# Package 'coexist'

February 19, 2015

**Type** Package

**Title** Species coexistence modeling and analysis

**Version** 1.0

**Date** 2012-08-01

**Author** Youhua Chen

**Maintainer** Youhua Chen <yhchen@zoology.ubc.ca>

**Description** species coexistence modeling under asymmetric dispersal and fluctuating source-sink dynamics;testing the proportion of coexistence scenarios driven by neutral and niche processes

**License** GPL (>= 2.0)

**Repository** CRAN

**Date/Publication** 2012-08-02 09:32:01

**NeedsCompilation** no

## R topics documented:

---

`coexist-package` *species coexistence modeling and analysis*

---

## Description

species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics; testing the proportion of coexistence scenarios driven by neutral process and niche processes respectively.

## Details

| | |
|---|---|
| Package: | coexist |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2012-08-01 |
| License: | GPL (>=2.0) |
| LazyLoad: | yes |

### Author(s)

Youhua Chen

Maintainer: Youhua Chen <yhchen@zoology.ubc.ca>

### References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

---

## batch.coexistence *batch anlaysis of coexistence summary tables*

---

### Description

batch analysis of basic statistics for each of the output scenarios under 2-species model as follows: number of parameter combination that lead to s1 species survive across the patches, while s2 species was extinct; number of parameter combination that lead to s2 species survive across the patches, while s1 species was extinct; number of parameter combination that lead to coexsistence of both both species across the patches; The other five parameters were "r1","r2","disp","comp1","comp2" respectively, representing the growth rates, patch connectivity, and competition ability of species 1 and 2.

### Usage

```
batch.coexistence(out, island = 10)
```

### Arguments

| | |
|---|---|
| `out` | a list of output scenarios for two-species model, each of which should be the combination of species' abundance across the sites by varying the parameters. |
| `island` | number of patches used in the modeling |

### Details

it is a batch form of sta.coexistence() function to handle different coexistence scenarios.

### Value

will return a list of matrices, each of which is identical to the output of sta.coexistence() function

### Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

### References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.coexistence, batch.monepar, batch.onepar, batch.mcoexistence

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (out, island = 10)
{
    colist <- list()
    scenarionum <- length(out)
    length(colist) <- scenarionum
    for (i in 1:scenarionum) {
        colist[[i]] <- sta.coexistence(out[[i]], island)
    }
    return(colist)
  }
```

---

`batch.mcoexistence` *batch anlaysis of multiple coexistence summary tables, the batch form for sta.mcoexistence() function*

---

## Description

batch analysis of basic statistics for each of the output scenarios under multiple-species model as follows: number of parameter combination that lead to s1 species survive across the patches, while s2 species was extinct; number of parameter combination that lead to s2 species survive across the patches, while s1 species was extinct; number of parameter combination that lead to coexsistence of both both species across the patches; The other five parameters were "r1","r2","disp","comp1","comp2" respectively, representing the growth rates, patch connectivity, and competition ability of species 1 and 2.

## Usage

```
batch.mcoexistence(out, island = 10, spnum = 2)
```

## Arguments

| | |
|---|---|
| `out` | |
| `island` | number of patches in the simulation, default is 10 |
| `spnum` | number of species in the simulation (>=2), default is 2 |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.mcoexistence

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (out, island = island, spnum = 2)
{
    colist <- list()
    scenarionum <- length(out)
    length(colist) <- scenarionum
    for (i in 1:scenarionum) {
        colist[[i]] <- sta.mcoexistence(out[[i]], island = island,
            spnum = spnum)
    }
    return(colist)
  }
```

---

batch.monepar — *batch analysis to explore multiple species coexistence density for a varying parameter under multiple-species modeling*

---

## Description

The batch form of sta.comparison() function, therefore handling different scenarios simultaneously.

## Usage

```
batch.monepar(coexistlist, coenum, island, spnum, parameters)
```

## Arguments

| | |
|---|---|
| coexistlist | a list of data generated by batch.mcoexistence() function |
| coenum | coexisting species number in a patch you want to explore across the scenarios. Should be >=2 and <=total species number |
| island | number of patches in the modeling |
| spnum | number of species in the moeling |

`parameters` a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the coexistence patch numbers under the cases when the parameter (for example, growth rate for species 1)=0.2,0.5 and 0.9 respectively.

