An introduction to the `metafolio` `R` package

Sean C. Anderson[1*], Jonathan W. Moore[1,2], Michelle M. McClure[3], Nicholas K. Dulvy[1] Andrew B. Cooper[2]

[1]Department of Biological Sciences, Simon Fraser University, Burnaby BC, V5A 1S6, Canada

[2]School of Resource and Environmental Management, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada

[3]Northwest Fisheries Science Center, National Marine Fisheries Service, Seattle, WA 98110, USA

The `metafolio` `R` package is a tool to simulate metapopulations and apply financial portfolio optimization concepts to those metapopulations. The package is primarily intended for salmon simulations, but could be adapted for other taxonomic groups. This vignette accompanies the paper *Portfolio conservation of metapopulations under climate change*. In this vignette we will describe the main functions in the `metafolio` package, show how to re-create the main analyses in our paper, and demonstrate some of the included plotting functions for exploring the output.

1 Installing the package

In an `R` console (version $\geq$ 3.0.0), you can install `metafolio` by first setting your working directory with `setwd()` to wherever you saved the package and then running:

```
install.packages("metafolio_0.3.0.tar.gz", repos = NULL,
  type = "source")
```

You can load the package with:

```
library(metafolio)
```

You can find a copy of this vignette and access the help files with:

```
vignette("metafolio")
?metafolio
help(package = "metafolio")
```

2 An example simulation

The main simulation function in `metafolio` is `meta_sim()`. We'll start by running a simulation using the base-case parameter values from the paper. First, we'll set up an `R` list object that contains the argument values for the stationary environmental stochasticity scenario.

```
arma_env_params <- list(mean_value = 16, ar = 0.1, sigma_env = 2,
  ma = 0)
```

The arguments `mean_value`, `ar`, and `ma` refer to the mean, autoregressive (AR), and moving-average (MA) components of an ARIMA model. See `?arima` for further explanation of ARIMA models in `R`. The argument `sigma_env` refers to the standard deviation of the environmental signal.

Next we'll run the simulation for one iteration. We'll simulate ten populations and re-assess the fishery every five years. Re-assessing the fishery means that the simulation fits a Ricker curve to the observed spawner-return data and updates the harvest-level targets. See the accompanying paper for details.

```
base1 <- meta_sim(n_pop = 10, env_params = arma_env_params,
  env_type = "arma", assess_freq = 5)
```

We can plot the output with the function `plot_sim_ts()`. The output is shown in Figure 1.

