1 Sample simulation with island population structure

The following code implements a landscape with 10 populations and no migration. Each population has two demographic stages.

```
> library(rmetasim)
> seedMigrationRate <- 0
> pollenMigrationRate <- 0
> habitats <- 10
> carryperpop <- 400
> stages <- 2
> rland <- NULL
> rland <- new.landscape.empty()
> rland <- new.intparam.land(rland, h = habitats, s = stages, totgen = 101)
> rland <- new.switchparam.land(rland, mp = 0)
> rland <- new.floatparam.land(rland)
> S <- matrix(c(0, 0, 1, 0), nrow = 2, byrow = TRUE)
> R <- matrix(c(0, 1.1, 0, 0), nrow = 2, byrow = TRUE)
> M <- matrix(c(0, 0, 0, 1), nrow = 2, byrow = TRUE)
> rland <- new.local.demo(rland, S, R, M)
> rland <- new.epoch.island(rland, 0, c(0, 0), c(0, 0), seedMigrationRate,
+     c(1, 0), c(1, 0), pollenMigrationRate, c(0, 1), c(0, 1),
+     carry = rep(carryperpop, habitats))
> rland <- new.locus(rland, type = 2, ploidy = 1, transmission = 1,
+     numalleles = 4, allelesize = 100)
> for (x in 1:9) {
+     rland <- new.locus(rland, type = 2, ploidy = 2, transmission = 0,
+         numalleles = 4, allelesize = 100)
+ }
> for (x in 1:10) {
+     rland <- new.locus(rland, type = 1, ploidy = 2, transmission = 0,
+         numalleles = 4)
+ }
> rland <- new.individuals(rland, c(500, 0, 500, 0, 500, 0, 500,
+     0, 500, 0, 500, 0, 500, 0, 500, 0, 500, 0, 500, 0))
```

1.1 Calculate Weir's theta

Set the sample size of individuals to select from each population for calculation of θ

```
> sampsize <- 5
```

This code calculates Weir and Cockerham's θ Weir and Cockerham (1984), an estimator of Wright's measure of population structure F_{ST}. It samples a maximum of 5 individuals per population. This document uses low values for execution speed, but any number up to the population size can be used. Beware that large numbers consume large amounts of resources.

```
> theta.0 <- popstruct(landscape.to.rtheta(rland, sampsize))
> print(theta.0)
```

```
             theta      Fhat
 [1,]  0.018895349        NA
 [2,] -0.033060921 0.9628529
 [3,] -0.020318021 1.0865724
 [4,]  0.012748295 1.0139342
 [5,] -0.030662489 0.9937332
 [6,] -0.003333333 0.9866667
 [7,] -0.043989153 0.9761976
 [8,]  0.005214454 1.0047004
 [9,]  0.001695165 1.0613208
[10,] -0.013991163 1.0603829
[11,]  0.032186295 0.9611391
[12,] -0.026362559 0.9330569
[13,]  0.019281915 1.0106383
[14,]  0.032579787 1.0106383
[15,] -0.018175074 1.0148368
[16,] -0.001483680 1.0148368
[17,]  0.007942399 0.9085511
[18,] -0.012133767 1.0121338
[19,] -0.003206084 1.0468237
[20,]  0.021002710 0.8943089
```

```
> print(mean(theta.0[, 1]))
```

```
[1] -0.002758494
```

1.2 Run simulation

Simulation runs for 100 generations total. Every 20 generations, θ is calculated for the landscape. In addition, the state of the landscape is saved in three ways (as demonstration): to a r binary data file ("current.Rdata"), a metasim text file ("current.dat"), and as an element pushed onto an R list called l.exp (not actually written to disk). This example is certainly overkill in terms of formats, but it's probably a good idea to write the state of the landscape out periodically during a really long simulation.

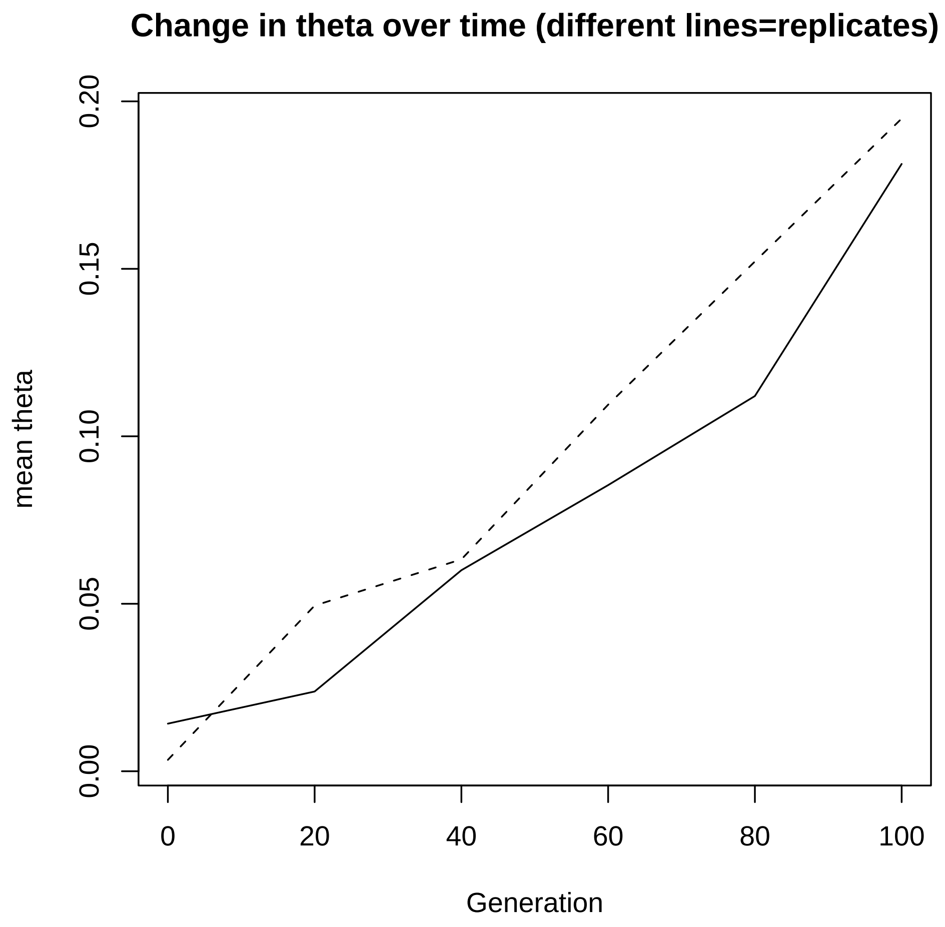
This code sets up a simulation and runs it numsteps*stepsize generations and repeats from the same starting conditions numreps times. The state of all sampled landscapes are stored in the structure l.exp. (l.exp[[1]][[2]] is the first replicate, second time-click, etc..).

```
> numreps <- 2
> numsteps <- 5
> stepsize <- 20
> l.exp <- list(numreps)
> rland.start <- rland
> for (j in 1:numreps) {
+     rland <- rland.start
+     l.exp[[j]] <- list(numsteps + 1)
+     l.exp[[j]][[1]] <- rland
+     for (i in 1:numsteps) {
+         rland <- simulate.landscape(rland, stepsize)
+         save(rland, file = "current.Rdata")
+         write.landscape(rland, "current.dat")
+         l.exp[[j]][[i + 1]] <- rland
+     }
+ }
```

1.3 Plot θ

Calculate θ as before. Plot θ over time. Each replicate is represented as a different line type.

```
> pstruc <- matrix(0, nrow = numsteps + 1, ncol = numreps)
> for (j in 1:numreps) {
+     for (i in 1:length(l.exp[[j]])) {
+         pstruc[i, j] <- mean(popstruct(landscape.to.rtheta(l.exp[[j]][[i]],
+             sampsize))[, 1])
+     }
+ }
> Gen <- c(0, (stepsize * c(1:numsteps)))
> ylimits <- range(pstruc)
> plot(pstruc[, 1] ~ Gen, main = "Change in theta over time (different lines=replicates
+     xlab = "Generation", ylab = "mean theta", type = "n", ylim = ylimits)
> for (i in 1:numreps) {
+     points(pstruc[, i] ~ Gen, type = "l", lty = i)
+ }
```



1.4 Write foreign files for import into other programs

These lines use write.landscape.foreign to write GDA files, one for the last time point for each replicate

```
> for (i in 1:numreps) {
+     lastland <- l.exp[[i]][[length(l.exp[[i]])]]
+     write.landscape.foreign(lastland, fn = paste("replicate",
+         i, ".nex", sep = ""), fmt = "GDA")
+ }
```

Last line of document

References

Weir, B. S. and Cockerham, C. C. (1984). Estimating F-statistics for the analysis of population structure, *Evolution* **38**: 1358–1370.