Core Hunter user manual

http://www.corehunter.org

1 Overview

Core Hunter is a library for selecting and analyzing genetic resources. The latest updates can be found on the Core Hunter website: `http://www.corehunter.org`.

2 Building the Core Hunter software

If you have downloaded a binary release of the software, containing prebuilt jar packages, you can skip this section. If you have downloaded a source code archive, read this section for instructions on how to build the source.

2.1 Requirements

Building the Core Hunter software requires Maven and a JDK version 1.6+:

- You can download and install Maven from
  `http://maven.apache.org/download.cgi`
- You can download and install a Java Development Kit (JDK) from
  `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
  The JDK also includes a Java Runtime Environment (JRE) needed to run the software.

2.2 Building the software

Please use the appropriate script to build the Core Hunter software on your machine.

2.2.1 On a Unix/Linux flavored OS

Run

```
./build.sh
```

to build the software. This will create a `bin` subdirectory with compiled jar files of both the `corehunter` and `coreanalyser` tools.

2.2.2 On Windows

If your downloaded source archive contains the Windows build script

```
build.bat
```

run this script to build the software. This will create a `bin` subdirectory with compiled jar files of both the `corehunter` and `coreanalyser` tools.

To build an older release which does not contain the Windows build script yet, run:

```
mvn package
```

In this case no `bin` folder is created so it is required to manually grab the compiled jar files from

```
corehunter-cli/target/corehunter-cli-[VERSION]-SNAPSHOT-jar-with-dependencies.jar
coreanalyser-cli/target/coreanalyser-cli-[VERSION]-SNAPSHOT-jar-with-dependencies.jar
```

after running the Maven build command, replacing `[VERSION]` with the appropriate version number. In this case we strongly advise you to manually create the `bin` folder and to copy the compiled jar files to this location, renaming them to

```
corehunter-cli.jar
coreanalyser-cli.jar
```

for a simplified user experience.

3 Running the Core Hunter software

Currently the advised method for running the Core Hunter software is to use the command-line interface (CLI). We also provide a basic wrapper script to run the CLI from R. In the future we will release a user-friendly graphical interface (GUI) as well as a more flexible R interface. Check the website for updates on this.

3.1 Requirements

Running the Core Hunter software requires Java version 1.6+:

- You can download and install a Java Runtime Environment (JRE) from
  `http://www.java.com/en/download/index.jsp`

3.2 Running the command-line interface

Assuming that your current working directory is the root directory of Core Hunter, and assuming that it contains a `bin` subdirectory with compiled jar files of the CLIs, an example command to run Core Hunter is:

```
java -jar bin/corehunter-cli.jar -MR 0.7 -SH 0.3 data/examples/bul.csv core.csv
```

This uses the default Mixed Replica algorithm to sample a core from the example dataset `bul.csv`, optimizing both Modified Rogers' distance (with weight 0.7) and Shannon's diversity index (with weight 0.3). The sampling intensity defaults to 20% and also the default runtime limit of 60 seconds is applied. The resulting core subset is stored in the file `core.csv`. Alternatively, an optimization algorithm of choice can be selected, e.g.:

```
java -jar bin/corehunter-cli.jar -remc -MR 0.7 -SH 0.3 data/examples/bul.csv core.csv
```

Run the help command

```
java -jar bin/corehunter-cli.jar -help
```

for an overview of all parameters. The general usage of Core Hunter is described by

```
corehunter [options] [measures] <collection_file> <coresubset_file>
```

Three different stop criteria can be applied:

- A maximum runtime (60 seconds by default). **Note:** It is strongly advised to increase this runtime limit for datasets larger than the example `bul.csv` dataset.
- A required minimum progression in the score of the current core (by default not used)
- A maximum time without improvement (stuck time, by default also not used)

For more information about these stop criteria look in the *common advanced search options* section of the output of the help command.

