CraneFoot v3.2 user's guide

Ville-Petteri Mäkinen
Laboratory of Computational Engineering
Helsinki University of Technology

December 6, 2006

Getting started

Windows installation

! CraneFoot is a command line software and as such not very well suited for users who are accustomed to graphical user interfaces. There is, however, a web based interface on the CraneFoot home page that makes the drawing of pedigrees easier, but for practical reasons this option might not offer the same features as a proper installation.

1. Unpack the installation package using WinZip or equivalent (usually it is enough to just double-click the icon). It is highly recommended that the unpacked executable *cranefoot.exe* and the exemplary data files are are at this point copied to a folder such as 'c:\cf3\' so that they are easy to access from the command prompt. You can find these files from the 'win32' and 'example' folders within the installation package.

2. Open the command prompt from the accessories leaf in the start menu. Now proceed to the directory where you unpacked the executable and test CraneFoot by typing[1]

```
cd c:\cf3\
.\cranefoot config.txt
```

[1] Press the return key after both lines to invoke the commands.

You should get two new files into the current directory: a PostScript file *myPedigree.ps* and a text file *myPedigree.topology.txt*.

3. Next, create a new directory to another location, for example 'c:\cf_test\'. Copy the two main input files *config.txt* and *pedigree.txt* to this directory and then type

```
cd c:\cf_test\
c:\cf3\cranefoot config.txt
```

Again, you should get the same output files. You can use the previous command in any directory and supply any configuration file you want, but make sure all the files you list in the configuration file are accessible. Sometimes you may need to use full paths. For instance, suppose your data files are in 'c:\cfiles\'. You could then write the instructions

```
PedigreeFile  c:\cfiles\pedigree.txt
PedigreeName  c:\cfiles\myPedigree
```

in your configuration file, the precise role of which is to be discussed in detail later. Note that since the instructions are white space separated, the paths must not contain white space characters. It might also be necessary not to use directory names over 8 characters long.

- There is a more elegant way of calling MS-DOS programs, and if the above method seems cumbersome, try searching the Internet for "path windows", it should give enough links to relevant pages.
- You should be able to cope with just two prompt commands: `cd target` changes directory to `target` and `dir` shows the contents of the current directory.
- The biggest problems in Windows come from the long and often complicated file and directory names. Use `dir` to see a shorter version of a file name that you can use as part of your commands. You can also open the properties menu (right-click on an icon) to look at the file path that you can type in the prompt. If your file path contains space characters, surround it with double quotes. For example, if you have 'c:\pedigree project 1\', type

```
cd "c:\pedigree project 1\"
```

to enter the directory.

Linux/UNIX installation

Before you can start using CraneFoot on other than Windows systems, you have to unpack the source code and combile it to a binary executable. In most UNIX-based systems the compiler software is already installed. CraneFoot is written in C/C++ and usual commands for a C++ compiler include `c++` and `g++`, the latter is associated with the GNU compiler which is usually a safe choice. To make the executable, unpack the installation package (`unzip` or similar), go to the 'source' directory and type

```
g++ -O5 -o cranefoot main.cc *.cpp -lm
```

to invoke the compiler. When it finishes, you should have a new file *cranefoot* in the directory. To test the software, move the file to the 'example' directory and type

```
./cranefoot config.txt .
```

You should get two new files: a PostScript file *myPedigree.ps* and a text file *myPedigree.topology.txt*.

CraneFoot is a stand-alone executable, with no links to shared or dynamic libraries. Furthermore, you do not need any other files to run the program. If you have root privileges, you can copy the executable that you compiled to a location that is part of the execution path (e.g. /usr/local/bin). Otherwise you can ask the system administrator to do that for you. You can also use the executable directly through the full path: If it is in the current directory, just type `./cranefoot` or if it is in your home directory (e.g. /home/username) type `/home/username/cranefoot`.

Basic use

CraneFoot is designed to be compatible with the so called linkage format, where family relations are represented by a set of child-father-mother triplets. As a result, the natural format for the pedigree file is tabulated text with three columns, denoted by `NameVariable`, `FatherVariable` and `MotherVariable`. One of the goals in the development of CraneFoot was to create a software with as

little manual formatting of the input files as possible. Hence the order and indeed the location of the columns in the file is irrelevant, CraneFoot can detect them by itself if properly instructed.