## Value

will return a nested list, each list member is a list for one scneario, inside of which are the list members- matrices for each of possible parameters (for example, r1,r2,disp1,disp2,comp1,comp2,etc).

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`sta.mcomparison`, `batch.mpaircomp`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistlist, coenum, island, spnum, parameters)
{
    if (is.list(coexistlist)) {
        scenarionum <- length(coexistlist)
        pairlist <- list()
        length(pairlist) <- scenarionum
        for (i in 1:scenarionum) {
            pairlist[[i]] <- sta.mcomparison(coexistlist[[i]],
                coenum, island, spnum, parameters = parameters)
        }
        return(pairlist)
    }
  }
```

batch.mpaircomp *batch analysis to explore coexistence density for different scenarios of a pair of parameters, for the case of multiple species modeling*

## Description

batch version for the function sta.mpaircomparison()

## Usage

```
batch.mpaircomp(coexistlist,coenum,spnum,parameters)
```

## Arguments

| | |
|---|---|
| coexistlist | list of data generated by batch.mcoexistence() function |
| coenum | coexisting species number in a patch you want to explore across the scenarios. Should be >=2 and <=total species number |
| spnum | number of species in the model |
| parameters | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the coexistence patch numbers under the cases when each of the pairwise parameters (for example, growth rate and the competition ability of a species)=0.2,0.5 and 0.9 respectively. |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.mcomparison, sta.mpaircomparison

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistlist,coenum,spnum,parameters)
{
    if (is.list(coexistlist)) {
        scenarionum <- length(coexistlist)
```

```
        pairlist <- list()
        length(pairlist) <- scenarionum
        for (i in 1:scenarionum) {
            pairlist[[i]] <- sta.mpaircomparison(coexistlist[[i]],
                coenum, spnum = spnum, parameters = parameters)
        }
        return(pairlist)
    }
  }
```

---

`batch.n2n` *batch analysis of niche and neutral/nearly-neutral cases for two or multiple species modeling*

---

## Description

output will be used by plot_n2n() function to compare species coexistence status under niche and nearly-neutral parameter cases

## Usage

```
batch.n2n(colist,island)
```

## Arguments

| | |
|---|---|
| `colist` | output data from batch.coexistence() function for 2-species model, or batch.mcoexistence() funciton for multiple-species model |
| `island` | number of patches |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`plot_n2n`, `batch.coexistence`, `batch.mcoexistence`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (colist,island)
{
    resultlist <- list()
    scenarionum = length(colist)
    length(resultlist) <- scenarionum
    for (i in 1:scenarionum) {
        resultlist[[i]] <- sta.fitness(colist[[i]],island)
    }
    return(resultlist)
  }
```

---

`batch.onepar` *batch analysis to explore coexistence density for handling multiple model scenarios outputs for a varying parameter, for 2-species model*

---

## Description

batch version of sta.comparison() function

## Usage

```
batch.onepar(coexistlist, parameters)
```

## Arguments

| | |
|---|---|
| `coexistlist` | the basic statistics generated from batch.coexistence() function |
| `parameters` | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the coexistence patch numbers under the cases when the parameter (for example, growth rate for species 1)=0.2,0.5 and 0.9 respectively. |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`sta.coexistence`, `batch.coexistence`, `sta.comparison`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistlist, parameters = parspace)
{
    scenarionum <- length(coexistlist)
    pairlist <- list()
    length(pairlist) <- scenarionum
    for (i in 1:scenarionum) {
        pairlist[[i]] <- sta.comparison(coexistlist[[i]], parameters = parameters)
    }
    return(pairlist)
  }
```

---

`batch.paircomp` *batch analysis to explore coexistence density for handling different model scenarios for two focused parameters,for 2-species model*

