QCF Documentation

Release 0.3.0a

Paul Blischak

Feb 06, 2020

Contents

`qcf` is a C++ program for estimating quartet concordance factors from multilocus sequence data. It uses the LogDet transformation to calculate the frequency of gene tree quartets matching the three possible unrooted species tree topologies for a set of four taxa. `qcf` can handle multiple haplotypes per taxon/population, and outputs a CSV file that can be used to infer a species tree using scripts from the TICR pipeline and is compatible with the SNaQ method for phylogenetic network inference implemented in the Julia package PhyloNetworks.

CHAPTER 1

Documentation

1.1 Getting Started

1.1.1 Installation

`qcf` can be installed by cloning the code from GitHub using the following steps:

```
git clone https://github.com/pblischak/QCF.git        # 1. Clone the repo from GitHub
cd QCF/                                               # 2. cd into the QCF/ folder
make                                                  # 3. compile the qcf executable
make test                                             # 4. test that the executable␣
↪works
sudo make install                                     # 5. copy executable to /usr/
↪local/bin
```

The `sudo make install` step will also cp all the files in the `scripts/` folder to `/usr/local/bin`. Stable versions of QCF are also available on the Releases page.

1.1.2 Input Files

Phylip Files

Sequence data for each gene should be in its own file in Phylip format. The setup should be the same as if you were planning to run RAxML on each gene individually.

Example:

```
  16  500
sp1_1 AGTACAAGGTAGACAGTAGACG...
sp1_2 AGTACAAGGTAGACAGTAGACG...
sp2_1 AGTACAAGGTAGACAGTAGACG...
.
```

(continues on next page)

(continued from previous page)

```
.
.
spN_3 AGTACAAGGTAGACAGTAGACG...
```

Gene List File

The gene list file is a simple text file that has the name of each Phylip file that is to be included in an analysis on its own line.

Example:

```
gene1.phy
gene2.phy
gene3.phy
.
.
.
geneL.phy
```

If this file and the gene sequence files are not in the same directory, then you can add the relevant path information to the Phylip files here so that the program can still find them (e.g., `path/to/geneL.phy`).

Map File

The mapping file maps haplotypes to sampled taxa. The easiest way to do this is to sequentially number the haplotypes for each gene (e.g., SpeciesName_1, SpeciesName_2, etc.). Genes are treated as independent, so they can reuse the same haplotype names. Also, not all genes need to have all haplotypes. For each taxon, start with its name, followed by a colon (`:`), then the names of the haplotypes that are present in the Phylip files containing the sequence data, each separated by a comma (`,`). **There should be no spaces**. This format is the same as the one used by ASTRAL.

Example:

```
sp1:sp1_1,sp1_2,sp1_3
sp2:sp2_1,sp2_2
.
.
.
spN:spN_1,spN_2,spN_3,spN_4
```

1.1.3 Output Files

`qcf` by default will produce an output file that contains the estimated quartet concordance factors in a file called `out-qcf.CFs.csv`. If you also print the raw quartet scores, then the program will write another file called `out-raw.csv`. This file contains all of the raw scores for all haplotypes for each species quartet (it is not intended to be human readable). The `out-raw.csv` file is what can be passed to the `qcf_boot.py` Python script to conduct bootstrap resampling for confidence interval estimation.

1.2 Tutorial

Below we will go through an example analysis using some simulated data that is available in the `example/` folder in the QCF GitHub repository.