The Core Analyser tool can be applied using a similar command, e.g.:

```
java -jar bin/coreanalyser-cli.jar data/examples/bul.csv
```

which will analyze the given dataset using all evaluation measures included in Core Hunter. Alternatively you may also specify multiple datasets, in this case they will all be analyzed separately and individual values will be reported, as well as minimum, maximum and mean values over the different datasets. The general usage of Core Analyser is described by

```
coreanalyser [options] <file1> [<file2> [<file3> ...]]
```

3.3 Running Core Hunter from R

We also provide a basic script `corehunter.R` to run the Core Hunter and Core Analyser tools from R[1]. To use it set your working directory in R to the Core Hunter root directory, using

[1] If you cannot find the script `corehunter.R` in your downloaded package, you can also obtain it from the website (`http://www.corehunter.org`) and copy the script to the Core Hunter root directory.

```
setwd(...)
```

Then load the script with

```
source("corehunter.R")
```

and use the commands `corehunter.run` and `coreanalyser.run` to run the Core Hunter and Core Analyser tools, respectively. Both commands have a required `options` argument which is an option string similar to those used in the command-line interface. For example you can print the Core Hunter help information using

```
corehunter.run("-help")
```

and sample a core subset from the `bul.csv` dataset using a command like

```
corehunter.run("-MR 0.7 -SH 0.3 data/examples/bul.csv core.csv")
```

which samples a core with the same parameters as the first example discussed in subsection 3.2. To analyze the `bul.csv` dataset, run the following command:

```
coreanalyser.run("data/examples/bul.csv")
```

The `coreanalyser.run` command returns a data frame containing the evaluation scores for each diversity measure (columns) and given dataset (rows). For example, when evaluating two core sets `core1.csv` and `core2.csv` the command

```
eval = coreanalyser.run("core1.csv core2.csv")
```

will return a data frame with the following format:

```
           MR   MRmin  CE   CEmin  SH   HE   NE   PN   CV
core1.csv  ...  ...    ...  ...    ...  ...  ...  ...  ...
core2.csv  ...  ...    ...  ...    ...  ...  ...  ...  ...
```

By default the R script assigns 512 megabytes of RAM memory to the Java virtual machine. However, when sampling from larger datasets this may not be sufficient and may result in a Java heap space error, indicating that the Java virtual machine ran out of memory. Therefore, if desired, the assigned amount of memory can be changed using the optional `mem` parameter which is available for both `corehunter.run` and `coreanalyser.run`. For example, the following command assigns 1 gigabyte of RAM to Core Hunter:

```
corehunter.run("-MR 0.7 -SH 0.3 data/examples/bul.csv core.csv", mem="1g")
```

The `mem` parameter should be a string consisting of an integer (e.g. 512, 1, 2, ...) followed by a unit indicator (e.g. m for megabytes, or g for gigabytes), similar to the specifications of the Java `-Xmx` and `-Xms` options used for memory management. See table 1 for some examples.

A more flexible R interface will be released in the future, please check the website for updates.

`mem`	assigned RAM memory
"256m"	256 megabytes
"512m"	512 megabytes
"1g"	1 gigabyte
"2g"	2 gigabytes

Table 1: *Some examples for the optional* `mem` *parameter of* `corehunter.run` *and* `coreanalyser.run`*. The default value is "512m".*

4 Dataset file format

The Core Hunter file format is demonstrated in the example dataset file `bul.csv` found in the `data/examples` directory. Dataset files are comma separated files (.csv) in which the first column denotes all genetic markers used in the dataset. The second column indicates the different alleles that may occur at each specific marker locus. Subsequent columns describe the accessions included in the dataset.

For each combination of accession, marker and allele the file indicates the allelic frequency observed in this accession at the respective marker locus. Usually allelic frequencies of alleles at the same locus should sum to 1. In case of missing data, fields may be left blank. The same file format is used for the output files generated by Core Hunter to store core subsets.

5 Questions

Any questions about Core Hunter should be directed to

- Herman De Beukelaer (herman.debeukelaer@gmail.com)
- Guy Davenport (daveneti@gmail.com)

Please check the website `http://www.corehunter.org` for updates. You can also subscribe to our user group at `https://groups.google.com/d/forum/corehunter-users` to receive the latest news on Core Hunter or to post questions.