Instructions are the human interface of CraneFoot. In fact, you do not submit the pedigree data directly to the software, rather you give it a list of instructions that specify where to find the data, how to read it and where to store the results. The only requirement is that the data file is tabulated by a delimiter character, and the first row contains unique names for each column – a common description of a spreadsheet.

To draw a pedigree, you need at least five instructions:

```
PedigreeFile        pedigree.txt
PedigreeName        myPedigree
NameVariable        NAME
FatherVariable      FATHER
MotherVariable      MOTHER
```

`PedigreeFile` contains the file path to the primary input file. In this case it is assumed that the file is in the current working directory, hence the file name alone should suffice. Note, however, that depending on your platform (MS Windows in particular) you may have to list the full path regardless of the file location. As to `PedigreeName`, CraneFoot needs to know where to save the results, therefore you must provide a name for your pedigree that is used as a template for naming the output files.

At this point the most important thing to note is the freedom of choosing the name, father and mother variables . Although here the columns are named 'NAME', 'FATHER' and 'MOTHER', they could be named differently, for example 'ID', 'FatID', and 'MotID' if so desired. Even more significant is the lack of fixed location, the mother column (`MotherVariable`) could be the first, third or last column – it makes no difference to CraneFoot as long as the column heading is consistent with the one provided in the configuration file. The only restriction is that a heading must not contain white space characters since the configuration file must be white space delimited.

Assuming you have successfully installed and tested CraneFoot (go back to the first section if not) you should already have the output files *myPedigree.ps* and

myPedigree.topology.txt available. The topology file is a printout of the logical relations between the individuals. It also contains the symbol coordinates, which is necessary if you want to draw the figure yourself based on the layout from CraneFoot.

The PostScript file with the suffix .ps has three parts: First, a table of contents shows every successfully drawn subgraph of the entire data set, then a list of errors is printed and finally the figures themselves on separate pages. In this case, we have just one subgraph with a generic name 'family' (we will shortly discuss how to partition the data into separate families). Furthermore, there were no errors, and thus the second part is omitted. The result is depicted on Figure 1[2].

Sometimes it is useful to have a clean image of each subgraph in separate output files. The encapsulated Postscript files (.eps) depict only the subgraphs themselves without any extra information, and you can use `FigureLimit` and `PageSize` to control the process.

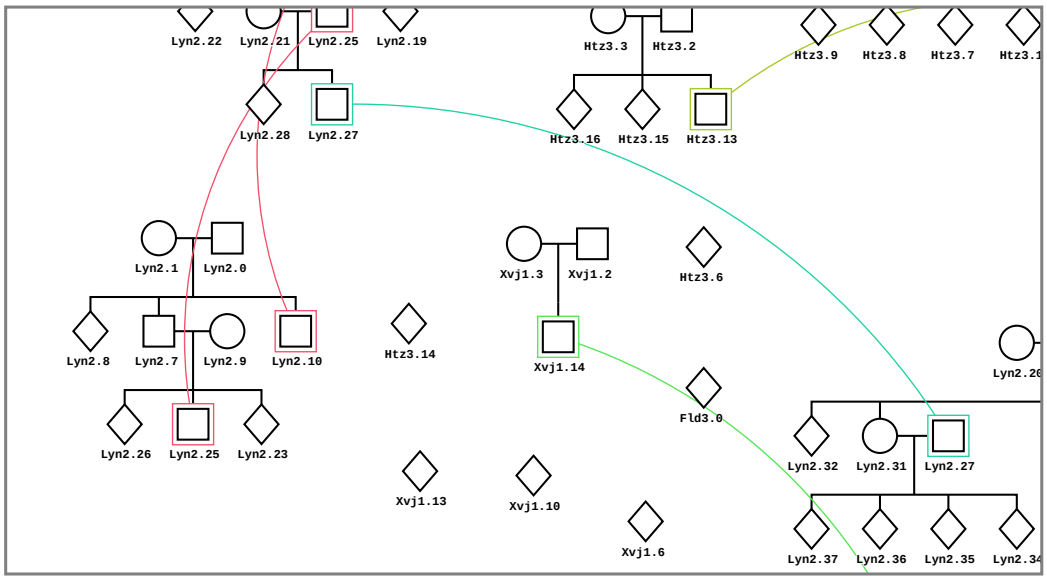

Figure 1: A clip from the basic drawing of the sample pedigree in *pedigree.txt*. Notice that CraneFoot automatically sets parents' gender based on topology even if no gender information is available.

