The `micropan` package vignette

Lars Snipen and Kristian Hovde Liland

1 Introduction

As a result of ever-improved sequencing technologies over several years it is now possible to sequence prokaryote genomes in very little time at a small cost. This has made it possible to re-sequence many strains within the same species, to investigate the diversity, to look for a core set of genes common to all strains or to be able to distinguish pathogenic and non-pathogenic strains etc. It has since long been recognized that the diversity between prokaryote strains is much greater than between, say, individuals in a human subpopulation. All humans share the same genes, and the differences between individuals are usually found in tiny mutations like single nucleotide polymorphisms or in the regulatory parts of the genome. In contrast, two different strains of *E. coli* may be many times more diverse than humans and chimpanzees, with perhaps 20% of the genes in one strain lacking in the other. For this reason, the set of genes found in the entire *E. coli* species is many times larger than what we observe in a single genome, and this collection of all genes utilized by a species is denoted its *pan-genome*.

In this R package we provide a set of tools to analyze microbial pan-genomes. We find R suitable as a 'workbench' since it is easy to make small scripts performing analyses and visualize the results. R is also used a lot in bioinformatics, and it is quite likely you will have used R for other purposes already. However, the comparison of genomes involves heavy computations, and for this job there are other tools already optimized for this purpose. Instead of trying to re-implement some of this in R (most likely with huge effort and poor results), we have instead made use of external software where appropriate. Thus, some of the functions in this package only make a call to an external software, and to obtain full functionality you need to have these external softwares installed on your system, see below.

This vignette is not a users guide. Instead, we have prepared the document **casestudy.pdf** which will take you through some standard analyses of a small pan-genome. We reccomend you take a look at that document after reading this.

2 The genome-table

A pan-genome study may involve many genomes, and corresponding many files. We strongly reccomend that you create a table of meta-data for each pan-genome study. We will refer to this as the genome-table. This table can be created as a `data.frame` in R or in some spreadsheet. In the latter case you save it as a text file and read it into R using e.g. `read.table` (remember to set `stringsAsFactors=FALSE`). The genome-table will typically have one row for each genome in the study, and the columns contain your meta-data about each genome.

The very first column you create in this table is the column named GID.tag! A central concept in the `micropan` package is the `GID.tag`, which is a unique Genome IDentifier for every genome in the study. This is a short text of the format "GIDx", where x is some integer which is unique for each genome. This tag is attached to all sequences and files of a genome using the function `panPrep`. You can read the Help-file (`?panPrep`) or the casestudy document for more details on `panPrep`. Enter your desired `GID.tag` texts for each genome, just make certain they are all unique.

Another useful column contains a short and meaningful name for each genome (e.g. strain). The GID.tag is the default label used when plotting and displaying results in the `micropan` functions, but they are cryptic and in most cases a meaningful short name is better. Supplying the GID.tag column and the name column enable plotting functions to replace the tags by 'real' names.

Colors can be used to display *a priori* genome grouping information when plotting. Genomes that somehow 'belong together' should have identical colors. You may have several alternative color columns in your genome-table, and supplying one of them together with the GID.tag column enable plotting functions to color genomes the way you want.

Each genome should also have an associated filename. In case you have several sequence files for each genome (e.g. genome sequences, protein sequences, etc.) use the same name for all files, but store them in separate folders (e.g. the `genome` folder, `protein` folder etc.)

We like to point out that a genome-table is never required by any functions in the `micropan` package. This is just something we reccomend from our own experience.