---

## Description

batch version for sta.paircomparison() function

## Usage

```
batch.paircomp(coexistlist, parnum, parameters)
```

## Arguments

| | |
|---|---|
| `coexistlist` | list of data generated by batch.coexistence() function |
| `parnum` | number of parameters you want to choose and compare pairwisely in the model, should be less than the total number of parameters i.e.,, for two species model<=7. |
| `parameters` | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the coexistence patch numbers under the cases when each of the pairwise parameters (for example, growth rate and the competition ability of a species)=0.2,0.5 and 0.9 respectively. |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.coexistence, sta.paircomparison

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistlist, parnum = parnum, parameters = parspace)
{
    scenarionum <- length(coexistlist)
    pairlist <- list()
    length(pairlist) <- scenarionum
    for (i in 1:scenarionum) {
        pairlist[[i]] <- sta.paircomparison(coexistlist[[i]],
            parnum = parnum, parameters = parameters)
    }
    return(pairlist)
  }
```

---

batch.pdf.onepar *batch mode to plot matrix heatmap graphics for different model scenarios but only working on the sampling points of one parameter (x-axis) and generate pdf graphics*

---

## Description

batch mode to plot matrix heatmap graphics for different model scenarios but only working on the sampling points of one parameter (x-axis) and generate pdf graphics

## Usage

```
batch.pdf.onepar(parmatlist, pagesetup = c(2, 2), path = NULL)
```

## Arguments

| | |
|---|---|
| parmatlist | a list of data generated from batch.monepar() function |
| pagesetup | how many plots would print in a page of the pdf document, default is 2*2=4 plots |
| path | local disk path for saving the pdf graphics, if given, the file name will be changed by adding a postfix as "00yh"+a rand uniform number+".pdf" to avoid re-write another file with the same file name. If not given, the file name would be saved to a default path "c://outcome/" with a postfix "singleparameter"+ a random number+".pdf" |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.pdf.pairpar, batch.monepar

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (parmatlist, pagesetup = c(2, 2), path = NULL)
{
    scenarionum <- length(parmatlist)
    parnum <- length(parmatlist[[1]])
    if (length(path) != 0) {
        randnum <- runif(1)
        pos <- unlist(gregexpr("/", path))
        folder <- substr(path, 1, pos[length(pos)] - 1)
        dir.create(folder, showWarnings = F)
        filename = paste(path, "00yh", randnum, ".pdf", sep = "")
    }
    else {
        randnum <- runif(1)
        dir.create(folder, showWarnings = F)
        filename = paste(folder, "singleparameter", randnum,
            ".pdf", sep = "")
    }
    pdf(filename)
    par(mfrow = pagesetup)
    for (each in 1:parnum) {
        for (i in 1:scenarionum) {
            xname = paste("Model", i, sep = "-")
            title = names(parmatlist[[i]])[each]
            t <- parmatlist[[i]][[each]]
            t <- t[order(t[, 1], decreasing = F), ]
            t <- t[-1, -1]
            make.heatmap(t, xname = xname, xlab = c(0.1, 0.25,
                0.5, 0.75, 0.9), ylab = c(1:9), title = title)
        }
    }
    dev.off()
}
```

batch.pdf.pairpar *batch analysis to plot matrix heatmaps for pairwise parameter matrices for different scenarios and generate pdf graphics*

## Description

batch analysis to plot matrix values for pairwise parameter matrix

## Usage

```
batch.pdf.pairpar(parmatlist, pagesetup = c(2, 2), path = NULL)
```

## Arguments

| | |
|---|---|
| parmatlist | a list of data generated from batch.mpaircomp() function |
| pagesetup | how many plots would print in a page of the pdf document, default is 2*2=4 plots |
| path | local disk path for saving the pdf graphics, if given, the file name will be changed by adding a postfix as "00yh"+a rand uniform number+".pdf" to avoid re-write another file with the same file name. If not given, the file name would be saved to a default path "c://outcome/" with a postfix "pairwiseparameters"+ a random number+".pdf" |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.pdf.onepar, batch.monepar, batch.mpaircomp

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (parmatlist, pagesetup = c(2, 2), path = NULL)
{
    scenarionum <- length(parmatlist)
    parnum <- length(parmatlist[[1]][[1]])
    if (length(path) != 0) {
```

