Passenger Mutation Simulation

Thomas O. McDonald and Franziska Michor

September 7, 2016

Example simulation involving passenger mutations only

A simple simulation for branching process with passenger mutations only. Each individual may split or die. Upon splitting, a new mutant is born with the same rate and mutation probabilities as the parent. The process can be run with global paramters for all ancestor clones or by defining ancestors separately. We show an example of both.

C++ Simulation set up

The simulation can be run with only an input file if all ancestors are treated exactly the same. The included file `inputdata.txt` is loaded with the command line flag `-in`. The contents of the file are given below.

```
cat examples/constant-rate/passenger/inputdata.txt
```

```
# SIMULATION INPUTS - USE "#" TO COMMENT OUT LINES
#
# SIMULATION PARAMETERS
tot_life, 10000
max_pop, 100000
ancestors, 10
ancestor_clones, 1
num_sims, 10
# observation_times, 1 10 100 500 1000
observation_frequency, 1
allow_extinction, 0
detection_threshold, 0.0
trace_ancestry, 1
count_alleles, 1
# RATE PARAMETERS
birth_rate, 1.0
death_rate, 0.5
mutation_prob, 0.01
```

The process begins with a single clone of 10 ancestors and runs until 100,000 individuals are alive or the time reaches 10,000. The simulation is performed 10 times and the number of individuals is output to `timedata.txt` at the given observation times (NOTE: this overides the observation frequency parameter). Extinction is not allowed, so simulations that go extinct are outputted to `timedat.txt`, but an extra one is ran. The birth and death rates are 1.0 and 0.5 respectively, and the units are the inverse of the time units. The probability of a mutation at any split is 0.01.

Running the simulation in bash works as follows. Three files will be made in your directory, labeled `timedata.txt`, `clonedata.txt`, and `sim_stats.txt`.

```
./SIApop -in ./simulation/directory/inputfile.txt -out ./simulation/directory/
```

Each file displays different information.

- `sim_stats.txt` contains information about the simulation. Many of the data are copied from `inputdata.txt` so that a new simulation can be run as an extension of the current one.

- `clondata.txt` gives complete information about each clone and is in the same format as an ancestor file so a simulation can be resumed.
- `timedata.txt` contains time course data for each clone to plot trajectories.

Using the Ancestor Flag

If we wish to distinguish between ancestor clones, we can include a file containing information about individual ancestor clones to import into the simulation. Our ancestor file contains information about 5 clones in tab-delimeted format.

```
cat examples/constant-rate/passenger/ancestors.txt
```

```
unique_id   numcells    birthrate   deathrate   mutprob
a1  10  1.50    1.0 0.01
a2  11  1.80    1.0 0.01
a3  120 1.30    1.0 0.02
a4  120 1.10    1.5 0.02
a5  13  0.30    0.2 0.01
```

Running the process in bash is the same as before, but with the added flag `-anc`.

```
./SIApop -in ./simulation/directory/inputfile.txt -out ./simulation/directory/
  -anc ./simulation/directory/ancestors.txt
```