3 External software

Pan-genomics will involve the comparison of (many) biological sequences. There are some 'workhorse' tools of bioinformatics that we also make use of here. R is an excellent workbench environment, with easy access to graphics and statistical analyses, but for heavy sequence comparisons it is slow, and these types of jobs should be exported to the prupose-built softwares used by the bioinformatics community. Thus, before you start with pan-genomics using the `micropan` package you should install on your system the following free softwares:

- The Prodigal gene finder (`https://github.com/hyattpd/Prodigal/releases`)
- The BLAST+ executables from NCBI (`ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/`)
- The HMMER3 from Janelia farm (`http://hmmer.org/download.html`)
- The barrnap software for finding rRNA-genes (`https://github.com/tseemann/barrnap`)

3.1 Prodigal

We take it for granted that many pan-genome analyses will involve draft genomes where no gene predictions have been made. The Prodigal gene finder can be used to predict genes in a genome sequence. It is one of several alternative tools for this job, but the one we have chosen as an option in our R package. As with all gene predictions, no results are perfect and we do not suggest that Prodigal is always the best tool. It is, however, always easy to use and in our experience the predictions made by Prodigal are at least as accurate as the other standard gene finders available.

In the `micropan` package Prodigal is used by the function `prodigal`.

At the time of writing Prodigal is available as a binary executable file from `http://code.google.com/p/prodigal/downloads/list`. There is no installation program, but you simply download the executable and put it where you like on your system.

For `prodigal` to work, you must change the name of this binary to `prodigal` (or `prodigal.exe` in Windows), no more and no less (delete all version numbers etc.). Also, you have to manually edit the `PATH` environment variable on your system, and add to this the path to where you have the `prodigal` binary. As an example, on my Ubuntu linux system the `prodigal` binary resides in `/home/larssn/bin/Prodigal/`, and the command

```
export PATH="/home/larssn/bin/Prodigal":$PATH
```

updates the `PATH` variable. Add this to the `.bashrc` file or something similar to make it permanent. On my Windows systems the `PATH` variable is edited in the Control Panel.

If you are using the RStudio IDE software you may find it necessary to also create an `.Renviron` file on your system, and add the `PATH` information in this. On my Ubuntu linux system it looks something like this:

```
PATH = "/home/larssn/bin/Prodigal:/home/larssn/bin/HMMER3:
/home/larssn/bin/BLAST:/home/larssn/bin/barrnap"
```

Notice that each path is separated by a colon, and you may add as many paths as you like. On my Windows system I do not need the `.Renviron` file at all, RStudio works fine without!

To verify that you have `prodigal` properly installed, start R and run the line

```
> system("prodigal -h")
```

in the command window. You should get a listing of the available options in `prodigal`.

3.2 BLAST

The Basic Local Alignment Search Tool is known to most people by its web-interface where you can search huge databases for sequences similar to your query sequence. It is, however, also a stand-alone tool you can install on your local computer. At the time of writing the BLAST+ executables can be downloaded from `ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/`.

Executables are available for all platforms, and installing it is quite straightforward on all systems. You may need to manually edit your environment variable `PATH` and your `.Renviron` file as mentioned above. A successful installation means you can write in the R command window

```
> system("blastp -help")
```

and you should get a listing of how to use the command `blastp`. In the `micropan` package BLAST is used by the function `blastAllAll`.

3.3 HMMER3 and Pfam

The HMMER3 software is used to build profile Hidden Markov Models (pHMM) and to search with sequences against a database of such models. A pHMM is a sequence model describing a position specific pattern. It is typically used to describe conserved parts of proteins. The Pfam database is the best known collection of pHMMs, and HMMER3 is the tool associated with Pfam.

HMMER3 can be downloaded from `http://hmmer.org/download.html`. Again, if you are using RStudio you may need to add the path to HMMER3 to the `.Renviron` file. Verify that HMMER3 works properly by running

```
> system("hmmscan -h")
```

in the R command window. You should get a listing of available options for `hmmscan`. In the `micropan` package HMMER3 is used by the function `hmmerScan`.

In order to use the `hmmerScan` function you also need to have a pHMM database on your system. The Pfam-A database is the natural choice. At the time of writing this can be downloaded from `ftp://ftp.ebi.ac.uk/pub/databases/Pfam/current_release`. You will typically need the file named `Pfam-A.hmm.gz`. Uncompress this file, and then run `hmmpress` on it (see the HMMER3 user manual). NB! This is done only once, and should *not* be done from the R command window, but directly in a system command window.