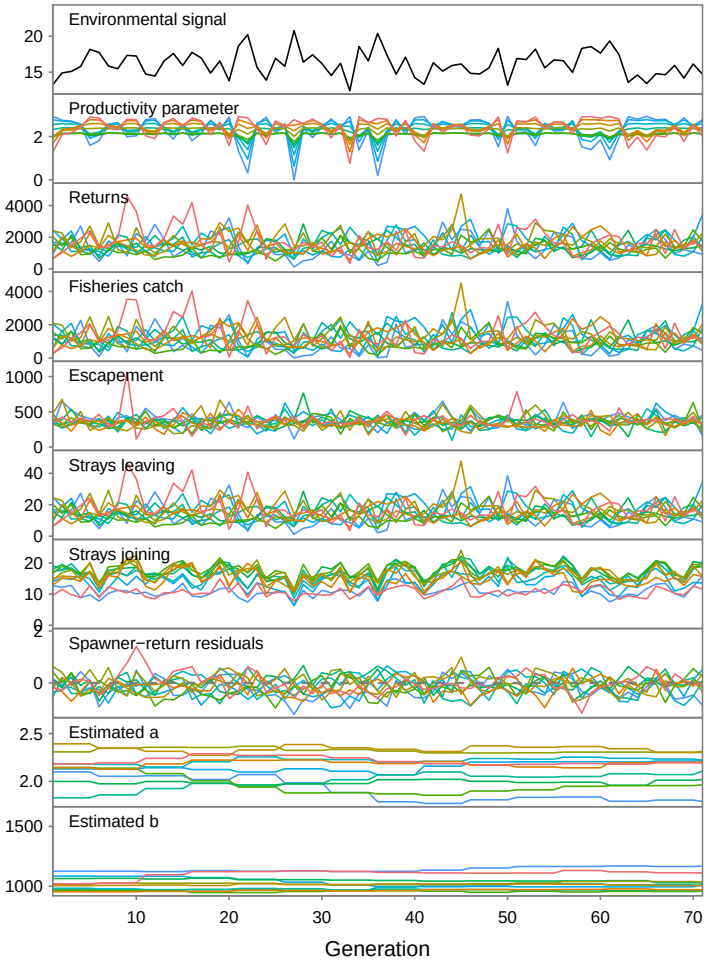

Figure 1: An example simulation with stationary environmental stochasticity and the base-case parameter values.

```
plot_sim_ts(base1, years_to_show = 70, burn = 30)
```

3 Exploring prioritization strategies

One of the key elements to the analysis in our paper is the choice of which populations to prioritize for conservation. We can try different prioritization strategies by manipulating the "investment weights" in each stream of salmon. In the case of salmon, we represent these with the Ricker b_i parameters, which indicate the maximum population capacity of streams i 1 through n.

In this example, we'll create one scenario in which we conserve response diversity from across the spectrum of possible responses and another scenario in which we conserve one half of the response diversity. We've set up the weights carefully so that each metapopulation contains the same number of populations (10) and the same total capacity. We set the Ricker b_i parameters equal to the nominal level of five salmon for the streams that we choose not to prioritize.

```
w_plans <- list()
w_plans[["balanced"]] <- c(5, 1000, 5, 1000, 5, 5, 1000, 5,
    1000, 5)
w_plans[["one_half"]] <- c(rep(1000, 4), rep(5, 6))
w <- list()
for(i in 1:2) { # loop over plans
 w[[i]] <- list()
 for(j in 1:80) { # loop over iterations
   w[[i]][[j]] <- matrix(w_plans[[i]], nrow = 1)
 }
}
```

We've now created a nested list of stream capacities (`w`). (`w` refers to the mnemonic "weights" for investment weights.) The first level of the list contains the two different scenarios The second level of the list contains the b_i values for each iteration. Each iteration will have re-sampled process noise and observation error when run with `meta_sim()`. Here, we're keeping the b_i values the same between iterations, but that might not be the case if we wanted to stochastically simulate the b_i values. We're only going to run 80 iterations to reduce the runtime of the example, but in reality you would likely want to run many more iterations.

We can now stochastically simulate with these strategies using the function `run_cons_plans()` and plot the output with `plot_cons_plans()` (Figure 2).

```
set.seed(1)
arma_sp <- run_cons_plans(w, env_type = "arma", env_params =
  arma_env_params)
```

