qfaBayes - An R package for Bayesian Quantitative Fitness Analysis

Jonathan Heydari

April 11, 2014

1 Introduction

Quantitative Fitness Analysis (QFA) is an experimental and computational workflow for comparing fitnesses of microbial cultures grown in parallel. QFA can be applied to focused observations of single cultures but is most useful for genome-wide genetic interaction or drug screens investigating up to thousands of independent cultures. The central experimental method is the inoculation of independent, dilute liquid microbial cultures onto solid agar plates which are incubated and regularly photographed. Photographs from each time-point are analyzed, producing quantitative cell density estimates, which are used to construct growth curves, allowing quantitative fitness measures to be derived. Culture fitnesses can be compared to quantify and rank genetic interaction strengths or drug sensitivities. The effect on culture fitness of any treatments added into substrate agar (e.g. small molecules, antibiotics or nutrients) or applied to plates externally (e.g. UV irradiation, temperature) can be quantified by QFA.
Detailed descriptions of how to carry out QFA experiments are available in open access articles, particularly in Banks et al. (2012) and Addinall et al. (2011). The purpose of this document is to describe, in detail, some of the computational methods available in the qfaBayes R package for summarising experimentally observed growth curves during QFA, and to demonstrate the computational component of QFA using some small, example datasets.

2 QFA data

The raw experimental data generated by QFA consists of timeseries photographs of cultures growing on agar plates. The first step in the computational component of the QFA workflow is to convert these photographic observations into cell density estimates for cultures in each position on each plate analysed. The Colonyzer image analysis tool (Lawless et al. (2010)) is designed for this task and can be downloaded from its website. Once all the images have been successfully analysed, the next step is to use the qfaBayes R package to associate culture locations with genotypes and to construct growth curves (cell density estimates over time) for each culture.

3 Installing the qfaBayes package

The qfaBayes package source code is available for download from R-Forge, and so it should be possible to install the latest version using the R package management system on a wide range of operating systems by executing the following command within an R environment:

```
install.packages("qfaBayes",repos="http://r-forge.r-project.org")
```

Once installed, the package can be loaded ready for use with

```
library(qfaBayes)
```

Please note that this installation method will typically only work using the latest version of R (which can be freely downloaded from the R website). Alternatively, instructions for accessing the source code for the package from are available here.

4 Function documentation

The following command will provide an overview of functions available within the qfaBayes package together with brief descriptions of what they do and links to detailed descriptions indicating input arguments and output:

```
help(package="qfaBayes")
```

This document can be accessed at any time with:

```
vignette("qfaBayes")
```

5 General overview

This R package is intended to fit the three Bayesian Models described in J Heydari, C Lawless, D Lydall and D J Wilkinson. *Bayesian hierarchical modelling for inferring genetic interactions in yeast. Journal of the Royal Statistical Society: Series C (Applied Statistics)*, in submission. The three Bayesian models are the SHM (Single hierarchical model), IHM (Interacation hierarchical model), and JHM (Joint hierarchical model). There are six demos included in this packge, three of which (`SHM_simple_C`, `IHM_simple_C` and `JHM_simple_c`)have been made for ease of use, hiding all post-processing. The other three (`SHM_C`, `IHM_C` and `JHM_C`) consist of many lines of code that can be more easily edited to create a much more tailored post-processing. Variables `burn`, `iter` and `thin` control the burn-in period, output sample size and thinning. These are all set to 1 in all the six of the demo's so that a user can get familiar with the workflow first, it is essential that you change them to much larger numbers for sufficient convergence to happen. Typically the models need a computational time that ranges from range from days to weeks.

5.1 SHM

`SHM_simple_C` This demo runs the SHM for the *ura3*Δ data set at 27 °C trimmed to only include plate 15, which accounts for only 50 of the 4294 *orf*Δs avalible.