In genetic studies it is often the case that you have a number of small pedigrees of

[2]The figure may not be exactly what you got due to the indeterministic layout algorithm.

unrelated families. It is therefore important to treat them separately and, for that purpose, you can use the `SubgraphVariable` to name each family and to draw them on separate pages. Try removing the comment character '#' at the subgraph instruction in the sample file *config.txt* and running CraneFoot again, you should get a rewritten output document with multiple pages. Note that a subgraph of one individual is not considered a proper family.

Using multiple input files

Before going any further, it must be emphasized that you need not put your data in multiple input files, you can use just one. This feature is optional and intended for cases where the user wants to integrate information from multiple sources when drawing a pedigree. For example, you can have extensive historical knowledge of a pedigree, but you receive clinical information on only a handful of individuals. Often it is prudent to combine the data into a single file, but CraneFoot can do that for you, if necessary.

Previously it was stated that the location of a particular variable (i.e. column) is of no consequence to CraneFoot if the heading is correct. On that note, the flexibility can still be increased if users can define the source file for each variable[3]. In order for this to work, there has to be enough information to associate an individual in the main pedigree file with a row in an auxiliary input file. The only way to accomplish this is to make sure that every input file contains a name column, that is, a column with a heading specified by `NameVariable`. A second condition is that the father and mother variables must be in the pedigree file, this is merely a technical convenience.

Since the whole point of this feature is to add flexibility, it is not mandatory to have the same individuals present in every input file. In fact, whether there are too many or too few individuals in a given input file compared to the pedigree file means only that some data on some individuals is missing – it does not prevent the software from drawing the pedigree. The next section on visualization features contains practical examples of using multiple files.

[3]The default source file is defined by `PedigreeFile`.

Visualization features

As is evident in Figure 1, information about the individuals' sex should be given to CraneFoot so that children could be drawn properly. The mechanism is the same as before: You add a `GenderVariable` instruction to the configuration file that tells which column contains the binary data. You can set the character that corresponds to male (female) sex by using the instruction `Male` (`Female`). The default is 'M' ('F').

The next two important features are the individual symbol color and filling pattern, defined by `ColorVariable` and `PatternVariable`, respectively. A color is defined as a six digit integer, with the first two digits denoting the red component intensity, the next two denoting green and the last two denoting blue. Patterns are also coded as integers, but there is no obvious logic since the patterns themselves are quite different from each other, except that the values range from 1 to 99. Figure 2 depicts some colors and all the available filling patterns.

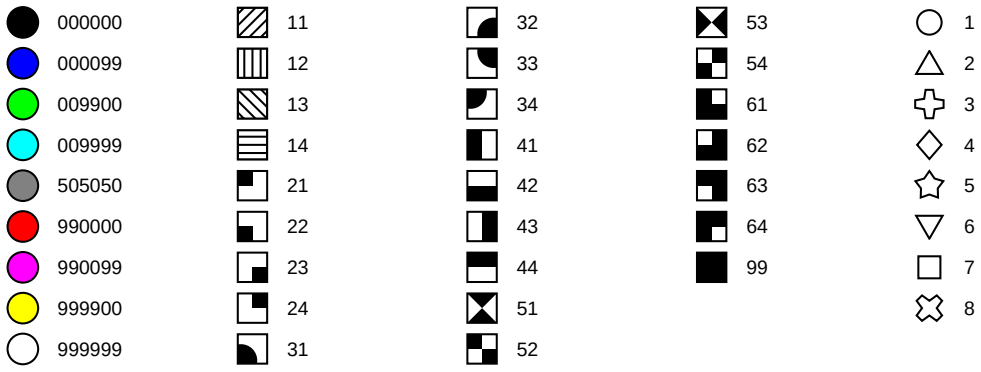

Figure 2: A list of some symbol colors and all pattern and shape codes.

In addition to color and filling pattern, the individuals can be labeled by a diagonal slash (`SlashVariable`, a common glyph for death), by a red arrow head (`ArrowVariable`, used for probands) and by square brackets around the individual's name (`TracerVariable`). All the above are binary in nature, and work similarly: Any positive integer will draw the glyph in question, otherwise it is not drawn.

Finally, textual information can be attached to any individual by one or more `TextVariable`s. Each text variable specifies a data column, and the particular

text segment in the column will be printed below the individual's symbol. CraneFoot is able to adjust the horizontal and vertical space, so you can use any number of segments of any length.

