 2 2
```

10.22.3 Links and References

Website: http://popgen.sc.fsu.edu/Migrate/Migrate-n.html

Manual: http://popgen.scs.fsu.edu/migratedoc.pdf

(Beerli, 2009)

10.22.4 Special PGDSpider input/output questions

- Input:
  - Select the data type:
    Need to define if the data file contains Microsatellite, Standard, Sequence or SNP data.
  - Are the data interleaved? :
    Define if the data in the file are interleaved (use more than one line) or not.

- Output:
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
  - If numeric SNP data: Enter the integer that codes for the nucleotide:
    Define the integer that codes for a specific nucleotide
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.23 MSA

MSA version 4.05

MSA is a universal, platform independent, data analysis tool. It was designed to handle large microsatellite data sets. Microsatellite analyzer calculates the standard suite of descriptive statistics and provides input files for other software packages (Dieringer and Schlotterer, 2003).

10.23.1 Data type handled

MSA can only handle Microsatellite data

10.23.2 MSA format

The input files can be generated using a spreadsheet software (such as Excel), where the data are arranged either as one column per locus or as two columns per locus. The input file has to be saved as “tab delimited text” file.

The MSA input files should follow these rules:

- The microsatellites data should be coded as the PCR product size
- Missing data are indicated by either an empty cell or a negative value (not ‘0’)
- For compatibility with PHYLIP the population labels are limited to 8 characters
- cell A1 contains a 1 or a 2, whether the data are arranged in the one column (1) or two column (2) format
- The first column encloses the names of the populations (no empty cell is allowed)
- The second column specifies whether the data are inbred (h) or outbred (d). The same allele needs to be entered twice when only a single allele was detected (empty cells are thought to be missing data).
- The third column allows one to group populations. In the absence of grouping give the same number to all populations. Only consecutive group numbers are allowed, but groups are assigned without any constraints in order.
- The first two rows give information about each locus:
    - First row specifies the repeat type (1, 2, 3, etc). This is used to compute the number of repeats out of the PCR product size.
    - Second row indicates the length of the sequence flanking the microsatellite (in bp). This row can be empty.
- The third row contains the name of the microsatellite locus. In the two-column format, MSA allows two different names for the same locus (each entered in one cell)

- Example:

```
2                    2             2             2             2
              113           81            112           159
              1770   X13444 1818   X65444 1774   X66788 1772   X65644
Pop1   d      1      140    140                  147    159
Pop1   d      1      134    134                  147    151    184    186
Pop1   d      1      134    134    104    106    147    147    186    178
Pop2   d      1      134    136    104    100    159    153    186    172
Pop2   d      1      134    140    104    104    151    143    184    178
Pop2   d      1      134    134    104    104    147    151    186    188
Pop2   d      1      134    134                  147    141    184    178
Pop3   h      2                    104    104
Pop3   h      2      134    134    104    104    149    149    186    178
Pop3   d      2      134    134                  147    147    184    186
Pop3   h      2                    104    98
Pop3   d      2                    104    106                  186    178
```

10.23.3 Links and References

Website: http://i122server.vu-wien.ac.at/MSA/info.html/MSA_info.html

(Dieringer and Schlotterer, 2003)

10.23.4 Special PGDSpider input/output questions

- Input: none

- Output:
  - Enter the size of the repeated motif:
    This is needed to convert the Microsatellite data (PGD saves it as number of repeats) to the length of the PCR fragments (MSA can only save Microsatellite data as the length of the PCR fragment). If it is the same for all loci just enter one number else one has to enter a number for each locus separated by a comma.
  - Are data inbred (h) or outbred (d):
    Need to define if the data are in- or outbred
  - Specify the locus/locus combination which should be written to the output file (optional):
    If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.24 MSVar

MSVar version 0.4.1.b (7. April 1999)

This program is designed to help the user to explore the most probable demographic and genealogical histories consistent with a sample of chromosomes typed at one or more loci. It relies on Markov Chain Monte Carlo (MCMC) simulation (Beaumont, 1999).

10.24.1 Data type handled

MSVar can only handle Microsatellite data.

10.24.2 MSVar format

- The first row contains the number of loci
- Second row: number of alleles (allelic classes) at the first locus
- Third row: counts of chromosomes with the same length (same number of repeats)
- Fourth row: the number of repeats corresponding to counts above
- Fifth row: number of alleles at next locus
- Etc.

- Example:

```
4
2
28 20
0 7
2
11 29
0 3
3
12 14 6
0   1 2
2
50 6
0 2
anything you want down here
```

10.24.3 Links and References

Website: http://www.rubic.rdg.ac.uk/~mab/software.html

(Beaumont, 1999)

10.24.4 Special PGDSpider input/output questions

- Output:
    - Do you want to combine all populations? :
      Need to define if the MSVar file includes one population or a combination of all populations (all individuals of the different populations in one population)
    - Select the population you want to include:
      Need to choose the population which should be included.
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.25 NewHybrids

NewHybrids version 1.1 beta (7. April 2003)

NewHybrids is a program for computing the posterior distribution that individuals in a sample fall into different hybrid categories (Anderson and Thompson, 2002).

10.25.1 Data type handled

NewHybrids handles diploid Microsatellite, AFLP and Standard (multi-allelic markers) data types.

10.25.2 NewHybrids format

- whitespace (spaces and or tabs) separated text file *.txt/*.dat
- first line: `NumIndivs` number of individuals
- second line: `NumLoci` number of loci
- third line: `Digits` number of digits used to denote a particular allele
- fourth line: `Format Lumped` (genotype at a single locus is given by a single number) or `NonLumped`
- next lines: `LocusNames` names of all loci separated by whitespace
- next lines: genotype data
- first character: number of the individual (numbering must be serially)
- next characters: genotypes (all on same line or on different lines)
- `Lumped` format: two alleles are encoded as one number, `Digits` specify how many digits are used to represent each locus
- `NonLumped` format: alleles at each locus are given by a consecutive pair of numbers that are white space separated
- Missing data: `Lumped:` encoded as `0, NonLumped:` encoded as `-1` (each allele at the missing locus must have a `-1`)

AFLP data:

- `LumpedLumped` format
- `+` band is present
- − band is absent
- `0` missing data
- data types can be mixed

Example

- `Lumped` data file:

```
NumIndivs 2
NumLoci 6
Digits 1
Format Lumped
LocusNames sAAT1 sAAT2 sAAT3 ADA1 ADA2 ADH
1 11 11 11 0 11 32
2 21 11 21 11 11 12
```

- `NonLumped` data file:

```
NumIndivs 2
NumLoci 6
Digits 1
Format NonLumped
LocusNames sAAT1 sAAT2 sAAT3 ADA1 ADA2 ADH
1 123 143 -1 -1 144 144 120 122 157 158 144 144
2 135 135 134 140 144 144 120 122 161 161 144 144
```

- AFLP data file (4 Microsat loci, 5 AFLP loci):

```
NumIndivs 2
NumLoci 9
Digits 1
Format Lumped
LocusNames m1 m2 m3 m4 A1 A2 A3 A4 A5
1 11 12 13 11 + + + - +
2 22 33 11 22 - - 0 - -
3 12 13 13 11 + - - - +
```

10.25.3 Links and References

Website: http://ib.berkeley.edu/labs/slatkin/eriq/software/software.htm,

Manual: http://ib.berkeley.edu/labs/slatkin/eriq/software/new_hybs_doc1_1Beta3.pdf

(Anderson and Thompson, 2002)

10.25.4 Special PGDSpider input/output questions

- Input:
  - Select the data type:
    One has to define the type of the data (e.g.: Microsat, AFLP or Standard)
  - How are Microsat alleles coded?
    Need to define if the Microsat data are coded as number of repeats, as the length of the PCR fragments or as an arbitrary number.

  - Enter the size of the repeated motif:
    Need to define the size of the repeated motif, so that the number of repeats can be calculated (Microsat alleles have to be saved as number of repeats in the PGD format)

- Output:
  - Specify which data type should be included (optional):
    If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.26 NEXUS

NEXUS is a file format designed to contain systematic data. The goals of the format are to allow future expansion, to include diverse kinds of information, to be independent of particular computer operating systems, and to be easily processed by a program (Maddison, et al., 1997).

10.26.1 NEXUS format

NEXUS files are free-format, which means that the entire file could conceivably consist of a single, long line of text. It does not matter where the line is broken (as long as you don't split up a keyword or the name of a locus, allele or population), nor does it matter if one space or a dozen spaces are used to separate the individual words (tokens) in the file. Tokens may be casually defined as sequences of characters separated by whitespace (e.g., spaces, carriage returns, line feeds, tabs, etc.)

NEXUS files are for the most part not case-sensitive by default. A big exception is in the matrix command, where (by default) an allele named A is treated as being distinct from a.

The NEXUS files are built as follows:

- Comments can be added by enclosing text within brackets: `[comment]`
- The file has to start with: `#NEXUS`
- The tokens in a NEXUS file are organized into commands, which are in turn organized into blocks.
    - Commands: the first token in the command is the command name, which is followed by a series of tokens and whitespace; the command is terminated by a semicolon: `command-name token token . . . ;`
    - Blocks: series of commands, beginning with a BEGIN command and ending with an END command:

```
BEGIN block-name;
 command-name token . . . ;
 command-name token . . . ;
 ...
END;
```

The most used public blocks are:

(Tokens within [ ] are optional, within { | | } are mutually exclusive and underlined tokens are the default):

- TAXA:
  TAXA block defines the taxa and gives them names. The block also establishes the order (numbering) of the taxa. Taxa consist of the entities whose attributes might be recorded in characters block.

```
BEGIN TAXA;
 DIMENSIONS NTAX=number-of-taxa;
 TAXLABELS taxon-name [taxon-name ...] ;
END;
```

- CHARACTERS:
  This block contains the information about discrete and continuous data, including that for morphological structure and molecular sequences. Polymorphism and frequency data can be accommodated. Names can be given to the characters and their states.