It is expected that you will wish to create ouput with larger sample size, burn in period and thinning, to do this simply copy all the code from the demo and change the variables `burn`, `iter` and `thin`.
The typical model computation time of the SHM alone is one week with burn-in of 800000, sample size of 10000 and thinning of 10.
A more detailed workflow which is compatible with the standalone C code is given in demo `SHM_C`.

5.2 IHM

`IHM_simple_C` This demo runs the SHM for the *ura3*Δ data set at 27 °C, trimmed to only include plate 15, which accounts for only 50 of the 4294 *orf*Δs available. Next, the SHM is then run for the *cdc13-1*Δ data set at 27 °C, trimmed to only include plate 15, which accounts for only 50 of the 4294 *orf*Δs available. The IHM is then run using the output from the SHM output for the two data sets. Finally you will be asked if you wish to create a fitness plot with your results.
It is expected that you will wish to create ouput with larger sample size, burn in period and thinning, to do this simply copy all the code from the demo and change the variables `burn`, `iter` and `thin` where they appear. The typical model computation time of the IHM alone is one day with burn-in of 800000, sample size of 10000 and thinning of 10.
A more detailed workflow which is compatible with the standalone C code is given in demo `IHM_C`.

5.3 JHM

`JHM_simple_C` This demo runs the JHM for the *ura3*Δ and *cdc13-1*Δ data sets, both at 27 °C, trimmed to only include plate 15, which accounts for only 50 of the 4294 *orf*Δs available. At the end of the demo you will be asked if you wish to create a fitness plot with your results.
It is expected that you will wish to create ouput with larger sample size, burn in period and thinning, to do this simply copy all the code from the demo and change the variables `burn`, `iter` and `thin` where they appear. The typical model computation time of the JHM alone is two weeks with burn-in of 800000, sample size of 5000 and thinning of 10.
A more detailed workflow which is compatible with the standalone C code is given in demo `JHM_C`.

6 SHM_simple_C Demo

To see how the `SHM_simple_C` demo works the following detailed explaination is provided and is to be used in conjunction with the R package manual. The `SHM_simple_C` demo runs the following commands:

```
> data("URA3_Raw_extratrim_15")
> a<-a_15
```

The above commands load a trimmed ura3Del data set consisting of only of

master plate 15 to the variable `a_15` and copys this to a variable "a".

```
> data("priors_SHM")
> PRIORS=as.double((priors_SHM)[[1]])[1:18]
```

The above commands loads a broad set of prior values for the SHM to the variable `priors_SHM` and then copys this to variable PRIORS.

```
> qfa.variables(a)
```

The above command lists the avalible options for the treatment, screen number and master plate.

```
> Screen<-unique(a$Screen.Name)
```

The above command selects all screens for the following SHM analysis.

```
> SHM<-SHM_postpro(a=a,Treat=27,Screen=Screen,MPlate=15)
```

The above command runs the post processing for the SHM, organising the Colonyser ouput in variable `a` for use with the SHM. The dataset will be trimmed to only include the data corresponding to the specified `Treat`, `Screen` and `MPlate`.

```
> SHM_output<-SHM_main(burn=1,iters=1,thin=1,CAPL=50,
QFA.I=SHM$QFA.I,QFA.y=SHM$QFA.y,QFA.x=SHM$QFA.x,
QFA.NoORF=SHM$QFA.NoORF,QFA.NoTIME=SHM$QFA.NoTIME,PRIORS=PRIORS)
```

The above command runs the C code for the SHM in R. `burn`, `iters` and `thin` which have all been set to 1 for the purpose of the demo. `CAPL` is the number of orfs for the SHM to proform the analysis on. The output `SHM_output` will be a table of posterior samples, where each coloumn corresponds to a different model paramater; a table header is included to idendify each coloumn.

```
> ask_plot_simple()
```

The above command will ask the user if they would like an example of some plots created form the posterior sample.

```
> plot_SHM_simple(SHM_output,SHM)
```

The above commands outputs fitted logistic growth curve plots for each ORF, where black is for the repeat fitted curves and red for orf level fitted curves.