Now that the main features have been covered it is time to put them to practice. First, let's examine some of the instructions in *config.txt* in detail. The `GenderVariable` has two values: the column where the data is stored and the file where the column is stored. By opening the file *phenotypes.txt* you can see that it contains the name column 'NAME' that is needed in identifying the rows and a column named 'GENDER' as specified by the gender variable. Similarly, two of the text variables point to two columns in another auxiliary input file *genotypes.txt*. The order of the lines of text that will be written below the nodes is determined by the order in which the text instructions are given. Therefore, the first `TextVariable` points to the name column, just to ensure that the first piece of text is the individual's name. Also, remember from the previous discussion that it is not necessary to use multiple input files, here it is done just as an example.

Next, try removing the comment character '#' to activate the visualization instructions to see what happens (Figure 3 at the end of the document). It should be clear by now that the instruction interface is a powerful tool to choose different visualizations from a collection of data. For example, you could have several datasets from the same pedigree, each with a different set of individuals and you want to create visualizations from different points of view. Instead of making a new input file for each drawing, you can just change the instructions without manipulating the input files and thus save valuable time.

The third generation of CraneFoot suffers slightly from the indeterministic algorithm that creates somewhat different layouts every time, but in most cases you can circumvent this by using the `RandomSeed` instruction. It will ensure that the positions on the canvas remain the same for each run, provided that the textual features and page properties do not change. Other useful instructions are listed in the last section.

Configuration interface

File parameters

`PedigreeName`

Template name for output files: <*pedigreename*>_<*family*>*.eps* for fam-

ilies <*pedigreename*>*.ps* for pedigree document and <*base*>*.topology.txt* for the topology file.

`PedigreeFile` (required)
Pedigree data. Must contain name. father and mother columns.

Structural parameters

`AgeVariable`
Determine the order of siblings in the pedigree. For a given nuclear family, the oldest siblings are placed on the left.

`FatherVariable` (required in pedigree file)
Father identification.

`GenderVariable`
Gender data. Cannot be used simultaneously with `ShapeVariable`.

`MotherVariable` (required in pedigree file)
Mother identification.

`NameVariable` (required)
Unique name for each individual for identification.

`SubgraphVariable`
Family identification.

Visualization parameters

`ArrowVariable`
Black arrowhead pointing to the node, drawn if integer value greater than 0.

`ColorInfo` (multiple)
Labels for specific colors. Two value fields: code and label. Valid codes within $[000000, 999999]$, see Figure 2.

`ColorVariable`
Individual background color. Valid codes within $[000000, 999999]$, see Figure 2.

`LabelVariable`
Node labels. If none supplied, individual names are used.

`PatternInfo` (multiple)
Labels for specific patterns. Two value fields: code and label. Valid codes within $[1, 99]$, see Figure 2.

`PatternVariable`
Individual patterns. Valid codes within $[1, 99]$, see Figure 2.

`ShapeInfo` (multiple)
Labels for specific shapes. Two value fields: code and label. Valid codes within $[1, 8]$, see Figure 2.

`ShapeVariable`
Shapes of pedigree nodes. Valid codes within $[1, 8]$, see Figure 2. Cannot be used simultaneously with `GenderVariable`.

`SlashVariable`
Diagonal slash, drawn if integer value greater than 0.

`TextVariable` (multiple)
A line of text below a node.

`TracerVariable`
Square brackets around the name, drawn if integer value greater than 0.

Formatting and functional parameters

`Female`
Alphanumeric code for the female sex. The default is 'F'.

`FontSize`
Base font size. The default is 10pt.

`Male`
Alphanumeric code for the male sex. The default is 'M'.

`PageOrientation`
Keyword for page orientation for the output document. Possible choices are 'landspace' and 'portrait'. The default is 'portrait'.

`PageSize`
: Keyword for page sizes for the output documents. Possible choices are 'letter', and 'a0' to 'a5' for the first value (main document). The second value refers to the family outputs (.eps files) with an additional option of automatic size 'auto'. The defaults are 'letter' and 'auto' for the first and second value, respectively.

`VerboseMode`
: If 'off', runtime messages are suppressed. The default is 'on'.

`Delimiter`
: The character that separates the columns in the pedigree file. Use 'tab' for the horizontal tabulator and 'ws' for white space. The default is 'tab'.

`BackgroundColor`
: Six digit color definition for the canvas background. The default is 999999 (white).

`ForegroundColor`
: Six digit color definition for the general line art. The default is 000000 (black).

