An introduction to GAGA

Alex Murison and Christopher Wardell, alex.murison@icr.ac.uk

January 12, 2014

This vignette serves as an introduction for the R package GAGA. It covers the basic usage of the package and contains several worked examples. If you use this package, please cite: (CITATION).

Installation: The latest stable version can be installed from Bioconductor here...

The latest development version can be installed from our GitHub here: https://github.com/MurisonWardell.

```
> ## Install
> source("http://bioconductor.org/biocLite.R")
> biocLite("GAGA")
> ## Load
> library(GAGA)
```

Contents

1 Overview

1.1 Introduction

Why we wrote gaga, the type of input data (SNVs - and reference our papers) and the basic outputs (phylogenies, proportions and heatmap)

1.2 Genetic algorithms

Overview of genetic algorithms and the string encoding each individual.

1.3 GAGA input

1.4 GAGA output

Discussion that you might get a different answer every time and that the number of clones is probably the most important variable. The user MUST cycle through a number of clones and choose the lowest number of clones with the best answer. INCLUDE SAMPLE CODE!

2 Worked examples

A number of sample data sets are distributed with the GAGA package and are discussed in order of increasing complexity.

2.1 Example 1 - simple synthetic data

A very small and simple synthetic data set is included. To demonstrate that your GAGA installation is working, you can execute the following commands.

```
> ## Load library
> library(GAGA)
> ## Load simple data set
> data("gaga_simple_data")
> ## There  are three time points (T0, T1 and T2)
> ## and four mutations (M1, M2, M3, M4)
> gaga_simple_data

  names T0  T1  T2
1    M1  1 1.0 1.0
2    M2  0 0.5 1.0
3    M3  0 0.0 0.3
4    M4  0 0.0 0.5

> ## Execute gaga() function on the simple data set
> simpleDataSolution=gaga(gaga_simple_data, number_of_clones=4, nroot=1,iterations=3000)
```

The data represents

```
> ## Execute gaga() function on the simple data set
>
> #Top, zero-scoring solutions:
> #     x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20
> #[1,]  0  1  2  2 10  0  0  0  4   4   0   0   0   2   3   5   1   2   3   4
> #[2,]  0  1  2  2  8  0  0  0  4   4   0   0   0   2   3   5   1   2   3   4
> #[3,]  0  1  2  2 12  0  0  0  4   4   0   0   0   2   3   5   1   2   3   4
> #[4,]  0  1  2  2  9  0  0  0  4   4   0   0   0   2   3   5   1   2   3   4
> #[5,]  0  1  2  2 11  0  0  0  4   4   0   0   0   2   3   5   1   2   3   4
> #[6,]  0  1  2  2  9  0  0  0  3   3   0   0   0   2   3   5   1   2   3   4
```

2.2 Example 2

1.) Synthetic data as a proof of principle. Note the inclusion of the jittered data (explain the jitter) and show that it works]

2.3 Example 3

2.) The yeast data, as it's REAL data and has a definite answer

2.4 Example 4

3.) Perhaps include the data from the recent LM paper?