```
BEGIN CHARACTERS;
 DIMENSIONS [NEWTAXA NTAX=number-of-taxa] NCHAR=number-of-characters;
 [FORMAT
  [DATATYPE={STANDARD|DNA|RNA|NUCLEOTIDE|PROTEIN|CONTINUOUS}
  [RESPECTCASE]                                  default: A and a is the same
  [MISSING=symbol]                               default: ?
  [GAP=symbol]
  [SYMBOLS=”symbol [symbol...]”]
  [EQUATE=”symbol=entry [symbol=entry]”]
  [MATCHCHAR=symbol]
  [[No]LABELS]
  [TRANSPOSE]
  [INTERLEAVE]
  [ITEMS=([MIN][MAX][MEDIAN][AVERAGE][VARIANCE][STCERROR][SAMPLESIZE][STATES])]
  [STATESFORMAT={STATESPRESENT|INDIVIDUALS|COUNT|FREQUENCY}]
  [[No]TOKENS]
 ;]
 [ELIMINATE character-set;]
 [TAXLABELS taxon-name [taxon-name...];]
 [CARSTATELABELS character-number [charact-name] [/state-name [state-name..]]
  [, character-number [character-name] [/state-name [state-name...]] ...]
 ;]
 [CHARLABELS character-name [character-name...];]
 [STATELABELS character-number [character-name] [/state-name [state-name...]]
  [, character-number [character-name] [/state-name [state-name...]]  ...]
 ;]
 MATRIX data-matrix;
END;
```

- example:

```
BEGIN CHARACTERS;
 DIMENSION NCHAR=3;
 CHARSTATELABELS 1 hair/absent present, 2 color/red blue,
                 3 size/small big;
 FORMAT TOKENS;
 MATRIX
  taxon_1 absent red big
  taxon_2 absent blue small
  taxon_3 present blue small;
END;
```

- UNALIGNED:
  similar to a CHARACTERS block, but contains unaligned molecular sequence data.

```
BEGIN UNALIGNED;
 [DIMENSIONS NEWTAXA NTAX=number-of-taxa;]
 [FORMAT
  [DATATYPE={STANDARD|DNA|RNA|NUCLEOTIDE|PROTEIN}]
  [RESPECTCASE]
  [MISSING=symbol]
  [SYMPOLS=”symbol [symbol...]”]
  [EQUATE=”symbol=entry [symbol=entry...]”]
  [[No]LABELS]
 ;]
 [TAXLABELS taxon-name [taxon-name...];]
 MATRIX data-matrix;
END;
```

- DISTANCES:
  This block contains distance matrices

```
BEGIN DISTANCES;
 [DIMENSIONS [NEWTAXA] NTAX=number-of-taxa NCHAR=number-of-characters;]
 [FORMAT
  [TRIANGLE={LOWER|UPPER|BOTH}]
  [[NO]DIAGONAL]
  [[NO]LABELS]
  [MISSING=symbol]
  [INTERLEAVE]
 ;]
 [TAXLABELS taxon-name [taxon-name...];]
 [MATRIX distance-matirx;
END;
```

  - example:

```
BEGIN DISTANCES;
 FORMAT TRIANGLE=UPPER;
 MATRIX
  taxon_1 0.0 1.0 2.0
  taxon_2     0.0 3.0
  taxon_3         0.0;
END;
```

- DATA:
  DATA is a CHARACTERS block that includes not only the definition of characters but also the definition of taxa (this block is not recommended).
    - example:

```
BEGIN DATA;
 DIMENSIONS NTAX=5 NCHAR=20;
 FORMAT DATATYPE=DNA GAP=-;
 MATRIX
  taxon-1 A-CTAGGACTA---GATCAA
  taxon-2 A-CCAGGACTAGCGGATCAA
  taxon-3 A-CCAGGACTA---GATCAA
  taxon-4 AGCCAGGACTA---GTTCAA
  taxon-5 ATC-AGGACTA---GATCAA;
END;
```

- SETS:
  This block contains descriptions of collections of objects. These objects include characters, taxa, trees, states, and kinds of changes. In addition, partitions of characters, taxa, and trees can be formed.

```
BEGIN SETS;
 [CHARSET charstet_name [({STANDARD|VECTOR})]=character-set;]
 [STATESET stateset-name [({STANDARD|VECTOR})]=state-set;]
 [CHANGESET changeset-name=state-set<->state-set [state-set<->state-set...];]
 [TAXSET taxset-name [({STANDARD|VECTOR})]=taxon-set;]
 [TREESET treeset-name [({STANDARD|VECTOR})]=tree-set;]
 [CHARPARTITION partition-name [([{[NO]TOKENS}] [{STANDARD|VECTOR}])]
  =subset-name:character-set [, subset-name:character-set...];]
 [TAXPARTITION partition-name [([{[NO]TOKENS}] [{STANDARD|VECTOR}])]
  =subset-name:taxon-set [, subset-name:taxon-set...];]
 [TREEPARTITION partition-name [([{[NO]TOKENS}] [{STANDARD|VECTOR}])]
  =subset-name:tree-set [, subset-name:tree-set...];]
END;
```

- ASSUMPTIONS:
  contains assumptions about the data. These can include assignment of weights to various characters, specification of the nature of character changes, exclusion of particular characters, and designation of ancestral states.

```
BEGIN ASSUMPTIONS;
 [OPTIONS [DEFTYPE=type-name]
  [POLYTCOUNT={MINSTEPS|MAXSTEPS}]
  [GAPMODE={MISSING|NEWSTATE}];]
 [USERTYPE type-name[({STEPMATRIX|CSTREE})]=USERTYPE-description;]
 [TYPESET [*] typeset-name [({STANDARD|VECTOR})]=TYPESET-definition;]
 [WTSET [*] stset-name [({STANDARD|VECTOR} {TOKENS|NOTOKENS})]=WTSET-
   definition;]
 [EXSET [*] exset-name [({STANDARD|VECTOR})]=character-set;]
 [ANCSTATES [*] ancstates-name [({STANDARD|VECTOR} {TOKENS|NOTOKENS})]
   =ANCSTATES-definition;]
END;
```

- CODONS:
  contains information about the genetic code, the regions of DNA and RNA sequences that are protein coding, and the location of triplets coding for amino acids in nucleotide sequences.

```
BEGIN CODONS;
 [CODONPOSSET [*] name [({STANDARD|VECTOR})]=
  N: character-set,
  1: character-set,
  2: character-set,
  3: character-set;]
 [GENETICCODE code-name
  [([CODEORDER=123|other] [NUCORDER=TCAG|other] [[NO]TOKENS]
     [EXTENSIONS=“symbols-list“])]
   =genetic code description];]
 [CODESET [*] codeset-name {(CHARACTERS|UNALIGNED|TAXA)}
   =code-name:character-set or taxon-set [,code-name:character-set or
taxon-
     set...];]
END;
```

- TREES:
  stores information about trees.

```
BEGIN TREES;
 [TRANSLATE arbitrary-token-used-in-tree-description valid-taxon-name
  [, arbitrary-token-used-in-tree-description valid-taxon-name. . . ] ;]
 [TREE [*] tree-name= tree-specification;]
END;
```

- NOTES:
  allows attachment of additional information (text, pictures, etc.) to various objects (trees, taxa, characters, etc.) in the file.

```
BEGIN NOTES;
 [TEXT [TAXON=taxon-set] [CARACTER=character-set] [STATE=state-set]
   [TREE=tree-set]
  SOURCE={INLINE|FILE|RESOURCE}TEXT=text-or-source-description:]
 [PICTURE [TAXON=taxon-set] [CARACTER=character-set] [STATE=state-set]
   [TREE=tree-set]
  [FORMAT=[PICT|TIFF|EPS|JPEG|GIF}] [ENCODE={NONE|UUENCODE|BINHEX}]
  [SOURCE={INLINE|FILE|RESOURCE}PICTURE=picture-or-source-descriptior;]
END;
```

- The order of blocks is predetermined for some pairs of blocks but not others (most programs will require a CHARACTERS or DATA block to precede the ASSUMPTIONS block so that the characters will be defined)
- Names should be unique (no duplicate names), must be single words (no spaces) and cannot consist entirely of digits.

Example:

```
#NEXUS
BEGIN TAXA;
      Dimensions NTax=4;
      TaxLabels fish frog snake mouse;
END;

BEGIN CHARACTERS;
      Dimensions NChar=20;
      Format DataType=DNA;
      Matrix
        fish   ACATA GAGGG TACCT CTAAG
        frog   ACATA GAGGG TACCT CTAAG
        snake  ACATA GAGGG TACCT CTAAG
        mouse  ACATA GAGGG TACCT CTAAG
END;

BEGIN TREES;
      Tree best=(fish, (frog, (snake, mouse)));
END;
```

10.26.2 References

Maddison, D. R., D. L. Swofford, et al. (1997). "Nexus: An extensible file format for systematic information." Systematic Biology **46**(4): 590-621.

10.26.3 Special PGDSpider input/output questions

- Input:
    - Do you want to include the sequence not specified within the TaxSet in the SET block? :
      If one or more sequences are not specified within the TaxSet in the SET block, one need to specify if these sequences should be included (they are put all together in one population without a name) or not.

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
      PHYLIP
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.27 PED

The PED file format refers to the widely-used format for linkage pedigree data and used as input for the program PLINK. PLINK is a free, open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner (Anderson and Thompson, 2002).

10.27.1 Data types handled

PHYLIP is able to deal with diploid SNP data.

10.27.2 PED format

- whitespace (spaces and or tabs) separated text file *.ped
- each line correspond to one individual
- following first 6 columns are mandatory (The IDs are alphanumberic):
    - `Family ID`
    - `Individual ID`
    - `Paternal ID`
    - `Maternal ID`
    - `Sex` (1=male; 2=female; any other character=unknown)
    - `Phenotype` (only 1 phenotype! The phenotype can be either a quantitative trait or an affection status column: PLINK will automatically detect which type (i.e. based on whether a value other than 0, 1, 2 or the missing genotype code is observed))
- Comments: line starts with `#`
- Affection status, by default, should be coded:
    - -9 missing
    - 0 missing
    - 1 unaffected
    - 2 affected
- column 7 onwards: Genotypes
    - any character (e.g.: 1,2,3,4 or A,C,G,T or anything else)
    - missing genotype: `0`
    - All markers must be biallelic (diploid). Either both alleles should be missing or neither. Haploid data: encode them as diploid homozygote. Two alleles are shown after each other.

If specially specified following columns can be missing:

- `Family ID`
- `Individual ID`
- `Paternal ID` and `Maternal ID`
- `Sex`
- `Phenotype`

MAP files

- Each line of the MAP file describes a single marker and must contain exactly 4 columns:
  - chromosome (1-22, X, Y, MT or 0 if unplaced)
  - rs# or snp identifier
  - Genetic distance (morgans) (missing: 0)
  - Base-pair position (bp units) (Base-pair positions are expected to correspond to positive integers within the range of typical human chromosome sizes)
- The MAP file must contain as many markers as are in the PED file.
- The markers in the PED file do not need to be in genomic order, but the order MAP file should align with the order of the PED file markers).

Example

- PED files:

```
FAM001  1  0 0  1  2  A A  G G  A C
FAM001  2  0 0  1  2  A A  A G  0 0
```

```
1 1 0 0 1   1   A A   A A   A A   A A   A A
2 1 0 0 1   1   A C   A C   A C   A C   A C
3 1 0 0 2   1   A A   A A   A A   A A   A A
4 1 0 0 2   1   A C   A C   A C   A C   A C
```

- MAP files:

```
rs123456  0  1234555
rs234567  0  1237793
rs224534  0  -1237697
rs233556  0  1337456
```

```
1    snp1   0   1000
X    snp2   0   1000
Y    snp3   0   1000
XY   snp4   0   1000
MT   snp5   0   1000
```

10.27.3 Links and References

Website: http://pngu.mgh.harvard.edu/~purcell/plink/data.shtml#ped

(Anderson and Thompson, 2002)

10.27.4 Special PGDSpider input/output questions

- Input:
    - Include MAP file with loci information:
      Possibility to add a file with loci information
    - Is the “Family ID” column absent in the input file:
      Specify if the Family IDs are absent or not
    - Is the “Individual ID” column absent in the input file:
      Specify if the Individual IDs are absent or not
    - Is the “Parental ID” and the “Maternal ID” columns absent in the input file:
      Specify if the Paternal IDs and the Maternal IDs are absent or not
    - Is the “Sex” column absent in the input file:
      Specify if the Sexes are absent or not
    - Is the “Phenotype” column absent in the input file:
      Specify if the Phenotypes are absent or not

- Output:
    - Save additional file with loci information:
      Saves a MAP file with loci information
    - Replacement character for allele encoded as 0:
      Specify the character which should encode for allele 0 (0 encodes for missing data in PED)
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.28 PHYLIP

PHYLIP version 3.69 (September 2009)

PHYLIP, the Phylogeny Inference Package, is a package of programs for inferring phylogenies (evolutionary trees). It can infer phylogenies by parsimony, compatibility, distance matrix methods, and likelihood. It can also compute consensus trees, compute distances between trees, draw trees, resample data sets by bootstrapping or jackknifing, edit trees, and compute distance matrices (Felsenstein, 1989; Felsenstein, 2004).

10.28.1 Data types handled

PHYLIP is able to deal with following data types:

- nucleotide sequences
- protein sequences
- gene frequencies
- restriction sites
- restriction fragments
- distances
- discrete characters
- continuous characters

10.28.2 PHYLIP format

For most of the PHYLIP programs, information comes from a series of input files, and ends up in a series of output files.

Nucleotide sequences data:

- The first line contains the number of species and the number of characters. These are in free format, separated by blanks.
- The next lines include information for each species: First, the species name has to be 10 characters long (it can include blanks and punctuation marks), followed by the data for that species (the data have to start at the 11th character of the line!). The name should be on the same line as the first character of the data.
- The conventions for interleaved data are different between the molecular sequence programs and the others. The molecular sequence programs can take the data in “aligned” or “interleaved” format:
    - In the interleaved format DNA sequences can be specified on several lines. It is important that the sequence length in each group is the same for all species. The sequences might look like this:

```
    2   39
Archaeopt CGATGCTTAC CGCCGATGCT
HesperorniCGTTACTCGT TGTCGTTACT

TACCGCCGAT GCTTACCGC
CGTTGTCGTT ACTCGTTGT
```

  - In the sequential format the character data can run on a new line at any time. Thus, it is legal to have:

```
Archaeopt 001100
1101
```

or even:

```
Archaeopt
0011001101
```

- Blanks and digits within sequences are allowed to make them easier to read

- Example:

```
   6   13
Archaeopt CGATGCTTAC CGC
HesperorniCGTTACTCGT TGT
BaluchitheTAATGTTAAT TGT
B. virginiTAATGTTCGT TGT
BrontosaurCAAAACCCAT CAT
B.subtilisGGCAGCCAAT CAC
```

Distance Matrix

- The first line of these input files must contain the number of species
- Then the species data follow, starting with a species name:
  - species names have to be ten characters long
  - Then a set of distances to all the other species follows for each species (the distance matrix can be upper- or lower-triangular or square). The distances can continue to a new line.

- Examples:
    - A square matrix:

```
     5
Alpha      0.000 1.000 2.000 3.000 3.000
Beta       1.000 0.000 2.000 3.000 3.000
Gamma      2.000 2.000 0.000 3.000 3.000
Delta      3.000 3.000 0.000 0.000 1.000
Epsilon    3.000 3.000 3.000 1.000 0.000
```

    - A lower-triangular input matrix with distances continuing to new lines as needed:

```
   14
Mouse
Bovine      1.7043
Lemur       2.0235  1.1901
Tarsier     2.1378  1.3287  1.2905
Squir Monk  1.5232  1.2423  1.3199
1.7878
Jpn Macaq   1.8261  1.2508  1.3887
1.3137  1.0642
```

10.28.3 Links and References

Website: http://evolution.genetics.washington.edu/phylip/doc/main.html

(Felsenstein, 1989; Felsenstein, 2004)

10.28.4 Special PGDSpider input/output questions

- Input:
    - What type of data is listed in the PHYLIP file:
      Need to define if the file contains molecular sequence data or distance matrix data
    - Specify the format of the data:
      Need to define if the data are simple (on one row), interleaved or sequential
    - Specify the format of the distance matrix:
      Need to define if the distance matrix is of the format lower-triangular, square matrix (both), upper-triangular (with diagonals) or upper-triangular (without diagonals).

- Output:
    - Select the kind of file you want to write:
      Need to define if a molecular sequence data file or a distance matrix data file should be written to the output file.
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

- Specify the locus/locus combination which should be written to the output file (optional):
  If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.29 SAM

SAM version 1.4 (17. April 2011)

SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments.SAM aims to be a format that:

- Is flexible enough to store all the alignment information generated by various alignment programs;
- Is simple enough to be easily generated by alignment programs or converted from existing alignment formats;
- Is compact in file size;
- Allows most of operations on the alignment to work on a stream without loading the whole alignment into memory;
- Allows the file to be indexed by genomic position to efficiently retrieve all reads aligning to a locus.

The program SAMtools provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format (Li, et al., 2009).

The conversion process of the format SAM needs the programs Samtools (version 0.1.12/0.1.06) and Bcftools, which can be downloaded from http://samtools.sourceforge.net. The paths to the program files (samtools.exe and bcftools.exe) have to be specified in the “Config” menu under “Options” (see section 5.3.1 PGDSpider menus).

10.29.1 Data type handled

SAM can handle data of following type:

- DNA
- UHTS (Ultra High-Throughput Sequencing data)

10.29.2 SAM format

SAM is a tab-delimited text format with following file extension: *.sam
It consist of a header and a alignment section.

Header section:

- Optional, but recommended
- Each header line begins with a “@”, followed by a two-letter record type code
- The following table gives the defined record types and and tags (* required when record type is present)

Type	Tag	Description
HD (header)	VN*	File format version
	SO	Sort order (unsorted/queryname/coordinate)
	GO	Group order (none/query/reference)
SQ (sequence dictionary)	SN*	Sequence name
	LN*	Sequence length
	AS	Genome assembly identifier
	M5	MD5 checksum of the sequence in the uppercase
	UR	URI of the sequence
	SP	species
RG (read group)	ID*	Unique read group identifier
	CN	Name of the sequencing center producing the read
	SM	Sample
	LB	Library
	DS	Description
	PU	Platform unit
	PI	Predicted median insert size
	DT	Date the run was produced
	PL	Platform/technology used to produce the reads
PG (Program)	ID*	Program name
	PN	Program name
	VN	Program version
	CL	Command line
	PP	Previous PG-ID
CO (comment)		One-line text comments

Alignment Section

The format of each field in a line is:

- QNAME and FLAG are required for all alignments
- SEQ and QUAL can be absent and represented as a * (if QUAL is present, it must have the same length as SEQ)
- Optional fields: all optional fields follow the format TAG:TYPE:VALUE (for more details http://samtools.sourceforge.net/SAM1.pdf)
    - TAG: two-character string. Each TAG can only appear once in one alignment line
    - TYPE: single case sensitive letter which defines the format of VALUE
    - Example: RG:Z:MarCHGS13

Field	**Regular expression**	**Range**	**Description**
QNAME	`[!-?A-~]f1,255g`		Query pair name if paired; or query name if unpaired (unique!)
FLAG	`[0-9]+`	[0,216-1]	bitwise FLAG (0: forward read, 16: reverse read)
RNAME	`[!-()+-<>-~][!-~]*`		Reference sequence name (If SQ is present in the header, RNAME and MRNM must appear in an SQ header record)
POS	`[0-9]+`	[0,229-1]	1-based leftmost position/coordinate of the clipped sequence
MAPQ	`[0-9]+`	[0,28-1]	Mapping quality (phred-scaled posterior probability that the mapping position of this read is incorrect), mapping quality is not available: 255
CIGAR	`([0-9]+[MIDNSHPX=])+`		extendend CIGAR string
RNEXT	`=|[!-()+-<>-~][!-~]*`		Ref. name of the mate/next fragment, '=' if the same as RNAME, '*' if pairing information is not available
PNEXT	`[0-9]+`	[0,229-1]	Position of the mate/next fragment, '0' if pairing information is not available
TLEN	`-?[0-9]+`	[-229,229]	observed Tempplate LENgth, '0' if pairing information is not available
SEQ	`[A-Za-z=.]+`		fragment sequence, '=' for a match to the reference, n/N/. for ambiguity
QUAL	`[!-~]+`	[0,93]	base quality, ASCII-33 gives the Phred base quality
TAG	`[A-Z][A-Z0-9]`		two-character tag, optional
TYPE	`[AifZH]`		casesensitive single letter which defines the format of VALUE (e.g.: RG for read group), optional
VALUE	`[^\t\n\r]+`		match TYPE, optional

CICAR format:

- CIGAR string is a comprised of series of operation length plus the operation:
- Example: `43M1I14M1D10M`
  43 bases which matches/mismatches to the reference sequence, followed by 1 insertion, followed by 14 matches/mismatches, followed by 1 deletion, followed by 10 matches/mismatches

op	Description
M	Alignment match (sequence match or mismatch)
I	Insertion to reference
D	Deletion from reference
N	Skipped region from reference
S	Soft clip on the read (clipped sequence present in <seq>)
H	Hard clip on the read (clipped sequence not present in <seq>)
P	Padding
=	sequence match
X	sequence mismatch

- Sum of lengths of the M/I/S/=/X operations ought to equals the length of SEQ

Example:

```
@HD     VN:1.0
@SQ     SN:chr20 LN:62435964
@RG     ID:L1 PU:SC_1_10 LB:SC_1 SM:NA12891
@RG     ID:L2 PU:SC_2_12 LB:SC_2 SM:NA12891
read_28833_29006_6945   99   chr20   28833   20   10M1D25M   =   28993  195
AGCTTAGCTAGCTACCTATATCTTGGTCTTGGCCG   <<<<<<<<<<<<<<<<<<<<<:<9/,&,22;;<<<
read_28701_28881_323b   147  chr20   28834   30   35M        =   28701  -168
ACCTATATCTTGGCCTTGGCCGATGCGGCCTTGCA   <<<<<;<<<<7;:<<<6;<<<<<<<<<<<<7<<<<
```

10.29.3 Links and References

Website: http://samtools.sourceforge.net,

Manual: http://samtools.sourceforge.net/SAM1.pdf

(Li, et al., 2009)

10.29.4 Special PGDSpider input/output questions

- Input:
    - Reference file:
      Choose the file with the reference sequences
    - Select what should be imported:
      Defines if all reads, the consensus sequences or only the variant sites (SNP) should be imported
    - Concatenate consensus sequences from different reference data (only works if you choose to import consensus sequences):
      Specify if consensus sequences coming from different reference sequences should be concatenated or not
    - What is the ploidy of the data:
      Define if the data are haploid or diploid
    - Only import following regions (optional):
      Defines which regions should be imported. Regions should be defined in following format: refSeqName:start:end, multiple regions: separate it with white spaces

- Output:
    - Save an additional file with reference sequences:
      Saves a file with the reference sequences
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).

10.30 STRUCTURE

STRUCTURE version 2.3.3 (January 2010)

The program STRUCTURE implements a model-based clustering method for inferring population structure using genotype data consisting of unlinked markers. It includes inferring the presence of distinct populations, assigning individuals to populations, studying hybrid zones, identifying migrants and admixed individuals, and estimating population allele frequencies in situations where many individuals are migrants or admixed (Falush, et al., 2003; Falush, et al., 2007; Pritchard, et al., 2000).

10.30.1 Data type handled

STRUCTURE can handle haploid and diploid data of following type:

- SNP (numeric)
- Microsatellites
- RFLP
- AFLP

10.30.2 STRUCTURE format

The STRUCTURE data file is arranged as a matrix, in which the data for individuals are in rows, and the loci are in columns. For a diploid organism, data for each individual can be stored either on 2 consecutive rows, where each locus is in one column, or alternatively on one row, where each locus is in two consecutive columns.

The rows contain the:

- Marker Names (Optional; string):
  The first row can contain a list of identifiers for each of the markers (loci) in the data set.
- Recessive Alleles (Data with dominant markers only; integer):
  SNPs or microsatellites data sets would generally not include this line. This row indicates which allele (if any) is recessive at each locus.
- Inter-Marker Distances (Optional; real numbers):
  The next row is a set of inter-marker distances, for use with linked loci. These should be genetic distances (e.g., centiMorgans). The markers must be in map order within linkage groups. When consecutive markers are from different linkage groups (e.g.: different chromosomes), these should be indicated by the value -1. A value -1 is also assigned to the first marker. All other distances should be non-negative.
- Phase Information (Optional; diploid data only; real number in the range [0,1]):
  This is for use with the linkage model only. A single row of probabilities that appears after the genotype data for each individual. There are two alternative representations for the phase information:
  - The two rows of data for an individual are assumed to correspond to the paternal and maternal contributions. The phase line indicates the probability that the ordering is correct at the current marker (set MARKOVPHASE=0) respectively.

  - the phase line indicates the probability that the phase of one allele relative to the previous allele is correct (set MARKOVPHASE=1)
- Individual/Genotype data (Required):
  Data for each sampled individual are arranged into one or more rows.

Each row of individual data contains the following elements, which are in different columns in the data file:

1. Label (Optional; string):
   A string of integers or characters used to name each individual in the sample.
2. PopData (Optional; integer):
   An integer designating a user-defined population from which the individual was obtained
3. PopFlag (Optional; 0 or 1):
   A Boolean flag which indicates whether to use (1) or not (0) use the PopData when learning samples are used. These are samples whose origin is unknown, but they are classified with the help of individuals whose origin is known.
4. Phenotype (Optional; integer):
   An integer designates the value of a phenotype of interest for each individual.
5. Extra Columns (Optional; string):
   It may be convenient for the user to include additional data in the input file which are ignored by the program. These go here, and may be strings of integers or characters.
6. Genotype Data (Required; integer):
   Each allele at a given locus should be coded by a unique integer (e.g. microsatellite repeats score).

- Missing data should be indicated by a number that is not present anywhere else in the data (often -9 by convention).

- Example:

```
            loc_a  loc_b  loc_c  loc_d  loc_e
George   1   -9     145    66     0      92
George   1   -9     -9     64     0      94
Paula    1   106    142    68     1      92
Paula    1   106    148    64     0      94
Matthew  2   110    145    -9     0      92
Matthew  2   110    148    66     1      -9
Bob      2   108    142    64     1      94
Bob      2   -9     142    -9     0      94
```

10.30.3 Links and References

Website: http://pritch.bsd.uchicago.edu/structure.html

Manual: http://pritch.bsd.uchicago.edu/structure_software/release_versions/v2.3.3/structure_doc.pdf

(Falush, et al., 2003; Falush, et al., 2007; Hubisz, et al., 2009; Pritchard, et al., 2000)

10.30.4 Special PGDSpider input/output questions

- Input:
    - What is the ploidy of the data:
      Must be defined if the data are haploid or diploid
    - “Phase information” row present:
      Specify if a phase information row is present or not
    - What is the missing value code:
      Enter the symbol coding for missing values, e.g.: -9, -999, etc.
    - Select the data type:
      Define if the data are Microsatellite, RFLP, SNP or AFLP data
    - How are Microsat alleles coded:
      Define if the Microsat data are coded as number of repeats, as length of the PCR fragments or as an arbitrary number
    - Enter the size of the repeated motif:
      Needed to convert the Microsatellite data (length of the PCR fragments) to number of repeat data (PGD can only save number of repeat Microsat data)
    - Are marker (locus) names included:
      Specify if locus names are included or not
    - Enter number of markers (loci) listed in the input file:
      If loci names are not present, define the number of loci.
    - Are individual names (labels) included in the input file:
      Specify if individual names are included or not
    - Is the “PopData” column (population identifier) present in the input file:
      Specify if the population identifiers are present or not
    - Are the "Recessive Alleles" row and/ or the "Inter-Marker Distance" row present in the input file:
      Define if both, only one or none of the two rows are present

- Output:
    - Specify which data type should be included (optional):
      If there is more than one allowed data type, one has to select the data type which should be included in the output file (only one data type can be analysed per file).
    - Specify the locus/locus combination which should be written to the output file (optional):
      If several locus or locus combinations exists (PGD format is able to store several locus or locus combinations), one has to specify the locus or locus combination which should be included in the output file.

10.31 VCF

VCF version 4.0 (14. May 2011)

VCF (Variant Call Format) format stores structural variant data.

The conversion process of the format VCF needs the programs Samtools (version 0.1.12/0.1.06) and Bcftools, which can be downloaded from http://samtools.sourceforge.net. The paths to the program files (samtools.exe and bcftools.exe) have to be specified in the “Config” menu under “Options” (see section 5.3.1 PGDSpider menus).

10.31.1 Data type handled

VCF can handle data of following type:

- SNP
- DNA
- UHTS (Ultra High-Throughput Sequencing data)

10.31.2 VCF format

VCF is a tab-delimited text format with following file extension: *.vcf

The format contains metainformation lines, a header line, and data lines which contains information about a position in the genome.

Meta-information lines

- begins with `##`
- often
- often as key=value pairs
- 'fileformat' (mandatory):
    - VCF format version
    - e.g.: `##fileformat=VCFv4.0`
- 'INFO':
    - `##INFO=<Flag_ID>,<Number_of_Values>,<Value_Type>,<Description>`
    - `<Number_of_Values>`: Integer that describes the number of values that can be included in the INFO field (values varies, unknown or unbounded: -1)
    - `<Value_Types>`: Integer, Float, Character, String and Flag. The 'Flag' type indicates that the INFO field does not contain a Value entry, and hence `<Number_of_Values>` should be 0 in that case.
- 'FILTER':
    - Filters that have been applied to the data
    - `##FILTER=<FILTER_ID>,<Description>`

- 'FORMAT':
  - `##FORMAT=<FORMAT_ID>,<Number_of_Values>,<Value_Type>,<Description>`
  - `<Value_Types>`: Integer, Float, Character, and String.

Header line

- tab delimited
- names the 8 fixed, mandatory columns:
  1. #CHROM
  2. POS
  3. ID
  4. REF
  5. ALT
  6. QUAL
  7. FILTER
  8. INFO
- If genotype data is present:
  9. FORMAT column header
  10. an arbitrary number of sample ids

Data line

Fixed fields:

- tab-delimited
- missing values: "."
- 8 fixed fields per record:

1. CHROM chromosome:
   - an identifier from the reference genome.
   - Alphanumeric String, required
2. POS position:
   - The reference position (1st base having position 1).
   - Positions are sorted numerically, in increasing order, within each reference sequence.
   - Integer, required
3. ID:
   - A unique identifier. If this is a dbSNP variant: use the rs number.
   - Alphanumeric String, Missing value: ”.”
4. REF reference base:
   - One of A, C, G, T, N. Bases should be in uppercase.
   - Multiple bases are permitted. The value in the POS field refers to the position of the first base in the String.
   - For InDels, the reference String must include the base before the event (which must be reflected in the POS field).
   - String, required

5. ALT:
    - Comma separated list of alternate non-reference alleles.
    - Options are A, C, G, T, Dn (for delete n bases starting with the base at POS), I<seq> (where <seq> is a list of ACGT bases to be inserted just after the base at POS).
    - If there are no alternative alleles, then period character should be used.
    - Bases should be in uppercase.
    - Alphanumeric String, Missing value: ”.”
6. QUAL:
    - Phred-scaled quality scores for the assertion made in ALT.
    - If ALT is ”.” (no variant) then this is -10log_10 p(variant) and if ALT is not ”.” this is -10log_10 p(no variant).
    - High QUAL scores indicate high confidence calls.
    - Although traditionally people use integer phred scores, this field is permitted to be a floating point so to enable higher resolution for low confidence calls if desired.
    - Numeric, Missing Value: -1
7. FILTER filter:
    - 0 if this position has passed all filters
    - If site not passed all filters, a semicolon-separated list of codes for filters that fail.
    - Alphanumeric String, Missing Value: ”.”
8. INFO additional information:
    - Alphanumeric String, Missing Value: ”.”
    - Encoded as a semicolon-separated series of short keys with optional values in the format: <key>=<data>[,data]. The subfields could be e.g.:
        - AA ancestral allele
        - AC allele count in genotypes, for each ALT allele, in the same order as listed
        - AN total number of alleles in called genotypes
        - AF allele frequency for each ALT allele in the same order as listed: use this when estimated from primary data, not called genotypes
        - DP combined depth across samples, e.g. D=154
        - MQ RMS mapping quality, e.g. MQ=52
        - NS Number of samples with data
        - BQ RMS base quality at this position
        - SB strand bias at this position
        - DB dbSNP membership
        - H2 membership in hapmap2
        - etc. The exact format of each INFO subfield should be specified in the meta-information.
    - It is not necessary to list all the properties that a site does NOT have, by e.g. H2=0.

Genotype fields:

- If genotype information is present, then the same types of data must be present for all samples.
- First a FORMAT field is given specifying the data types and order.

- This is followed by one field per sample, with the colon-separated data in this field corresponding to the types specified in the format.
- The first subfield must always be the genotype (GT)

- There are five core data types, which are defined as follows:
  - GT genotype (mandatory):
    - encoded as alleles values separated by one of ”/”, “|”, and “\”
    - e.g.: The allele values are 0 for the reference allele (what is in the reference sequence), 1 for the first allele listed in ALT, 2 for the second allele list in ALT and so on. For diploid calls examples could be 0/1 or 1|0 etc.
    - For haploid calls (Y, male X, mitochondrion) only one allele value should be given.
    - missing allele: ”.” (e.g.: ./. for a diploid).
    - The meanings of the separators are:
      '/' : genotype unphased
      '|' : genotype phased
      '\' : genotype phased but switch probability is high past this CHR/POS due to e.g. recombination hotspot.
  - GQ genotype quality:
    - encoded as a phred quality (genotype call is wrong)
    - max quality 99
    - Integer, Missing value: -1
  - DP:
    - read depth at this position for this sample
    - Integer, Missing value: -1
  - HQ haplotype qualities:
    - two phred qualities comma separated
    - max quality 99
    - Integer, Missing value: -1 for each quality. e.g. ”-1,-1”
  - FT:
    - sample genotype filter indicating if this genotype was “called” (similar in concept to the FILTER record for the entire CHROM/POS)
    - 0: indicate that all filters have been passed
    - a semi-colon separated list of codes for filters that fail
    - ”.”: indicate that filters have not been applied.
    - These values should be described in the meta-information in the same way as FILTERs
    - Alphanumeric String, Missing value: ”.”
  - Additional Genotype fields can be defined in the meta-information

Example:

```
##fileformat=VCFv4.0
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=1000GenomesPilot-NCBI36
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID        REF   ALT      QUAL FILTER  INFO
       FORMAT             NA00001             NA00002                 NA00003
20     14370    rs6054257 G     A        29   PASS    NS=3;DP=14;AF=0.5;DB;H2
       GT:GQ:DP:HQ        0|0:48:1:51,51      1|0:48:8:51,51         1/1:43:5:.,.
20     17330    .         T     A        3    q10     NS=3;DP=11;AF=0.017
       GT:GQ:DP:HQ        0|0:49:3:58,50      0|1:3:5:65,3           0/0:41:3
20     1110696  rs6040355 A     G,T      67   PASS    NS=2;DP=10;AF=0.333,0.667;AA=T;DB
       GT:GQ:DP:HQ        1|2:21:6:23,27      2|1:2:0:18,2           2/2:35:4
20     1230237  .         T     .        47   PASS    NS=3;DP=13;AA=T
       GT:GQ:DP:HQ        0|0:54:7:56,60      0|0:48:4:51,51         0/0:61:2
20     1234567  microsat1 GTCT  G,GTACT  50   PASS    NS=3;DP=9;AA=G
       GT:GQ:DP           0/1:35:4            0/2:17:2               1/1:40:3
```

10.31.3 Links and References

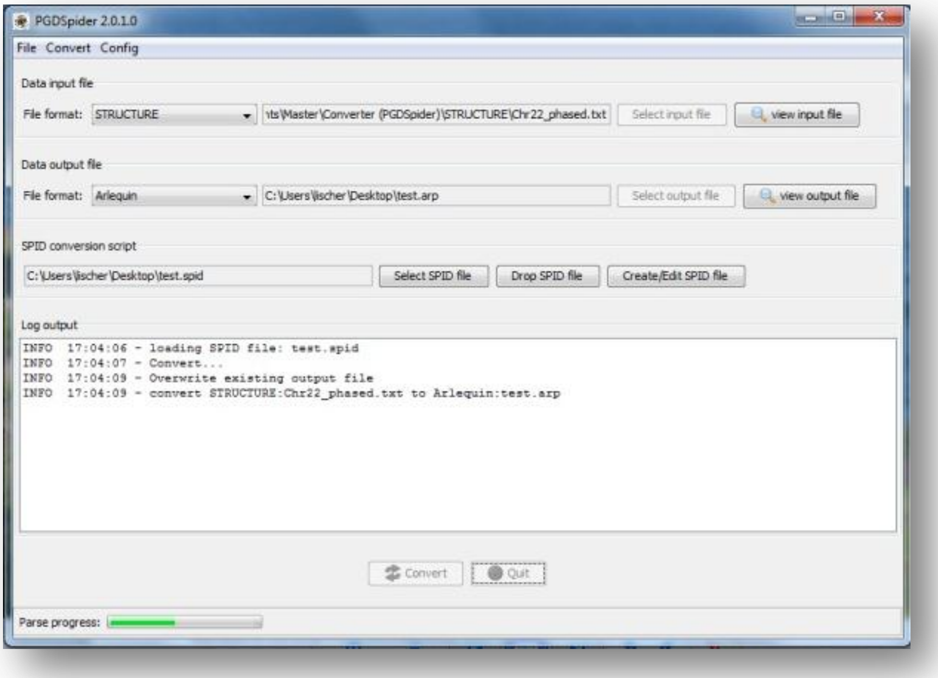
Website: http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-40

10.31.4 Special PGDSpider input/output questions

- Input:
  - What is the ploidy of the data:
    Define if the data are haploid or diploid
  - Only import following regions (optional):
    Defines which regions should be imported. Regions should be defined in following format: refSeqName:start:end, multiple regions: separate it with white spaces
- Output:
  - Save an additional file with reference sequences:
    Saves a file with the reference sequences
  - Specify which data type should be included (optional):
    If the input file contains sequence and SNP data, one has to select which should be included in the output file (only sequence or SNP can be analysed per file).

11 PGDSpider Screenshots

- PGDSpider while it is converting. At the bottom of the GUI the progress of the parser can be seen in the progress bar and during the writing process a waiting symbol appears:



File format: STRUCTURE

ts\Master\Converter (PGDSpider)\STRUCTURE\Chr22_phased.txt

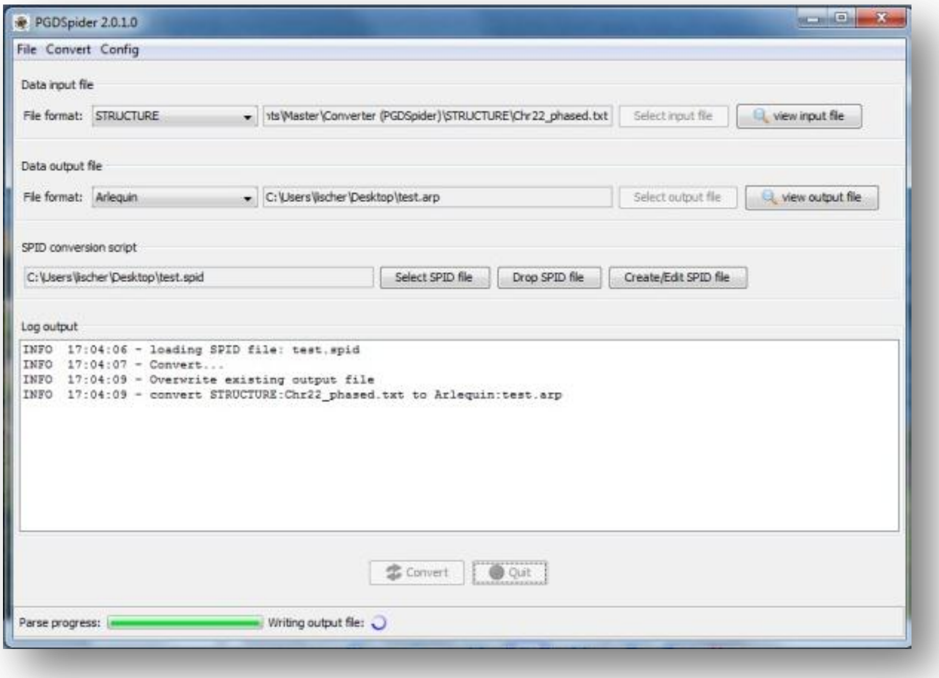
File format: Arlequin

C:\Users\lischer\Desktop\test.arp

C:\Users\lischer\Desktop\test.spid

```
INFO  17:04:06 - loading SPID file: test.spid
INFO  17:04:07 - Convert...
INFO  17:04:09 - Overwrite existing output file
INFO  17:04:09 - convert STRUCTURE:Chr22_phased.txt to Arlequin:test.arp
```

Fig. 10: Screenshot of the PGDSpider GUI during conversion. The parser progress is visible at the bottom left.



File format: STRUCTURE

ts\Master\Converter (PGDSpider)\STRUCTURE\Chr22_phased.txt

File format: Arlequin

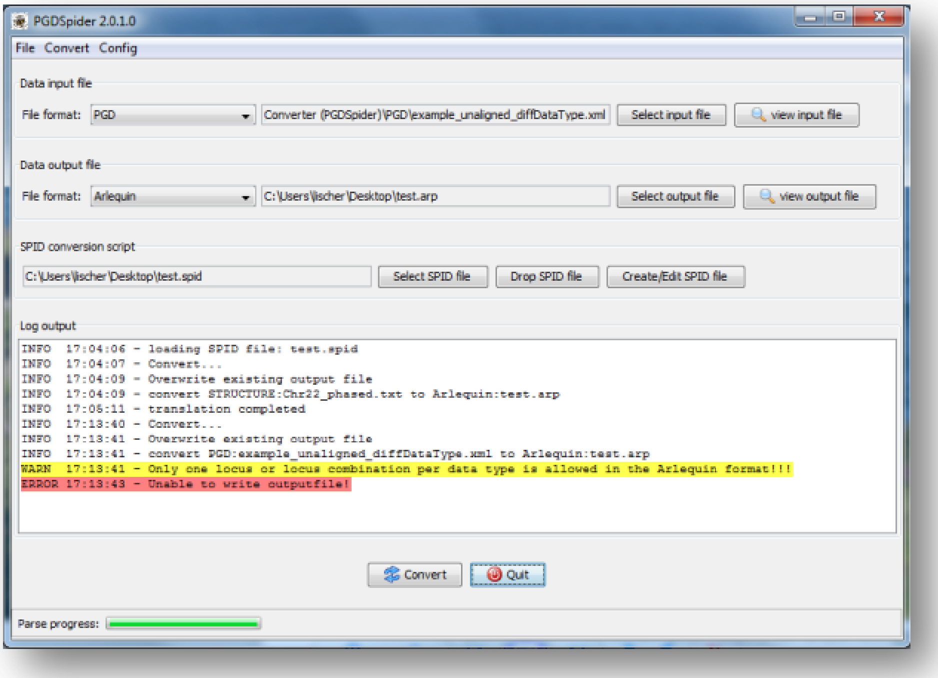
C:\Users\lischer\Desktop\test.arp

C:\Users\lischer\Desktop\test.spid

```
INFO  17:04:06 - loading SPID file: test.spid
INFO  17:04:07 - Convert...
INFO  17:04:09 - Overwrite existing output file
INFO  17:04:09 - convert STRUCTURE:Chr22_phased.txt to Arlequin:test.arp
```

Fig. 11: Screenshot of the PGDSpider GUI during conversion. The writing of the output file is shown by awaiting symbol

- PGDSpider GUI with “WARNING” and “ERROR” messages in the “Log Output”:



Data input file

File format: PGD Converter (PGDSpider)\PGD\example_unaligned_diffDataType.xml

Data output file

File format: Arlequin C:\Users\lischer\Desktop\test.arp

SPID conversion script

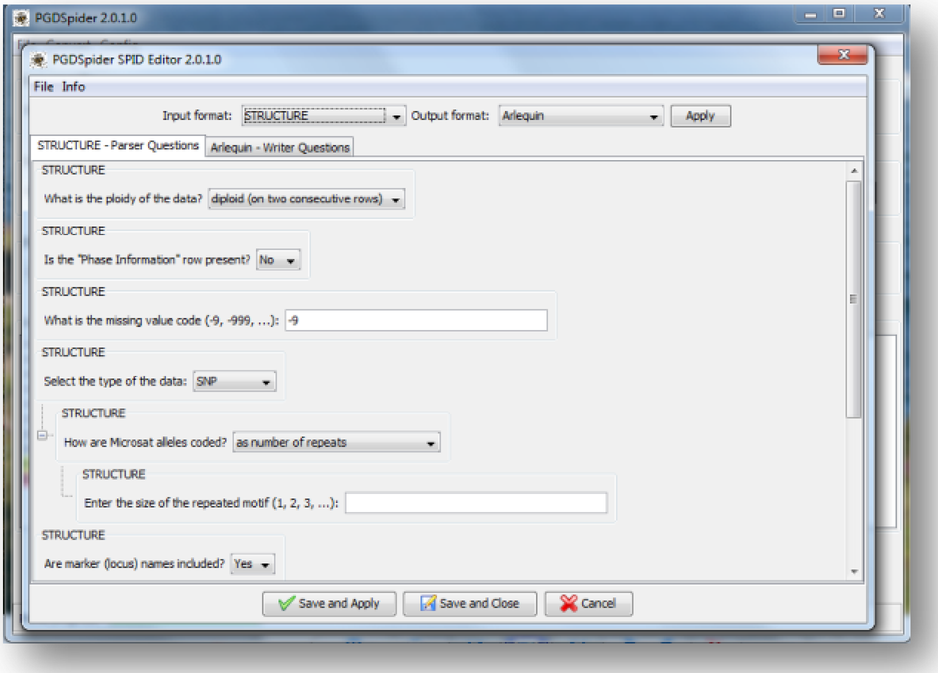
C:\Users\lischer\Desktop\test.spid

Log output

```
INFO  17:04:06 - loading SPID file: test.spid
INFO  17:04:07 - Convert...
INFO  17:04:09 - Overwrite existing output file
INFO  17:04:09 - convert STRUCTURE:Chr22_phased.txt to Arlequin:test.arp
INFO  17:05:11 - translation completed
INFO  17:13:40 - Convert...
INFO  17:13:41 - Overwrite existing output file
INFO  17:13:41 - convert PGD:example_unaligned_diffDataType.xml to Arlequin:test.arp
WARN  17:13:41 - Only one locus or locus combination per data type is allowed in the Arlequin format!!!
ERROR 17:13:43 - Unable to write outputfile!
```

Fig. 12: Screenshot of the PGDSpider GUI with WARNING and ERROR messages within the log output

- PGDSpider GUI during conversion with the SPID Editor containing questions concerning a file format:


PGDSpider 2.0.1.0
PGDSpider SPID Editor 2.0.1.0
File Info
Apply
STRUCTURE - Parser Questions
Arlequin - Writer Questions
Save and Apply
Save and Close
Cancel

Input format: STRUCTURE Output format: Arlequin

STRUCTURE
What is the ploidy of the data? diploid (on two consecutive rows)

STRUCTURE
Is the "Phase Information" row present? No

STRUCTURE
What is the missing value code (-9, -999, ...): -9

STRUCTURE
Select the type of the data: SNP

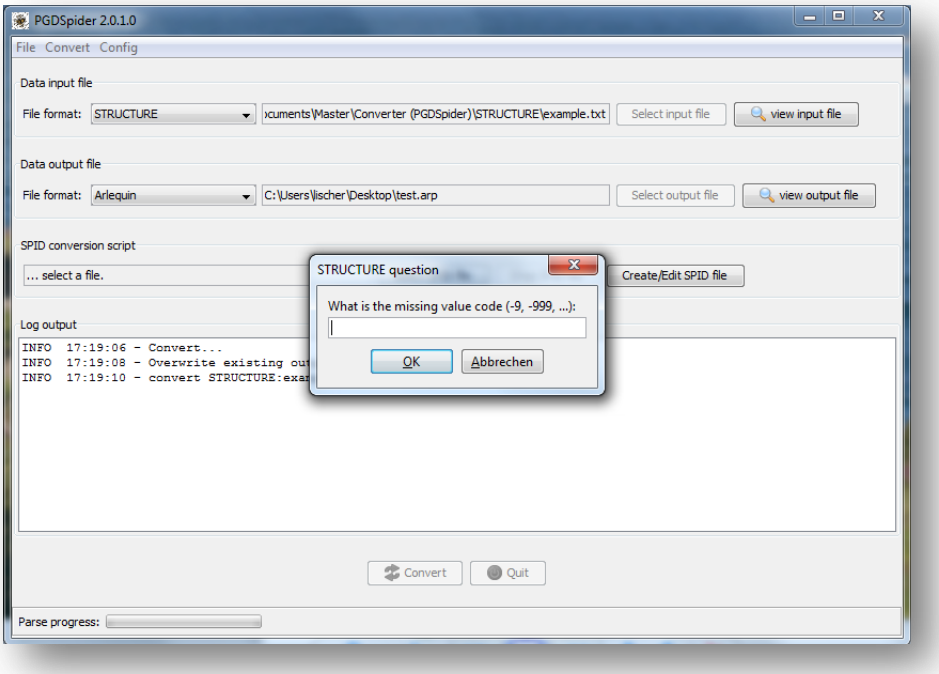
STRUCTURE
How are Microsat alleles coded? as number of repeats

STRUCTURE
Enter the size of the repeated motif (1, 2, 3, ...):

STRUCTURE
Are marker (locus) names included? Yes

Fig. 13: Screenshot of the SPID Editor with STRUCTURE Parser Questions.

- PGDSpider GUI during conversion with a question concerning a file format:



Data input file – File format: STRUCTURE – ›cuments\Master\Converter (PGDSpider)\STRUCTURE\example.txt

Data output file – File format: Arlequin – C:\Users\lischer\Desktop\test.arp

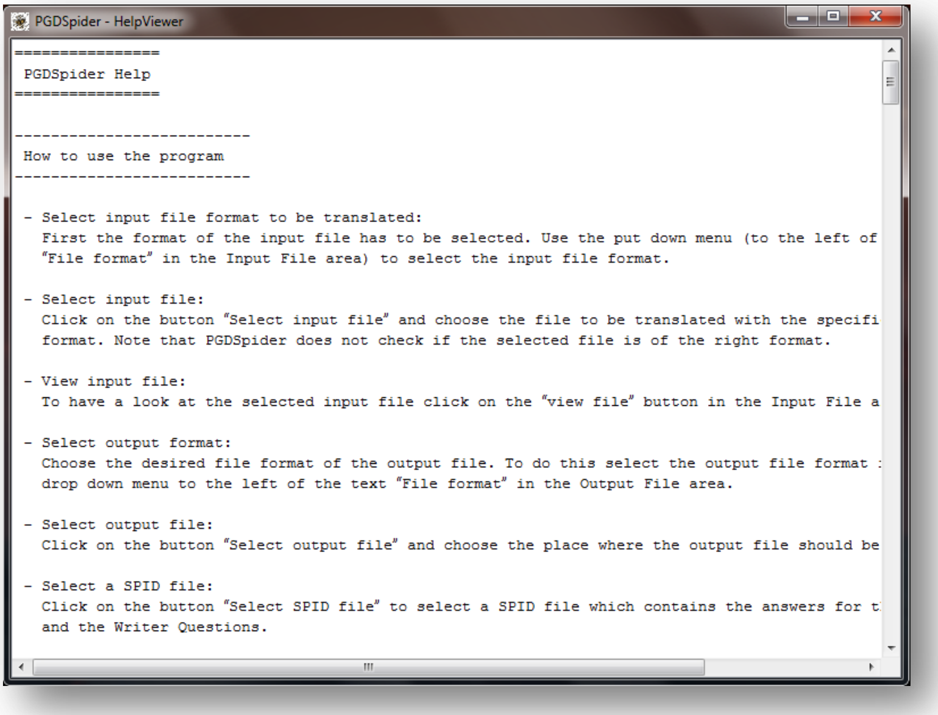
SPID conversion script: … select a file.

STRUCTURE question: What is the missing value code (-9, -999, ...):

```
INFO  17:19:06 - Convert...
INFO  17:19:08 - Overwrite existing out
INFO  17:19:10 - convert STRUCTURE:exa
```

Fig. 14: Screenshot of the PGDSpider GUI with a question concerning the STRUCTURE file format

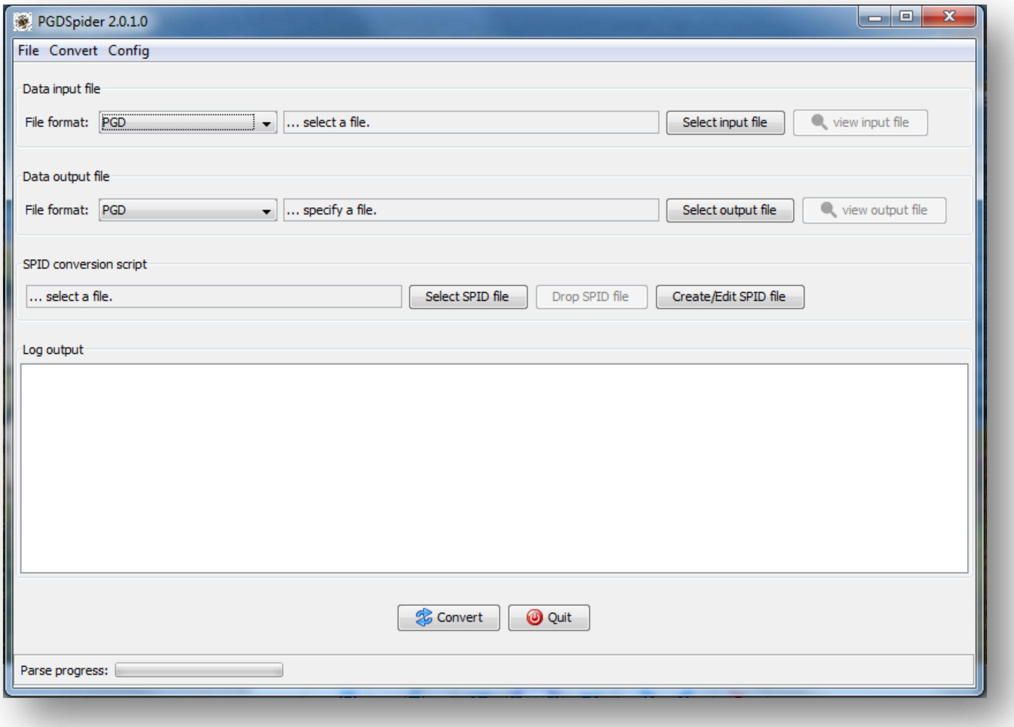
- English help file, found in the PGDSpider GUI "Info" menu under "Help":



```
================
 PGDSpider Help
================

--------------------------
 How to use the program
--------------------------

 - Select input file format to be translated:
   First the format of the input file has to be selected. Use the put down menu (to the left of
   "File format" in the Input File area) to select the input file format.

 - Select input file:
   Click on the button "Select input file" and choose the file to be translated with the specifi
   format. Note that PGDSpider does not check if the selected file is of the right format.

 - View input file:
   To have a look at the selected input file click on the "view file" button in the Input File a

 - Select output format:
   Choose the desired file format of the output file. To do this select the output file format
   drop down menu to the left of the text "File format" in the Output File area.

 - Select output file:
   Click on the button "Select output file" and choose the place where the output file should be

 - Select a SPID file:
   Click on the button "Select SPID file" to select a SPID file which contains the answers for t
   and the Writer Questions.
```

Fig. 15: Screenshot of the English help file.

- PGDSpider graphical user interfaces in different languages:

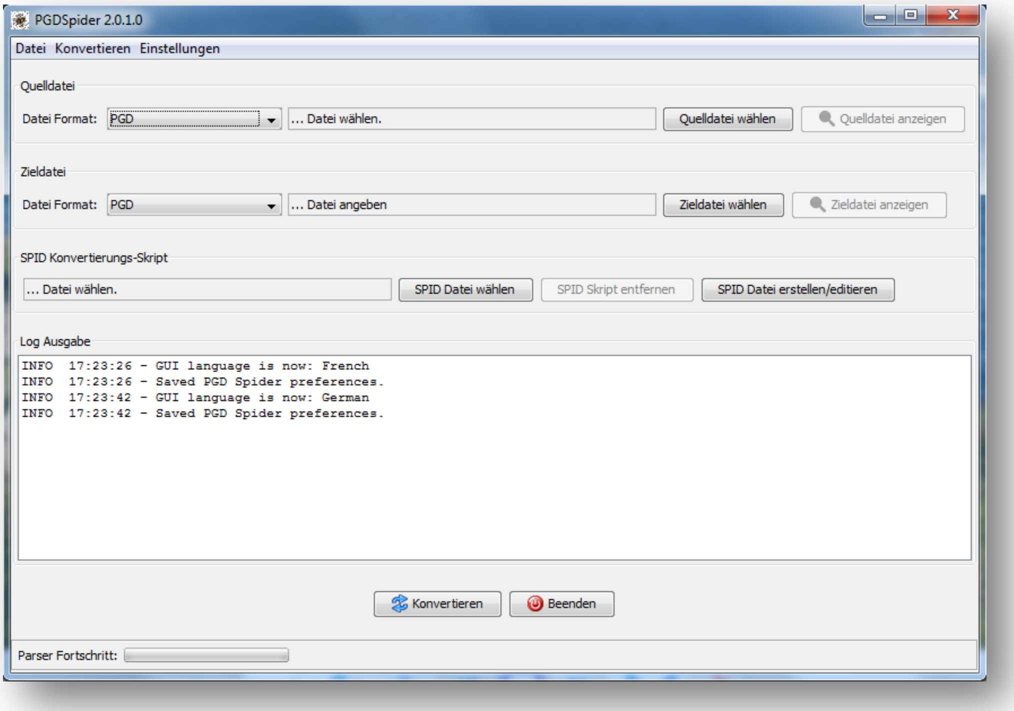


File format: PGD
... select a file.

File format: PGD
... specify a file.

... select a file.

Fig. 16: English version of the PGDSpider GUI



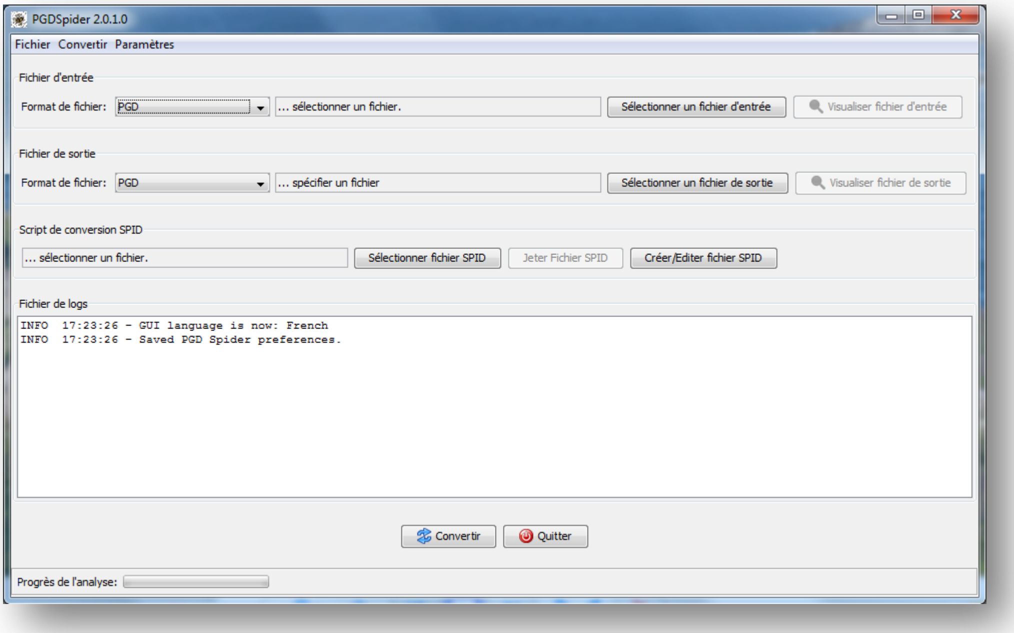
Datei Format: PGD
... Datei wählen.

Datei Format: PGD
... Datei angeben

... Datei wählen.

```
INFO  17:23:26 - GUI language is now: French
INFO  17:23:26 - Saved PGD Spider preferences.
INFO  17:23:42 - GUI language is now: German
INFO  17:23:42 - Saved PGD Spider preferences.
```

Fig. 17: German version of the PGDSpider GUI


PGDSpider 2.0.1.0
Fichier Convertir Paramètres
Fichier d'entrée
Sélectionner un fichier d'entrée
Visualiser fichier d'entrée
Fichier de sortie
Sélectionner un fichier de sortie
Visualiser fichier de sortie
Script de conversion SPID
Sélectionner fichier SPID
Jeter Fichier SPID
Créer/Editer fichier SPID
Fichier de logs
Convertir
Quitter
Progrès de l'analyse:

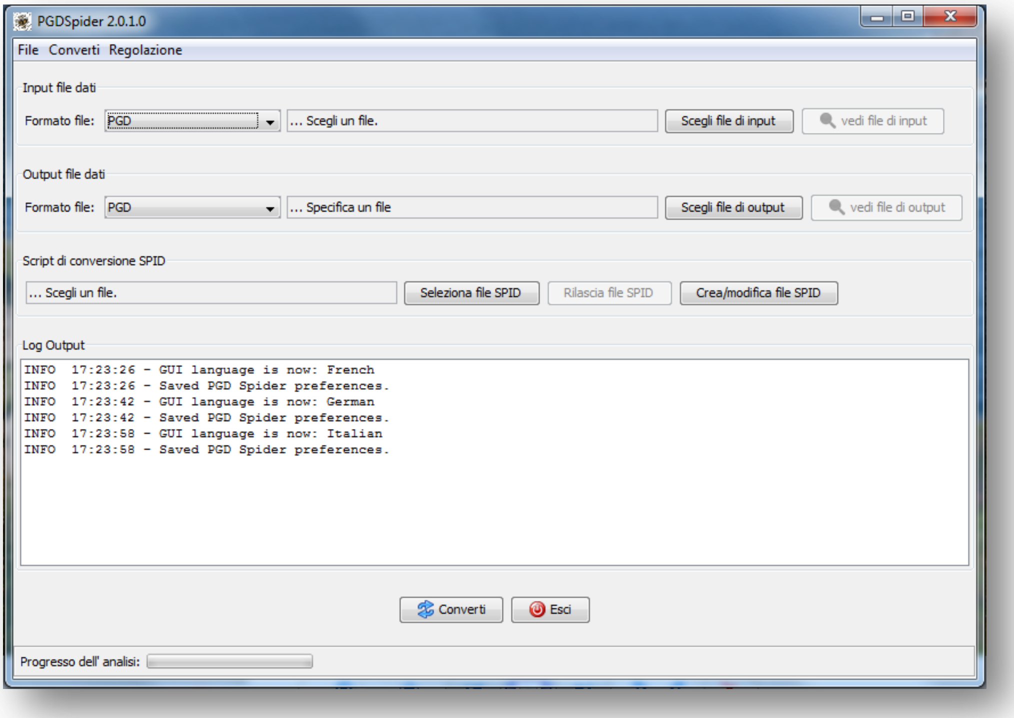
Format de fichier: PGD ... sélectionner un fichier.

Format de fichier: PGD ... spécifier un fichier

... sélectionner un fichier.

```
INFO  17:23:26 - GUI language is now: French
INFO  17:23:26 - Saved PGD Spider preferences.
```

Fig. 18: French version of the PGDSpider GUI



Formato file: PGD ... Scegli un file.

Formato file: PGD ... Specifica un file

... Scegli un file.

```
INFO  17:23:26 - GUI language is now: French
INFO  17:23:26 - Saved PGD Spider preferences.
INFO  17:23:42 - GUI language is now: German
INFO  17:23:42 - Saved PGD Spider preferences.
INFO  17:23:58 - GUI language is now: Italian
INFO  17:23:58 - Saved PGD Spider preferences.
```

Fig. 19: Italian version of the PGDSpider GUI

12 References (Literature)

Anderson, E.C. and Thompson, E.A. (2002) A model-based method for identifying species hybrids using multilocus genetic data, *Genetics*, **160**, 1217-1229.

Beaumont, M.A. (1999) Detecting population expansion and decline using microsatellites, *Genetics*, **153**, 2013-2029.

Beaumont, M.A. and Nichols, R.A. (1996) Evaluating loci for use in the genetic analysis of population structure, *Proceedings of the Royal Society of London Series B-Biological Sciences*, **263**, 1619-1626.

Beerli, P. (2006) Comparison of Bayesian and maximum-likelihood inference of population genetic parameters, *Bioinformatics*, **22**, 341-345.

Beerli, P. (2008) Migrate version 3.0 - a maximum likelihood and Bayesian estimator of gene flow using the coalescent. Distributed over the internet at http://popgen.scs.edu/migrate.html.

Beerli, P. (2009) How to use migrate or why are markov chain monte carlo programs difficult to use?, *Conservation Biology*, **17**.

Beerli, P. and Felsenstein, J. (1999) Maximum-likelihood estimation of migration rates and effective population numbers in two populations using a coalescent approach, *Genetics*, **152**, 763-773.

Beerli, P. and Felsenstein, J. (2001) Maximum likelihood estimation of a migration matrix and effective population sizes in n subpopulations by using a coalescent approach, *Proceedings of the National Academy of Sciences of the United States of America*, **98**, 4563-4568.

Belkhir, K., Borsa P., Chikhi L., Raufaste N., Bonhomme F. (1996-2004) GENETIX 4.05, logiciel sous Windows TM pour la génétique des populations. Laboratoire Génome, Populations, Interactions, CNRS UMR 5171, Université de Montpellier II, Montpellier (France).

Cann, H.M., et al. (2002) A human genome diversity cell line panel, *Science*, **296**, 261-262.

Cavalli-Sforza, L.L. (2005) Opinion - The Human Genome Diversity Project: past, present and future, *Nature Reviews Genetics*, **6**, 333-340.

Cock, P.J.A., et al. (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants, *Nucleic Acids Res*, **38**, 1767-1771.

Dieringer, D. and Schlotterer, C. (2003) MICROSATELLITE ANALYSER (MSA): a platform independent analysis tool for large microsatellite data sets, *Molecular Ecology Notes*, **3**, 167-169.

Excoffier, L. and Lischer, H.E.L. (2010) Arlequin suite ver 3.5: a new series of programs to perform population genetics analyses under Linux and Windows, *Mol Ecol Resour*, **10**, 564-567.

Falush, D., et al. (2003) Inference of population structure using multilocus genotype data: Linked loci and correlated allele frequencies, *Genetics*, **164**, 1567-1587.

Falush, D., et al. (2007) Inference of population structure using multilocus genotype data: dominant markers and null alleles, *Molecular Ecology Notes*, **7**, 574-578.

Felsenstein, J. (1989) PHYLIP - Phylogeny Inference Package (Version 3.2), *Cladistics*, **5**, 164-166.

Felsenstein, J. (2004) PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

Fischer, M.C., et al. (2011) Enhanced AFLP genome scans detect local adaptation in high-altitude populations of a small rodent (Microtus arvalis), *Molecular Ecology*, **20**, 1450-1462.

Flint, J., et al. (1999) Minisatellite mutational processes reduce F-st estimates, *Human Genetics*, **105**, 567-576.

Foll, M., et al. (2010) Estimating population structure from AFLP amplification intensity, *Molecular Ecology*, **19**, 4638-4647.

Foll, M. and Gaggiotti, O. (2006) Identifying the environmental factors that determine the genetic structure of Populations, *Genetics*, **174**, 875-891.

Foll, M. and Gaggiotti, O. (2008) A Genome-Scan Method to Identify Selected Loci Appropriate for Both Dominant and Codominant Markers: A Bayesian Perspective, *Genetics*, **180**, 977-993.

Glaubitz, J.C. (2004) CONVERT: A user-friendly program to reformat diploid genotypic data for commonly used population genetic software packages, *Molecular Ecology Notes*, **4**, 309-310.

Google (2009)

Goudet, J. (2001) FSTAT, a program to estimate and test gene diversities and fixation indices (version 2.9.3). .

Guillot, G. (2008) Inference of structure in subdivided populations at low levels of genetic differentiation-the correlated allele frequencies model revisited, *Bioinformatics*, **24**, 2222-2228.

Guillot, G., et al. (2005) A spatial statistical model for landscape genetics, *Genetics*, **170**, 1261-1280.

Guillot, G., et al. (2005) GENELAND: a computer package for landscape genetics, *Molecular Ecology Notes*, **5**, 712-715.

Guillot, G. and Santos, F. (2009) A computer program to simulate multilocus genotype data with spatially autocorrelated allele frequencies, *Mol Ecol Resour*, **9**, 1112-1120.

Guillot, G. and Santos, F. (2010) Using AFLP markers and the Geneland program for the inference of population genetic structure, *Mol Ecol Resour*, **10**, 1082-1084.

Guillot, G., et al. (2008) Analysing georeferenced population genetics data with Geneland: a new algorithm to deal with null alleles and a friendly graphical user interface, *Bioinformatics*, **24**, 1406-1407.

Hasegawa, M., et al. (1985) Dating of the Human Ape Splitting by a Molecular Clock of Mitochondrial-DNA, *Journal of Molecular Evolution*, **22**, 160-174.

Hey, J. and Nielsen, R. (2004) Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of Drosophila pseudoobscura and D-persimilis, *Genetics*, **167**, 747-760.

Hey, J. and Nielsen, R. (2007) Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics, *Proceedings of the National Academy of Sciences of the United States of America*, **104**, 2785-2790.

Hubisz, M.J., et al. (2009) Inferring weak population structure with the assistance of sample group information, *Mol Ecol Resour*, **9**, 1322-1332.

Kimura, M. (1969) Number of Heterozygous Nucleotide Sites Maintained in a Finite Population Due to Steady Flux of Mutations, *Genetics*, **61**, 893-&.

Kimura, M. and Ohta, T. (1978) Stepwise Mutation Model and Distribution of Allelic Frequencies in a Finite Population, *Proceedings of the National Academy of Sciences of the United States of America*, **75**, 2868-2872.

Lewis, P.O., D. Zakin (2001) Genetic Data Analysis: Computer program for the analysis of allelic data. Version 1.0 (d16c). .

Li, H., et al. (2009) The Sequence Alignment/Map format and SAMtools, *Bioinformatics*, **25**, 2078-2079.

Maddison, D.R., et al. (1997) Nexus: An extensible file format for systematic information, *Systematic Biology*, **46**, 590-621.

Nielsen, R. and Wakeley, J. (2001) Distinguishing migration from isolation: A Markov chain Monte Carlo approach, *Genetics*, **158**, 885-896.

Palsboll, P.J., et al. (2004) Discerning between recurrent gene flow and recent divergence under a finite-site mutation model applied to North Atlantic and Mediterranean Sea fin whale (Balaenoptera physalus) populations, *Evolution*, **58**, 670-675.

Pearson, W.R. (1990) RAPID AND SENSITIVE SEQUENCE COMPARISON WITH FASTP AND FASTA, *Methods in Enzymology*, **183**, 63-98.

Pritchard, J.K., et al. (2000) Inference of population structure using multilocus genotype data, *Genetics*, **155**, 945-959.

Rannala, B. and Mountain, J.L. (1997) Detecting immigration by using multilocus genotypes, *Proceedings of the National Academy of Sciences of the United States of America*, **94**, 9197-9201.

Raymond, M. and Rousset, F. (1995) Genepop (Version-1.2) - Population-Genetics Software for Exact Tests and Ecumenicism, *Journal of Heredity*, **86**, 248-249.

Rousset, F. (2008) GENEPOP ' 007: a complete re-implementation of the GENEPOP software for Windows and Linux, *Mol Ecol Resour*, **8**, 103-106.

Tamura, K., et al. (2007) MEGA4: Molecular evolutionary genetics analysis (MEGA) software version 4.0, *Molecular Biology and Evolution*, **24**, 1596-1599.

Tamura, K., et al. (2011) MEGA5: Molecular Evolutionary Genetics Analysis using Maximum Likelihood, Evolutionary Distance, and Maximum Parsimony Methods., *Molecular Biology and Evolution*.

Tang, J., et al. (2009) Identifying Currents in the Gene Pool for Bacterial Populations Using an Integrative Approach, *Plos Comput Biol*, **5**, **-**.

W3Schools (2008)

Waterston, R.H., et al. (2002) Initial sequencing and comparative analysis of the mouse genome, *Nature*, **420**, 520-562.

Wilson, G.A. and Rannala, B. (2003) Bayesian inference of recent migration rates using multilocus genotypes, *Genetics*, **163**, 1177-1191.

Wilson, I.J., et al. (2003) Inferences from DNA data: population histories, evolutionary processes and forensic match probabilities, *Journal of the Royal Statistical Society Series a-Statistics in Society*, **166**, 155-188.