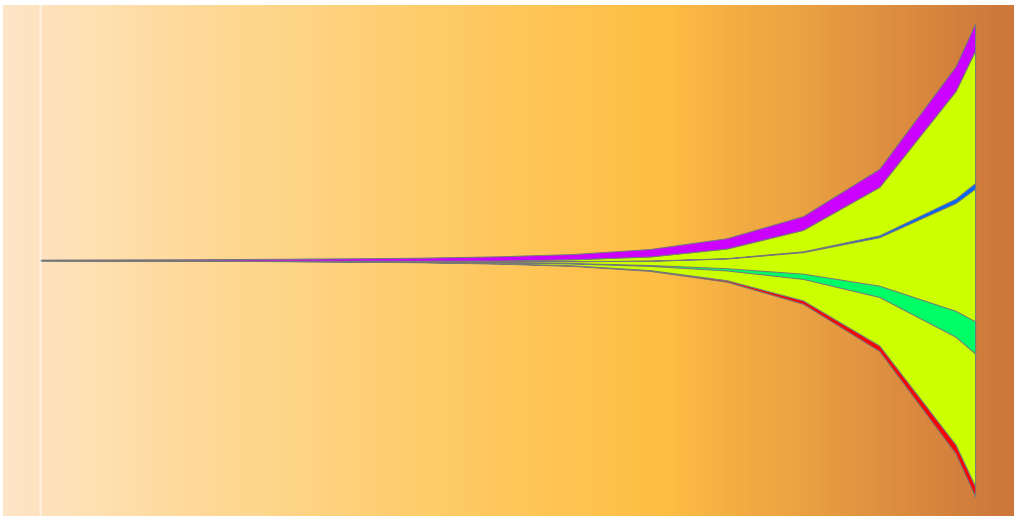
Importing into R and plotting

Once the simulation is complete, the data can be imported into R for analysis and plotting. We include a few simple functions to plot the data in different formats. The imported data contains 10 runs of the same simulation with the ancestor file included as above. Our plotting is first limited to a single run.

```
source('R/fishplot.R')
source('R/cloneplot.R')
library(readr)
library(tidyr)
library(dplyr)
library(ggplot2)
library(fishplot)
timedata <- read_delim('examples/constant-rate/passenger/timedata.txt', delim = "\t")
```

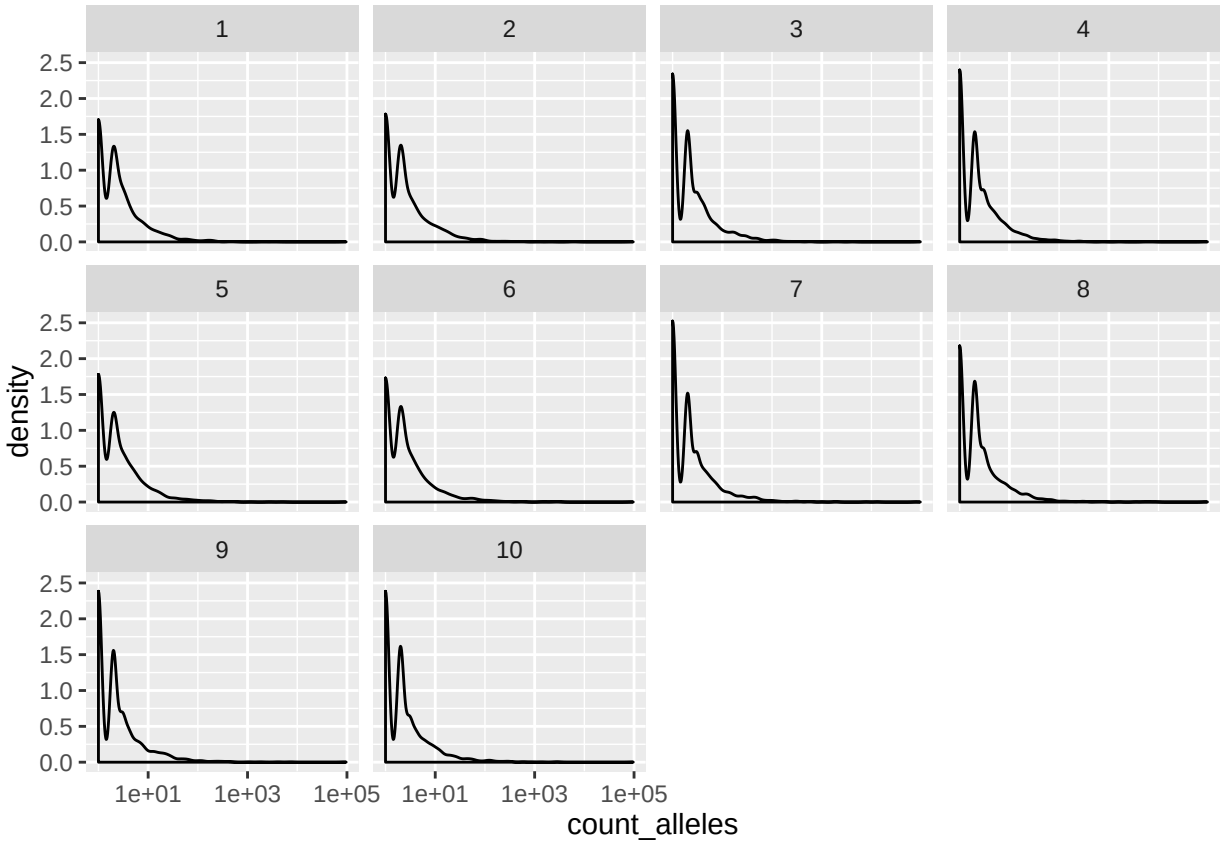
The fish plot from Chris A Miller shows the evolution and growth of all clones. We include an argument in the function for a minimum clone frequency required to view. Other options for the fishplot can be found with '?fishPlot'.

```
timedata %>% filter(run == 2) %>% fishPlotBD(0.01)
```



We can look at a histogram of allele frequencies with `clonedata.txt`.