`TimeLimit`
: Maximum allowed time to spend on layout search. This is not a hard limit: If there are multiple small families or the families are very large, the limit might not hold. The default is 5s per family.

`FigureLimit`
: Maximum number of separate family outputs (.eps files). The default is 0.

`RandomSeed`
: Starting value for the pseudo-random number generator that is used when computing the layouts. If active, `TimeLimit` is ignored.

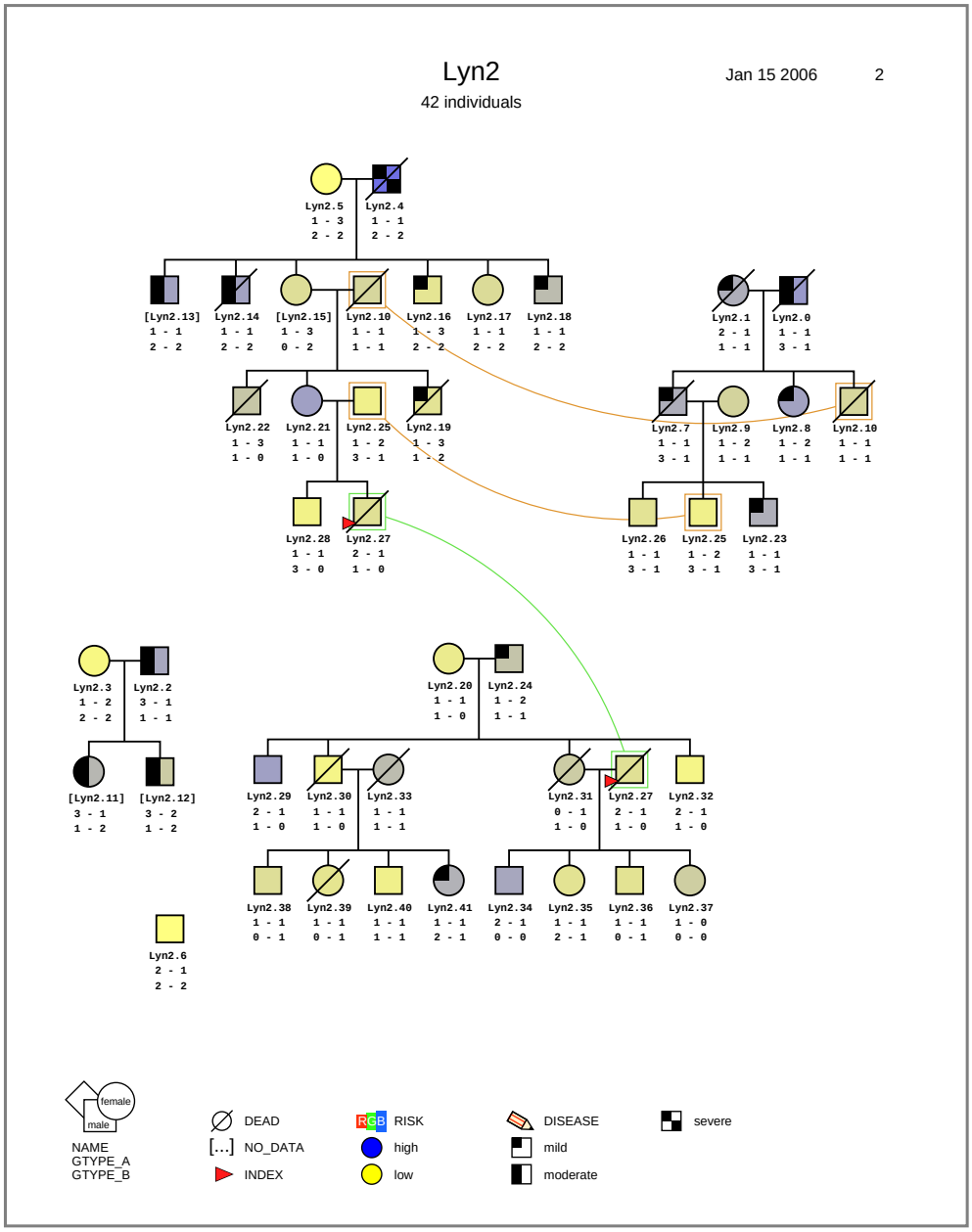

Figure 3: Page layout in CraneFoot's primary output file. The family name and size are on the top with date stamp and page number on the top right. The legend is either on the left or, if it is more space efficient, on the bottom as in this case.