# Table of Contents

# Visualisation of random drift

Three functions are included: `random.drift` generates the time sequence of changes in a community caused by random drift; `plot.rd` and `barplot.rd` visualise these changes. Simplified version (with fewer parameters to modify) of this function is also published as a Shiny application.

To define the functions directly from this website, use the following script:

```
source
('http://www.davidzeleny.net/anadat-r/doku.php/en:customized_functions:random.drift?do=export_code&codeblock=1')
```

## Arguments

- `no.spe` Number of species. Integer.
- `no.ind` Number of individuals. Integer.
- `no.gen` Number of generations. Integer.
- `ratio` Ratio between species abundances (could be relative or absolute numbers). Default is that all species have the same abundance.
- `set.seed` Should we fix the current permutation? Logical value.
- `seed` Seed for the generator of pseudorandom values.
- `replicates` Number of simulations to plot in one plot as panels.
- `draw.plot` Should the function `random.drift` directly plot the result? Logical value.
- `x` Object returned by `random.drift` function.
- `to` How many generations should be ploted? Maximum is `no.gen`.
- `col` Color palette of the species. Defaults is `rainbow (no.spe)`. Vector of colors of the length equal to `no.spe`.
- `sep` Should be the species trends visually separated by a line? Logical value, default TRUE.
- `col.sep` Color of the line separating the species. Vector of the length equal to one. Default `white`.

random.drift.r

```
random.drift <- function (no.spe = 10, no.ind = 100, no.gen = 1000,
ratio = rep (1/no.spe, no.spe), set.seed = FALSE, seed = 1234,
replicates = 1, draw.plot = TRUE, ...)
{
  replicates <- as.numeric (replicates)
  ratio <- ratio/sum (ratio)*no.ind

  com.res.out <- list ()
  for (i in seq (1, replicates))
  {
    if (set.seed) set.seed (seed+i-1)
    com0 <- ordered (unlist (lapply (1:no.spe, FUN = function (n) rep
(n, ceiling (ratio[n])))), levels = 1:no.spe)
    com <- sample (com0, no.ind)
```

```
    com.res <- matrix (NA, nrow = no.spe, ncol = no.gen)
    com.res[,1] <- as.vector (table (com))
    for (n in seq (2, no.gen))
    {
      die.reproduce <- sample (1:no.ind, 2)
      com [die.reproduce[1]] <- com[die.reproduce[2]]
      com.res[,n] <- as.vector (table (com))
    }
    com.res.out[[i]] <- com.res
  }
  com.res.out$pars <- list (no.spe = no.spe, no.ind = no.ind, no.gen =
no.gen, ratio = ratio, replicates = replicates)
  if (draw.plot) plot.rd (com.res.out, ...)
  return (com.res.out)
}

plot.rd <- function (x, to = x$pars$no.gen, col = rainbow
(x$pars$no.spe), sep = TRUE, col.sep = "white", max.no.bars = 100,
panel.letter.add = FALSE, title.add = TRUE)
{
  if (to > x$pars$no.gen) stop ("Argument 'to' cannot be larger than
the number of generations in 'x'.")
  pars <- x$pars
  if (pars$replicates > 1) par (mfrow = c(ceiling (sqrt
(pars$replicates)), ceiling (sqrt (pars$replicates))))
  for (i in seq (1, pars$replicates))
  {
    com.res <- x[[i]][, 1:to]
    no.gen.temp <- ncol (com.res)
    if (no.gen.temp > max.no.bars) com.res.draw <- com.res[, ceiling
(seq (1, no.gen.temp, len = max.no.bars))] else com.res.draw <- com.res
    barplot (com.res.draw, space = 0, border = NA, col = col, xlab =
'Number of generations', ylab = 'Number of individuals')
    if (sep) matplot (x = seq (0, ncol (com.res.draw)), rbind (cumsum
(com.res.draw[,1]), t(apply (com.res.draw, 2, cumsum))), type = 'S',
col = col.sep, lty = 'solid', add = T)
    if (title.add) title (main = list (paste (pars$no.ind,
'individuals,', no.gen.temp, 'generations\nStart: ', sum
(com.res[,1]>0), 'species, end: ', sum (com.res[,no.gen.temp]>0),
'species'), font = 1))
    if (no.gen.temp > max.no.bars) {coef <- no.gen.temp/max.no.bars;
pretty.tick <- pretty (1:no.gen.temp); axis (1, at = pretty.tick/coef,
labels = pretty.tick )} else axis (1)
    if (panel.letter.add) legend ('topright', LETTERS[i], bty = 'n',
adj = 1, text.font = 2)
  }

}

barplot.rd <- function (x, to = x$pars$no.gen, col1 = "grey", col2 =
```

```
rainbow (x$pars$no.spe), first = 1, last = to, sort.by = 'none',
panel.letter.add = FALSE, title.add = TRUE, alpha = 0.5)
{
  if (to > x$pars$no.gen) stop ("Argument 'to' cannot be larger than
the number of generations in 'x'.")
  pars <- x$pars
  if (pars$replicates > 1) par (mfrow = c(ceiling (sqrt
(pars$replicates)), ceiling (sqrt (pars$replicates))))
  for (i in seq (1, pars$replicates))
  {
    com.res <- x[[i]][, 1:to]
    no.gen.temp <- ncol (com.res)
    if (sort.by == 'first') sorting <- order (com.res[,first],
decreasing = T)
    if (sort.by == 'last') sorting <- order (com.res[,last], decreasing
= T)
    if (sort.by == 'none') sorting <- 1:nrow (com.res)
    barplot (com.res[sorting,first], col = col1, ylim = c(0, max
(com.res[, c(first, last)])), xlab = 'Species', ylab = 'No
individuals')
    barplot (com.res[sorting,last], add = T, col = scales::alpha
(col2[sorting], alpha = alpha))
    if (title.add) title (main = list (paste (pars$no.ind,
'individuals,', no.gen.temp, 'generations\nStart: ', sum
(com.res[,1]>0), 'species, end: ', sum (com.res[,no.gen.temp]>0),
'species'), font = 1))
    if (panel.letter.add) legend ('topright', LETTERS[i], bty = 'n',
adj = 1, text.font = 2)
  }
}
```