```
        randnum <- runif(1)
        pos <- unlist(gregexpr("/", path))
        folder <- substr(path, 1, pos[length(pos)] - 1)
        dir.create(folder, showWarnings = F)
        filename = paste(path, "00yh", randnum, ".pdf", sep = "")
    }
    else {
        randnum <- runif(1)
        dir.create(folder, showWarnings = F)
        filename = paste(folder, "pairwiseparameters", randnum,
            ".pdf", sep = "")
    }
    pdf(filename)
    par(mfrow = pagesetup)
    for (each in 1:parnum) {
        for (i in 1:scenarionum) {
            xname = paste("Model", i, sep = "-")
            title = names(parmatlist[[i]][[1]])[each]
            t <- parmatlist[[i]][[1]][[each]]
            t <- t(t)
            make.heatmap(t, xname = xname, xlab = c(0.1, 0.25,
                0.5, 0.75, 0.9), ylab = c(0.1, 0.25, 0.5, 0.75,
                0.9), title = title)
        }
    }
    dev.off()
}
```

---

`batch.read` *batch read different file data based on the order 1,2,3,4,5,6... in a folder*

---

## Description

batch version of read.data() function, can read all the files in a folder

## Usage

```
batch.read(path, index = NULL, spnum = 2, islandnum = 10)
```

## Arguments

| | |
|---|---|
| `path` | a vector recording all the files in a folder, which can be generated by convert.filenames() function |
| `index` | local fielname/path for recording all the parameter combination index matrix generated from make.parcomb() function |
| `spnum` | species number in the model, default is 2 |
| `islandnum` | patch number in the model,default is 10 |

## Details

return a list of data, each list member represents a model output

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

read.data, read.patchdata, make.parcomb

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (path = pathvector, index = NULL, spnum = 2, islandnum = 10)
{
    num <- length(path)
    outlist <- list()
    length(outlist) <- num
    if (length(index) == 0) {
        for (i in 1:num) {
            outlist[[i]] <- read.patchdata(path = path[i], spnum = spnum,
                islandnum = islandnum)
        }
    }
    if (length(index) != 0) {
        for (i in 1:num) {
            outlist[[i]] <- read.data(path = path[i], index = index,
                spnum = spnum, islandnum = islandnum)
        }
    }
    return(outlist)
  }
```

---

# comblist *construct a full list of all combinations of parameters*

---

## Description

similar to comblist2() function

## Usage

```
comblist(parvector, parnum)
```

## Arguments

| | |
|---|---|
| parvector | parvector is a parameter sampling point vector |
| parnum | parnum is the number of parameters |

## Value

return all combinations of parameter sampling point spaces

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

comblist2

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (parvector, parnum)
{
    combnum <- length(parvector)^parnum
    mat <- matrix(0, ncol = parnum, nrow = combnum)
    for (i in 1:parnum) {
        leg <- length(parvector)^(parnum - i)
        period = length(parvector)^i
        repeated = length(parvector)^(i - 1)
        fullcircle = leg * length(parvector)
        temp <- vector()
        length(temp) <- fullcircle
        for (k in 1:length(parvector)) {
            temp[((k - 1) * leg + 1):(k * leg)] = parvector[k]
        }
        for (ii in 1:repeated) {
            mat[((ii - 1) * fullcircle + 1):(ii * fullcircle),
                i] = temp
        }
    }
```

```
  return(mat)
}
```

---

comblist2 *construct a full list of all combinations of parameters*

---

## Description

similar to comblist() function

## Usage

```
comblist2(parlist)
```

## Arguments

| | |
|---|---|
| parlist | parlist is a parameter list, can vary on size |

## Value

return all possible value combinations of the parameter sampling spaces

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

comblist

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (parlist)
{
    combnum = 1
    parnum <- length(parlist)
    lvector <- vector()
    length(lvector) <- parnum
    for (i in 1:parnum) {
```

```
        combnum <- length(parlist[[i]]) * combnum
        lvector[i] = length(parlist[[i]])
    }
    mat <- matrix(0, ncol = parnum, nrow = combnum)
    for (i in 1:parnum) {
        leg <- prod(lvector[i:parnum])/lvector[i]
        repeated = combnum/prod(lvector[i:parnum])
        fullcircle = leg * lvector[i]
        temp <- vector()
        length(temp) <- fullcircle
        for (k in 1:length(parlist[[i]])) {
            temp[((k - 1) * leg + 1):(k * leg)] = parlist[[i]][k]
        }
        for (ii in 1:repeated) {
            mat[((ii - 1) * fullcircle + 1):(ii * fullcircle),
                i] = temp
        }
    }
    return(mat)
}
```

---

`competition` *perform competition analysis in the 2-species modeling*

---

## Description

this function is used to characterize species' competition across the patches for each time step

## Usage

```
competition(spvector, resource, comp1, comp2, grow, allee = 1)
```

## Arguments

| | |
|---|---|
| `spvector` | species-patch abundance matrix prior to dispersal in this time step |
| `resource` | carrying capability of the species at each patches, thus it is a matrix in terms of species-patches |
| `comp1` | competition coefficients for species 1 across patches |
| `comp2` | competition coefficients for species 2 across patches |
| `grow` | growth rates for both species across patches |
| `allee` | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species will be removed from that patch |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flex.competition, dispersal

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (spvector, resource, comp1, comp2, grow, allee = 1)
{
    islandnum <- dim(spvector)[2]
    for (i in 1:islandnum) {
        if (spvector[1, i] != 0 & spvector[2, i] != 0) {
            s1 <- spvector[1, i]
            s2 <- spvector[2, i]
            spvector[1, i] = s1 + (1 - comp1[1, i] * s1/resource[1,
                i] - comp1[2, i] * s2/resource[1, i]) * s1 *
                grow[1, i]
            spvector[2, i] = s2 + (1 - comp2[1, i] * s2/resource[2,
                i] - comp2[2, i] * s1/resource[2, i]) * s2 *
                grow[2, i]
        }
        if (spvector[1, i] != 0 & spvector[2, i] == 0) {
            s1 <- spvector[1, i]
            spvector[1, i] = s1 + (1 - comp1[1, i] * s1/resource[1,
                i]) * s1 * grow[1, i]
        }
        if (spvector[2, i] != 0 & spvector[1, i] == 0) {
            s2 <- spvector[2, i]
            spvector[2, i] = s2 + (1 - comp2[1, i] * s2/resource[2,
                i]) * s2 * grow[2, i]
        }
        if (spvector[1, i] < allee) {
            spvector[1, i] = 0
        }
        if (spvector[2, i] < allee) {
            spvector[2, i] = 0
        }
    }
    return(spvector)
}
```

compvar *competition parameters' matrix*

## Description

competition parameters' matrix

## Usage

```
compvar(island, rate = 0.5, scale = 2, type = "constant")
```

## Arguments

| | |
|---|---|
| island | patch number in the simulation |
| rate | basal dispersal rate across the patches |
| scale | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. |
| type | a model configuration vector describing the spatial patterns of each of the parameters, for example, a vector like this, c("decrease","decrease","decrease","increase","increase","increase", can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source patch to sink patches, so source patch the parameter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the parameter for the species there will be assigned a high value |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flex.competition, competition

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (island, rate = 0.5, scale = 2, type = "constant")
{
    comp <- matrix(0, ncol = island, nrow = 2)
    comp[1, ] = parsetting(island, rate, scale, type)
    comp[2, ] = 1 - comp[1, ]
    return(comp)
  }
```

---

`convert.filenames` *convert saved data sets' names in to a vector based on the order of numbers, which will be called by batch.read() function*

---

## Description

convert saved data sets' names in to a vector based on the order of numbers, which will be called by batch.read() function

## Usage

```
convert.filenames(folder)
```

## Arguments

| | |
|---|---|
| `folder` | the folder that stored all the model outputs, not a detailed filename |

## Value

return a vector for each member is a path pointed to a filename (a model output), which can be called by batch.read() function

