qfa - An R package for Quantitative Fitness Analysis

Conor Lawless

December 10, 2014

1 Introduction

Quantitative Fitness Analysis (QFA) is an experimental and computational workflow for comparing fitnesses of microbial cultures grown in parallel on solid agar surfaces. 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 qfa 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 qfa R package to associate culture locations with genotypes and to construct growth curves (cell density estimates over time) for each culture.

3 Installing the qfa package

The qfa 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("qfa",repos="http://r-forge.r-project.org")
```

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

```
library(qfa)
```

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 qfa package together with brief descriptions of what they do and links to detailed descriptions indicating input arguments and output:

```
help(package="qfa")
```

This document can be accessed at any time with:

```
vignette("qfa")
```

Documentation for specific functions can be obtained using the usual R mechanisms. For example, help on the function `colonyzer.read` can be obtained with:

```
?colonyzer.read
```

There is a short demo script comparing QFA of *cdc13-1* and *ura3*Δ at 27 °C to infer genes interacting with the telomere cap. Note that the curve-fitting functions can throw up many error messages which can be ignored. These occur when attempting to fit the model to missing cultures. This demo only contains data from the yeast deletion collection describing one plate out of a possible 15. This demo also only uses one replicate plate out of a possible eight for each screen in order to save analysis time. To repeat the analysis with all available replicates, uncomment the appropriate lines in the demo script. Note that the full analysis takes approximately 30 mins. The demo can be loaded with:

```
demo("telomereCap")
```

To help you build your own QFA script, you can use the contents of the demo script as a starting point. To see the contents of the demo script, use the following function from the qfa package:

```
showDemo("telomereCap")
```

5 General overview

This R package consists of a wide range of functions, which can be grouped according to their purpose.

5.1 Reading and formatting data

`colonyzer.read` This function reads in image analysis output from Colonyzer, together with files containing experimental metadata and it associates cell density estimates with culture type (e.g. genotype) and treatment (e.g. temperature, drug concentration) for each culture. All data are bound together into a data.frame object, with rows representing unique observations of individual cultures.

5.2 Summarizing observed growth curves

5.2.1 Generalised Logistic Growth Model: the `Glogist` function

The original QFA analysis presented in Addinall et al. (2011) involved fitting the logistic model to experimentally observed growth curves. During subsequent screens using automated incubators and automated imagers where we capture images much more frequently and cultures grow in environments with different degrees of humidity control, we have found that in many cases observed growth curves do not fit the logistic model perfectly. In particular we have found that the symmetry inherent in the logistic model is not always observed. In this package we use the more flexible generalised logistic differential equation, which has an additional shape parameter ν to account for asymmetrical growth curves. Importantly, the original logistic model can be recovered from the generalised model by setting $\nu = 1$:

$$\frac{dg}{dt} = rg\left(1 - \left(\frac{g}{K}\right)^{\nu}\right) \tag{1}$$

Which has an analytical solution:

$$g(t, g_0, r, K, \nu) = \frac{K}{\left(1 + \left(-1 + \left(\frac{K}{g_0}\right)^{v}\right) e^{-rvt}\right)^{\frac{1}{\nu}}} \tag{2}$$

K Culture carrying capacity (AU). Same units as (normalised) cell density observed in growth curve.

r Culture growth rate parameter (per day).

g_0 Inoculum denisty (AU). Same units as (normalised) cell density observed in growth curve.

ν Shape parameter. Recover logistic model with $\nu = 1$.

t Time since inoculation (d).

5.2.2 Smoothing or interpolation: the `loapproxfun` function

This package also provides a model-free alternative for summarising experimentally observed growth curves. The `loapproxfun` function is a function closure. Given a timeseries dataset (growth curve data) it returns an appropriate approximating function. If a loess smoothing span parameter appropriate for the data capture frequency (frequency of photographs) is specified, the approximating function will be a smoothed version of the data in the range of observations. For all points before the first observation, the approximating function takes the value of the first smoothed version of the data. Simiarly, beyond the final observation, the function returns the smoothed version of the data at the final timepoint. If an inappropriate span parameter is passed to this function it will return a linear interpolation approximating function instead. This can be more robust where the loess smoother would add spurious curves to datasets with sparse observations (e.g. data captured manually 2 or 3 times per day).

5.2.3 Summarizing growth curves: the `qfa.fit` function

This function fits the generalised logistic model (represented by the `Glogist` function) to sets of observed timecourses by default (though this can be disabled if model-based fitnesses are not of interest), returning the model parameter values (listed above) which best fit the data. These parameters can be used to construct model-based fitnesses later (see below). At the same time `qfa.fit` also summarizes data with the `loapproxfun` function. Storing the coefficients from the `loapproxfun` function is impractical and so these are used directly to generate two model-free fitness measures: a Single Time Point (nSTP) fitness surrogate and Area Under the growth Curve (nAUC). The "n" indicates "numerical" or model-free, and is used to differentiate from equivalent measures of fitness which can be derived from the genralised logistic growth curves. nSTP returns the cell density estimate for a time after inoculation specified by the `STP` argument to `qfa.fit` (the default is a value well after culture growth is complete). nAUC is the integral under the smoothed (or interpolated) curve from inoculation until a time `AUCLim` after inoculation. Again, `AUCLim` is an argument to `qfa.fit` and its default value is set to 5 days.

5.2.4 Generating fitnesses from the generalised logistic model: the `makeFitness` function

This function generates several fitnesses based on the generalised logistic model parameters estimated using the `qfa.fit` function:

MDR Maximum Doubling Rate is the generalised logistic version of the MDR fitness measure presented by Addinall et al. (2011).

$$MDR = \frac{r\nu}{log\left(1 - \frac{2^{\nu}-1}{(2^{\nu})\left(\frac{g_0}{K}\right)^{\nu}-1}\right)} \tag{3}$$

MDP Maximum Doubling Potential is the generalised logistic version of the MDP fitness measure presented by Addinall et al. (2011).

$$MDP = \frac{log(\frac{K}{g_0})}{log(2)} \tag{4}$$

$MDR \times MDP$ The generalised logistic version of the $MDR \times MDP$ fitness measure presented by Addinall et al. (2011).

AUC Area Under the Growth curve described by the generalised logistic model. This is essentially the integral under the modelled growth curve (above the inoculum density) from inoculation to an arbitary later timepoint.

$$AUC = \int_0^{AUCLim} g(t, g_0, r, K, \nu), \mathrm{d}x - g_0 AUCLim \tag{5}$$

DT Time required for culture to reach twice its current cell density. Setting t=t0 recovers maximum doubling time (i.e. doubling time at inoculation, assuming no lag effect).

$$DT = \frac{\log\left(\frac{2^\nu\, e^{\nu\, r\, t}\left(\frac{K}{G0}\right)^\nu - 2^\nu\, e^{\nu\, r\, t}}{\left(\left(\left(\frac{K}{G0}\right)^\nu + e^{\nu\, r\, t} - 1\right)^{\frac{1}{\nu}}\right)^\nu - 2^\nu\, e^{\nu\, r\, t}}\right) - \nu\, r\, t}{\nu\, r} \tag{6}$$

5.3 Inferring genetic interaction strengths

Addinall et al. (2011) present methods for statistical epistasis analysis based on linear error models. For appropriately designed experiments, these functions can be used to compare sets of query mutation observations with expected observations, given observations of control mutation fitnesses and the expected effect of the query mutation, given genome wide observations. Effectively, we use genome-wide observations to construct a linear predictor of query mutation fitness given control mutation fitness, and test for the significance of deviations from this prediction. Mutation fitnesses come from multiple, replicate observations which can be summarised by mean or median fitness and significance of deviations can correspondingly be estimated by Student's t-test or the Mann-Whitney test after correction for multiple comparisons. Analysis based on mean/t-test is preferred to that using median/Mann-Whitney test, since the latter have greater statistical power, however, in the case where it has not been possible to perform adequate quality control on the source data (e.g. there are occasionaly contaminants, or missing cultures, resulting in statistical outliers) the former may be preferable.
Genetic interaction strengths and the statistical significance of observed strengths can be generated using the `qfa.epi` function, once logistic model fits have been carried out, as above.

5.4 Auxiliary functions

Together with the functions for carrying out the raw analysis above, we provide several functions for visualising the data, the fit of the logistic model to the data and the visualisation of evidence for epistatic interaction. These visualisation tools are important for tracking bugs and increasing user confidence in the validity of the sophisticated QFA workflows.
QFA experiments are often used to compare the fitnesses of independent microbial strains under two different environmental conditions (e.g. query and control conditions), or in two different genetic backgrounds, to search for evidence for

drug interactions or genetic interactions for example. Such comparisons become difficult to visualise by static scatterplot for genome-wide QFA due to the sheer number of strains examined, simply because simultaneous, legible labelling of 4,000 genes on a single plot is not practical.

The `qfa.epiplot` function generates static, vector graphics scatterplots for comparing fitness summaries derived from a pair of QFA experiments. Plots produced by the `qfa.epiplot` function static plots were used to demonstrate genetic interactions in Addinall et al. (2011). Figure 2 from that paper is an example, where text labels for several hundred interesting genes obscure each other.

A dynamic, interactive version of these plots, which labels and highlights user-specified genes, instead of attempting to label all points at once, can be generated (under Windows and OSX) using the `visTool` function. This is likely preferable to the static equivalent for most users. The datasets from Addinall et al. (2011) are included in a demo version of this function: `visToolDemo()`. Interacting with this dynamic plotting tool is more fully documented here.