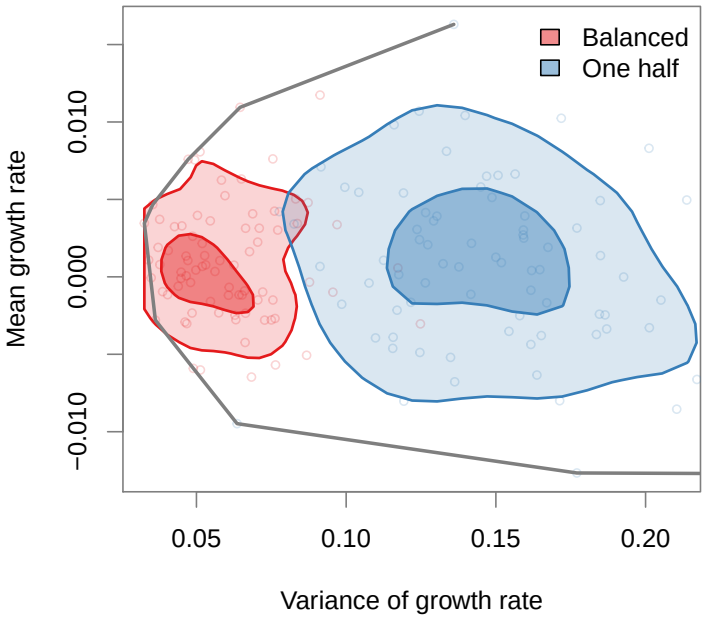

Figure 2: Two spatial conservation strategies shown in risk-return space with stationary environmental stochasticity. The dots show simulated metapopulations and the contours show 25% and 75% quantiles across 80 simulations per scenario. The grey line indicates the efficient frontier across all simulated metapopulations. The efficient frontier represents the minimum expected mean growth rate for a given expected variance in growth rate.

```
[1] "Completed 1 of 2 conservation plans to evaluate"
[1] "Completed 2 of 2 conservation plans to evaluate"


plot_cons_plans(arma_sp$plans_mv,
  plans_name = c("Balanced", "One half"),
  cols = c("#E41A1C", "#377EB8"), xlab = "Variance of growth rate",
  ylab = "Mean growth rate")
```

4 Generating alternative environmental time series

We can use the function `generate_env_ts()` to create alternative environmental time series. The function can generate five kinds of time series: sine waves, regime shifts, linear changes, and constant values. Each type has its own set of parameter arguments that are passed in a list format. See `?generate_env_ts` for examples of all of these.

We can see demonstrations of all environmental time series types with the default argument values with the following code (Figure 3).

```
types <- c("sine", "arma", "regime", "linear", "constant")
x <- list()
for(i in 1:5) x[[i]] <- generate_env_ts(n_t = 100, type = types[i])
par(mfrow = c(5, 1), mar = c(3,3,1,0), cex = 0.7)
for(i in 1:5) plot(x[[i]], type = "o", main = types[i])
```

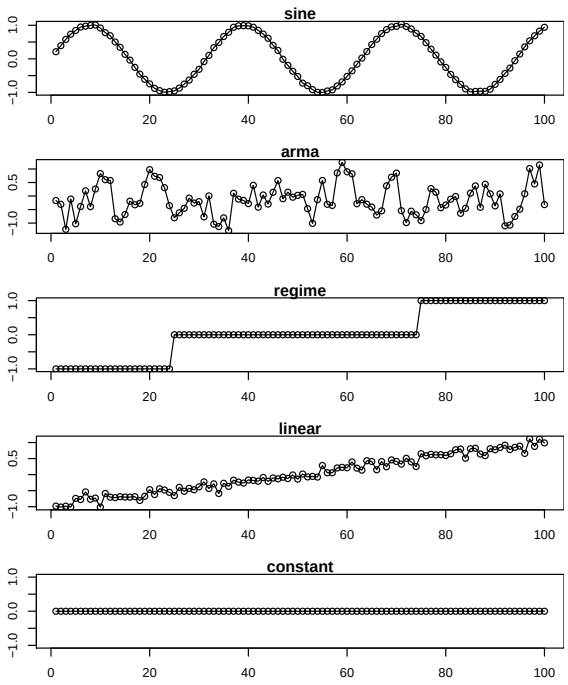

Figure 3: Example environmental time series.

5 Additional plotting functions

We can visualize the variability in the Ricker a parameters using the function `plot_rickers()` (Figure 4).

```
plot_rickers(base1, pal = rep("black", 10))
```