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`read.data`, `batch.read`, `read.patchdata`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (folder)
{
    files <- list.files(path = folder, full.names = TRUE)
    newf <- vector()
    fnum <- length(files)
    length(newf) <- fnum
    dataorder <- rep(0, 1, fnum)
    for (i in 1:fnum) {
        pos.end <- unlist(gregexpr("00yh", files[i]))[1] - 1
        pos.start <- unlist(gregexpr(paste(folder, "/out", sep = ""),
            files[i]))[1] + 15
        dataorder[i] <- as.numeric(substr(files[i], pos.start,
            pos.end))
    }
    for (i in 1:fnum) {
        newf[dataorder[i]] <- files[i]
    }
    newf <- newf[!is.na(newf)]
    return(newf)
  }
```

---

`dispersal` *perform dispersal analysis for each simulation time step for 2-species modeling*

---

## Description

this function is used to characterize species' dispersal across the patches for each time step

## Usage

```
dispersal(spvector, initp,dismat, allee = 1)
```

## Arguments

| | |
|---|---|
| `spvector` | species-patch abundance matrix prior to dispersal in this time step |
| `initp` | initial population size for a species that will be released at the source patch at each time step |
| `dismat` | a dispersal matrix, or connectivity matrix among the patches (i.e., matrix is the same for all species). Different from flex.dispersal() function, which is a list of matrices, and allowed different species will have different dispersal rates across patches |

| | |
|---|---|
| allee | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species will be removed from that patch |

## Details

return an updated species-patch abundance matrix at next time step

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flex.dispersal, competition

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (spvector, initp, dismat, allee = 1)
{
    spvector <- spvector %*% dismat
    spvector[1, 1] = initp
    spvector[2, 1] = initp
    spvector[which(spvector < allee)] = 0
    return(spvector)
  }
```

---

# dispvar *dispersal parameters' matrix*

---

## Description

dispersal parameters' matrix

## Usage

```
dispvar(island, rate = 0.5, scale = 2, type = "decrease")
```

## Arguments

| | |
|---|---|
| island | patch number in the simulation |
| rate | basal dispersal rate across the patches |
| scale | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. |
| type | a model configuration vector describing the spatial patterns of each of the parameters, for example, a vector like this, c("decrease","decrease","decrease","increase","increase","increase", can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source patch to sink patches, so source patch the parameter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the parameter for the species there will be assigned a high value |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flex.dispersal, dispersal

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (island, rate = 0.5, scale = 2, type = "decrease")
{
    dismat <- matrix(0, ncol = island, nrow = island)
    if (type == "decrease") {
        for (i in 1:as.integer((island - 1)/2)) {
            dismat[i, i + 1] = rate
        }
        for (i in (as.integer((island - 1)/2) + 1):(island -
            1)) {
            dismat[i, i + 1] = rate/scale
```

```
        }
        diag(dismat) <- 1 - rowSums(dismat)
    }
    if (type == "increase") {
        for (i in 1:as.integer((island - 1)/2)) {
            dismat[i, i + 1] = rate/scale
        }
        for (i in (as.integer((island - 1)/2) + 1):(island -
            1)) {
            dismat[i, i + 1] = rate
        }
        diag(dismat) <- 1 - rowSums(dismat)
    }
    if (type == "constant") {
        for (i in 1:(island - 1)) {
            dismat[i, i + 1] = rate
        }
        diag(dismat) <- 1 - rowSums(dismat)
    }
    if (type == "mosaiclow") {
        for (i in 1:(island - 1)) {
            if (i%%2 == 0) {
                dismat[i, i + 1] = rate
            }
            else {
                dismat[i, i + 1] = rate/scale
            }
        }
        diag(dismat) <- 1 - rowSums(dismat)
    }
    if (type == "mosaichigh") {
        for (i in 1:(island - 1)) {
            if (i%%2 == 0) {
                dismat[i, i + 1] = rate/scale
            }
            else {
                dismat[i, i + 1] = rate
            }
        }
        diag(dismat) <- 1 - rowSums(dismat)
    }
    return(dismat)
}
```

---

`fast.flexsim` *faster multiple species simulation, but just a bit, not fast actually*

---

## