two-type-inference

In this vignette, we consider the problem of inferring birth, death, and mutation rates for a two-type Markov branching process in which either cell type is capable of mutating into the other at any time. If you've already looked at `first-model.Rmd` and `one-type-double-logistic.Rmd`, most of this should seem pretty familiar. First we load the `bpinference` package.

```
library(bpinference)
#> Loading required package: rstan
#> Loading required package: StanHeaders
#> Loading required package: ggplot2
#> rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)
#> For execution on a local, multicore CPU with excess RAM we recommend calling
#> options(mc.cores = parallel::detectCores()).
#> To avoid recompilation of unchanged Stan programs, we recommend calling
#> rstan_options(auto_write = TRUE)
```

Next, we define the structure of our model. For details on how this works see `first-model.Rmd`.

```
e_mat =  rbind(c(2, 0),c(1,1),c(1,1), c(0,2), c(0,0),c(0,0))
p_vec = c(1, 1, 2, 2, 1, 2)
func_deps = c('c[1]','c[2]','c[3]', 'c[4]', 'c[5]', 'c[6]')
n_params <- 6
n_deps <- 0
mod = bp_model(e_mat, p_vec, func_deps, n_params, n_deps)
```

We now have a `bp_model` object which we can use for simulation and inference. Let's confirm that the sample moments we get from simulating the process agree with what we should expect analytically. To make simulation faster we'll start with a small number of initial cells. This will means that the data won't be perfectly normal, but the moments should still agree with our analytical results regardless of the exact distribution of the data.

```
simulation_params = c(0.20, 0.10, 0.05, 0.20, 0.25, 0.20)
z0 <- c(100, 50) # initial population vector
times <- 1
simulation_dat <- bpsims(mod, simulation_params, z0, times, 5000)
mom <- calculate_moments(e_mat, p_vec, simulation_params, z0, 1)
print(paste("true mean vector:", toString(mom$mu_mat), "sample mean vector:", toString(c(mean(simulatio
#> [1] "true mean vector: 97.8054257205992, 59.8852387358044 sample mean vector: 97.8966, 60.0432"
print(paste("true covaraince matrix:", toString(mom$sigma_mat), "sample mean vector:", toString(cov(simu
#> [1] "true covaraince matrix: 45.0418961482491, 3.02422137683959, 3.02422137683959, 32.0727485045799
```

Now that we've confirmed our simulation is accurate, we'll generate a more realistic dataset with more ancestor cells and fewer replicates. After getting the data we define our priors, generate a stan model, and start sampling.

```
z0 <- c(1000, 500) # initial population vector
times <- seq(0,5)
simulation_dat <- bpsims(mod, simulation_params, z0, times, 50)
stan_dat <- stan_data_from_simulation(simulation_dat, mod)
priors <- rep(list(prior_dist(name="normal",params=c(0,.25), bounds=c(0,5))),6)
generate(mod, priors, "two_type_inference.stan")
ranges <- matrix(rep(c(0,1), mod$nparams), mod$nparams, 2,byrow = T)
init <- uniform_initialize(ranges, 4)
```

```
options(mc.cores = parallel::detectCores())
stan_mod <- stan_model(file = "two_type_inference.stan")
fit_data <- sampling(stan_mod, data = stan_dat, control = list(adapt_delta = 0.95), chains = 4, refresh
samples <- data.frame(extract(fit_data))
```

Let's visualize our posteriors. We'll do this using violin plots:

```
samp_df = reshape::melt(samples[,1:6])
#> Using  as id variables
ggplot(samp_df, aes(x=factor(variable),y=value)) + geom_violin() + geom_hline(yintercept = .2) + geom_hl
```