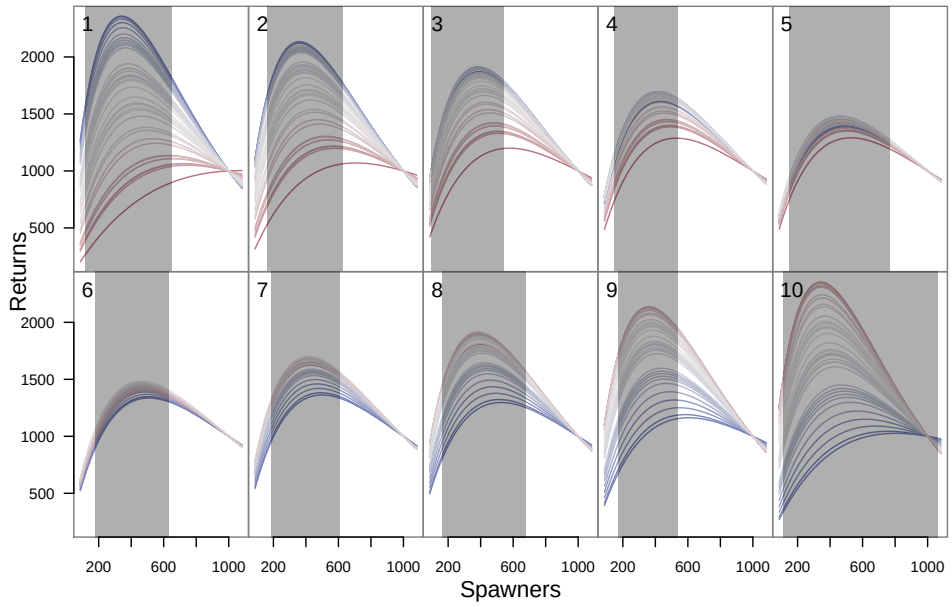

Figure 4: Ricker curves from a sample of 40 years in the example simulation. Each panel represents a different stream population. Population 1 is more productive in cool conditions and population 10 is more producitive in warm conditions. The colour of the Ricker curves represents the relative temperatue in that year (warm: red; cool: blue). The grey shaded area represents the variation in spawners observed within the simulation.

We can look at the correlation between salmon returns in the various streams using the function `plot_correlation_between_returns()` (Figure 5).

```
plot_correlation_between_returns(base1)
```

6 Optimizing metapopulation portfolios

A standard procedure in financial portfolio management is to determine optimal investment weights of the assets in a portfolio. The portfolios made up of these