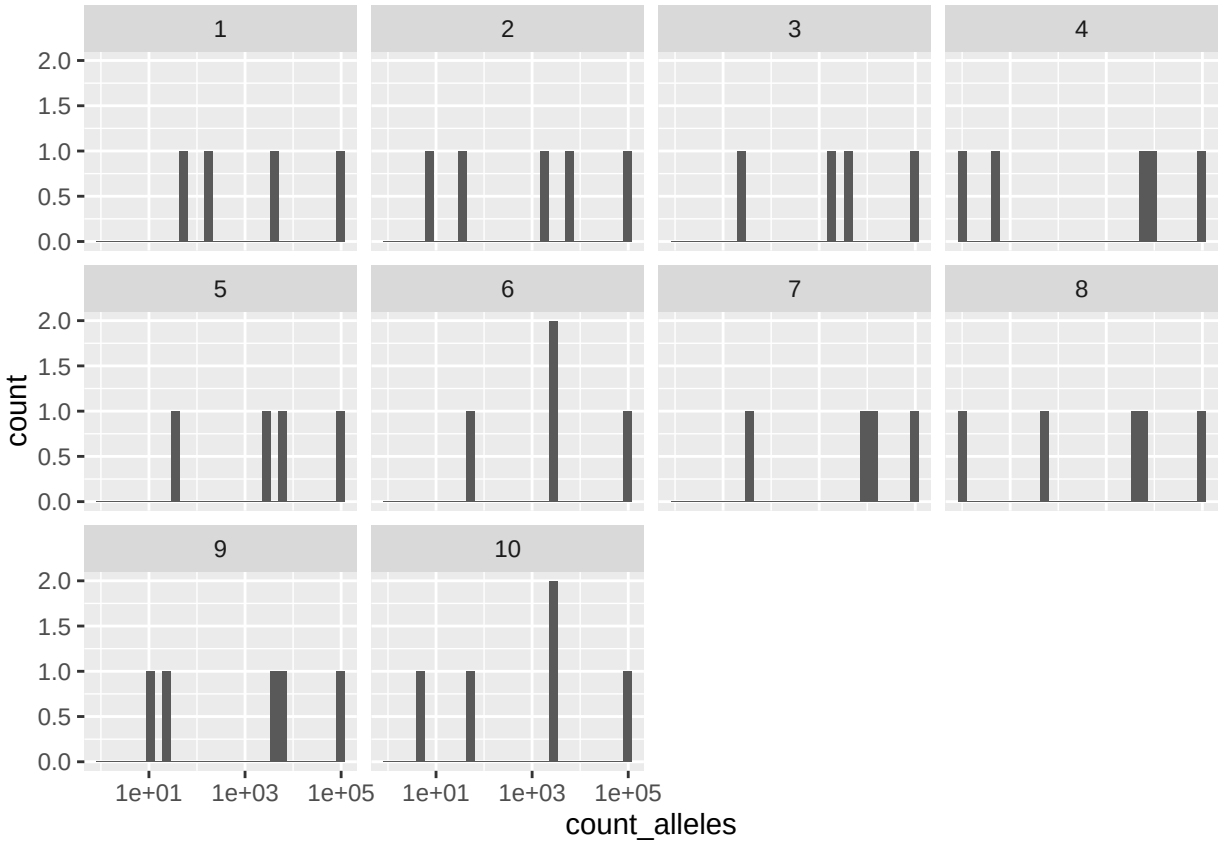
```
clonedata <- read_delim('examples/constant-rate/passenger/clonedata.txt', delim = "\t")
clonedata %>%
  ggplot(aes(x = count_alleles)) + geom_density() + scale_x_log10() +
  facet_wrap(~run)
```



We can treat the ancestors like barcoding data where each ancestor clone corresponds to a cell/cell population with a unique barcode that can be sequenced at the end of an experiment. Since these studies only sequence barcodes and not all mutations, the data should be modified so that only the ancestor types are present. This isn't very useful in this example due to the small number of ancestors.

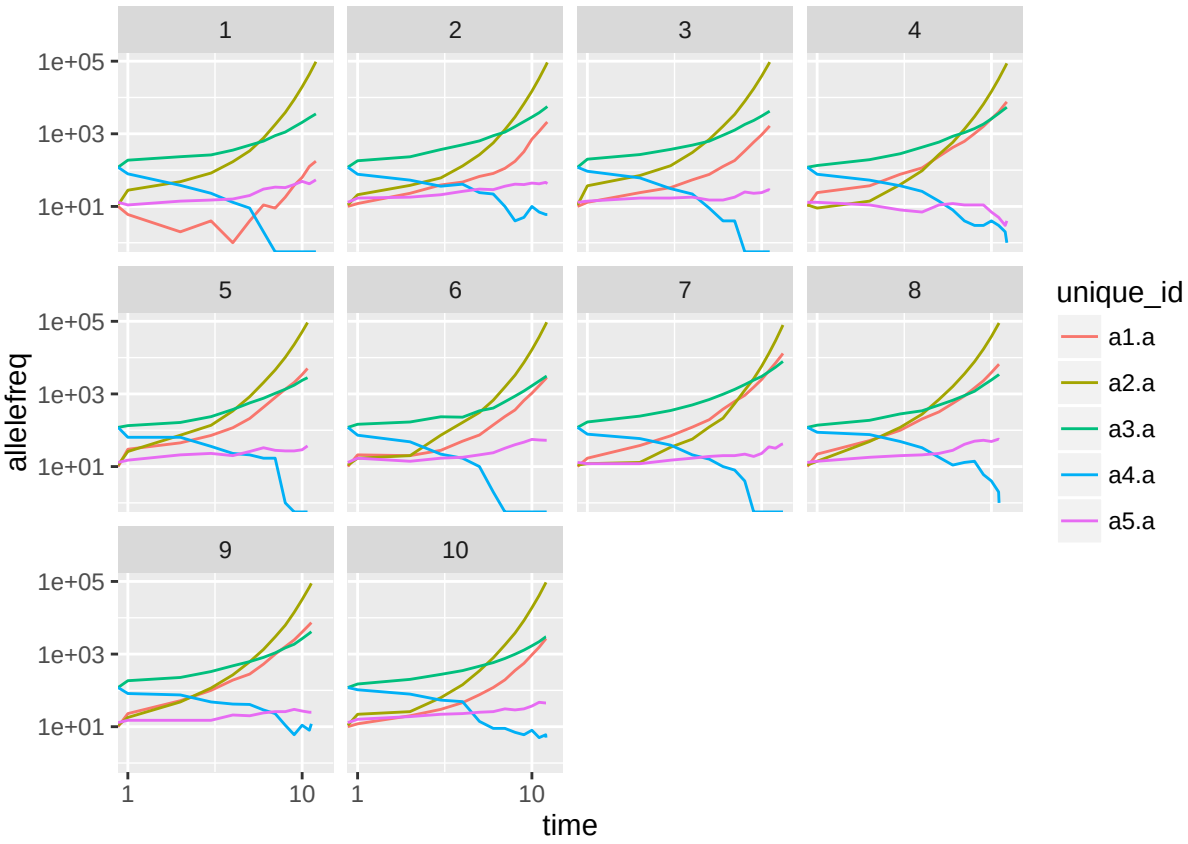
```
barcode <- clonedata %>% filter(unique_id %in% c('a1.a', 'a2.a', 'a3.a', 'a4.a', 'a5.a'))
barcode %>%
  ggplot(aes(x = count_alleles)) + geom_histogram() + scale_x_log10() +
  facet_wrap(~run)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The trajector of ancestors can also be plotted using the time course data.

```
timedata %>% filter(unique_id %in% c('a1.a', 'a2.a', 'a3.a', 'a4.a', 'a5.a')) %>%
  ggplot(aes(x = time, y = allelefreq, colour = unique_id)) +
  geom_line() + scale_x_log10() + scale_y_log10() +
  facet_wrap(~run)
```


1
2
3
4
5
6
7
8
9
10
1e+05
1e+03
1e+01
allelefreq
1
10
time
unique_id
a1.a
a2.a
a3.a
a4.a
a5.a