1.2.1 Step 0: Get `qcf`

If you don't have the repo, clone it and install qcf. If you already have the QCF repository then go ahead and skip to the next step.

```
git clone https://github.com/pblischak/QCF.git
cd QCF
make
make test
sudo make install
```

1.2.2 Step 1: Look at Example Files

Now we'll run the example analyses. First, we'll change into the *example/* folder in the QCF repo.

```
# cd into the example folder (wherever it is on your computer)
cd /path/to/QCF/example

# check out what files are there
ls
```

You should see the Phylip files containing the sequence data, the `genes.txt` file containing a list of all of the genes, and the `map.txt` file.

```
# Aside: Making the gene list file can be done in the folder with all of the
# Phylip files using the following code within a terminal
ls -1 *.phy > genes.txt
```

1.2.3 Step 2: Run `qcf`

The most basic analysis that we can do with `qcf` is to just calculate QCF scores with the `genes.txt` and `map.txt` files.

```
qcf -i genes.txt -m map.txt --prefix example1
```

If you want to incorporate uncertainty into the estimation of the QCF scores, there is an option to perform bootstrap resampling of sites within each gene. This will make the analysis slower depending on how many bootstrap replicates you perform.

```
# add -b <#> for bootstrapping
qcf -i genes.txt -m map.txt -b 500 --prefix example2
```

To calculate confidence intervals for the QCF scores, we'll add the `--printRaw` flag when calling `qcf`. This will generate an extra output file that can be used with the `qcf_boot.py` script to

```
# add the --printRaw flag
qcf -i genes.txt -m map.txt -b 500 --prefix example3 --printRaw
```

Using the raw output from the previous step, we'll use the Python script `qcf_boot.py` to resample gene-level quartets to calculate QCF values and their 95% confidence intervals.

```
qcf_boot.py -i example3-raw.csv -b 500 --prefix resampled
```

This will generate the file `resampled-boot.CFs.csv`. This file can be used to infer a species tree with scripts from the TICR pipeline (Stenz et al. 2015), which are packaged with QCF in the `scripts/` folder (should be available if you ran `sudo make install`).

Note: Citing TICR

If you use these scripts please be sure to cite the TICR pipeline:

Stenz, N. W. M., B. Larget, D. A. Baum, and C. Ane. 2015. Exploring Tree-Like and Non-Tree-Like Patterns Using Genome Sequences: An Example Using the Inbreeding Plant Species *Arabidopsis thaliana* (L.) Heynh. *Systematic Biology* 64:809–823.

To use these scripts, you will also need to install QuartetMaxCut, which is available here. The TICR README has a lot of helpful information for using these scripts as well.

```
# Get a tree topology using QuartetMaxCut
# Usage:
#   perl get-pop-tree.pl <bootstrap QCF file>
perl get-pop-tree.pl resampled-boot.CFs.csv

# Estimate branch lengths in coalescent units
# Usage:
#   Rscript --vanilla getTreeBranchLengths.R <bootstrap file prefix> <outgroup>
Rscript --vanilla getTreeBranchLengths.R resampled-boot 6
```

The `resampled-boot.CFs.csv` file is also formatted to be analyzed using the SNaQ species network inference method in the PhyloNetworks package. Documentation for running SNaQ is available on the PhyloNetworks website.

Analyzing Genes in Parallel

If you have a large number of genes, it is possible to analyze smaller numbers of genes separately and in parallel to make analyses more computationally efficient. To do this, instead of listing all genes in one file, create several files listing different groups of genes, analyze each one on its own (you can use the same map file for each), and then combine them using the `qcf_boot.py` script. Because the results are combined using `qcf_boot.py`, each analysis will have to be run with the `--printRaw` flag.

```
# First we'll analyze gene set 1
qcf -i genes1.txt -m map.txt -b 500 --prefix out1 --printRaw

# Now gene set 2
qcf -i genes2.txt -m map.txt -b 500 --prefix out2 --printRaw
```

Now we'll calculate QCFs and their confidence intervals across the independent runs we just completed. The qcf_boot.py script is written such that it can combine the raw data across any number of independent runs.

```
#
qcf_boot.py -i out1-raw.csv out2-raw.csv -b 500 --prefix resampled2
```

If you have more than 2 input files, you can list them all after the `-i` flag:

```
qcf_boot.py -i out1-raw.csv out2-raw.csv out3-raw.csv <...more files...> \
            -b 500 --prefix resampled3
```

An easy way to list them all would be to do something like this:

```
qcf_boot.py -i $(ls *-raw.csv) -b 500 --prefix resampled4
```