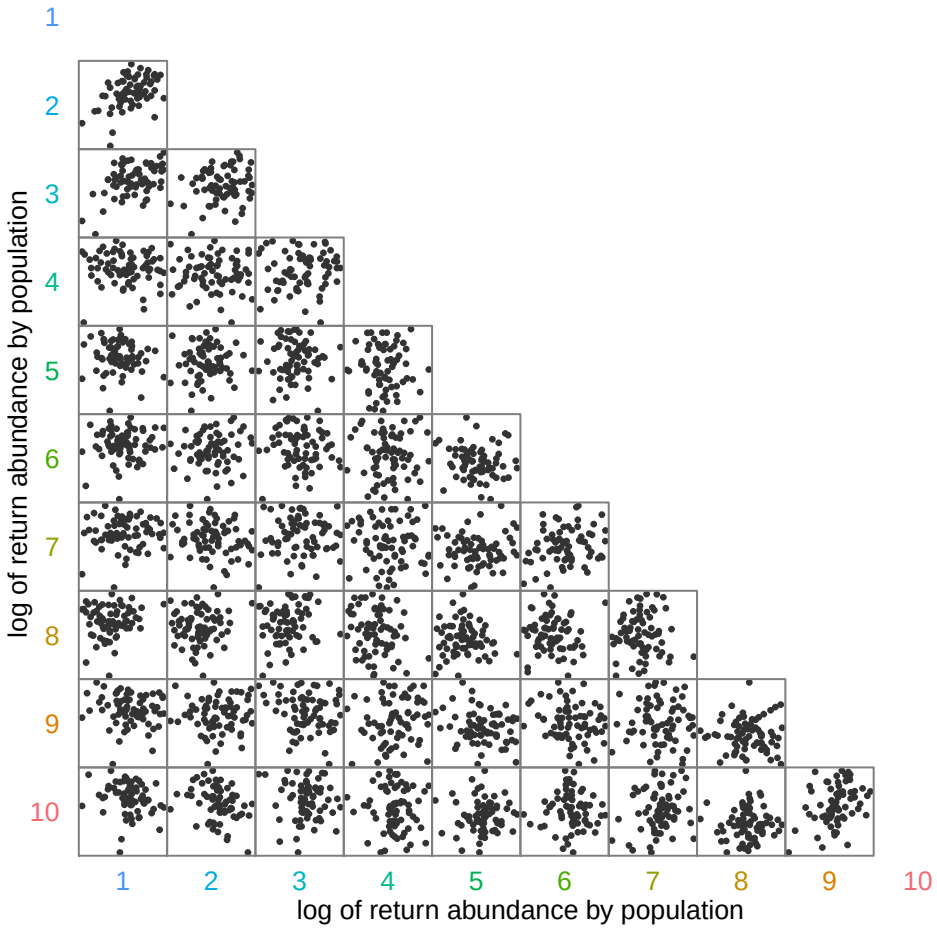

Figure 5: A plot comparing the log(returns) between each population. The population IDs are coloured from warm tolerant (warm colours) to cool tolerant (cool colours). Note how populations 1 and 10 have asynchronous returns whereas populations with more similar thermal-tolerance curves (say populations 9 and 10) have more synchronous dynamics. Populations with thermal tolerance curves in the middle (e.g. population 6) are less correlated with other populations. Their population dynamics end up primarily driven by demographic stochasticity and less so by temperature-induced systematic changes in productivity.

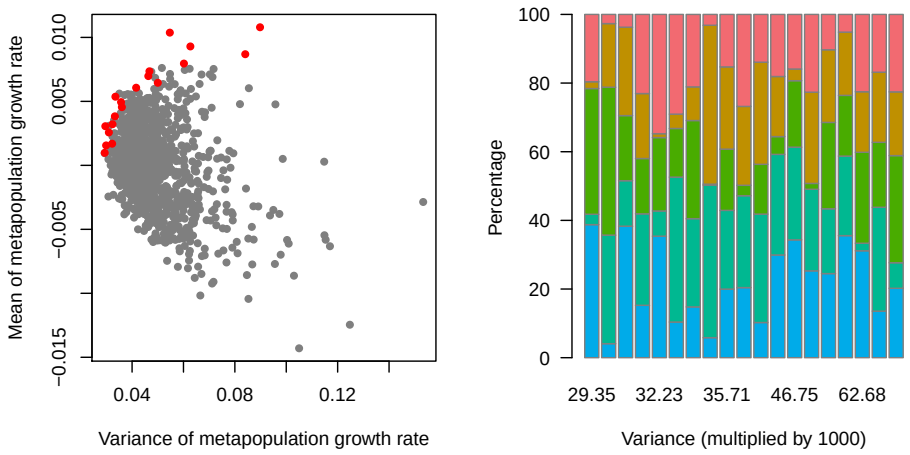

Figure 6: Efficient frontier of metapopulation portfolios (red dots). Each dot represents a different set of weights of the Ricker b parameters. The colours in the right panel correspond to the five populations with warm tolerant populations in warmer colours and cool tolerant populations in cooler colours.

optimal investment weights are referred to as the efficient frontier. This efficient frontier describes a set of portfolios which have minimal risk for a specified level of return or maximum return for a specified level of risk. While it would be complicated to determine the optimal metapopulation portfolios via algebra we can do so by letting `metafolio` sample from possible investment weights — a form of Monte Carlo sampling (Figure 6).

```
set.seed(999)
weights_matrix <- create_asset_weights(n_pop = 5, n_sims = 1000,
  weight_lower_limit = 0.001)
mc_ports <- monte_carlo_portfolios(weights_matrix = weights_matrix,
  n_sims = 1000, mean_b = 1000)
```

```
col_pal <- rev(gg_color_hue(5))
ef_dat <- plot_efficient_portfolios(port_vals = mc_ports$port_vals,
  pal = col_pal, weights_matrix = weights_matrix)
```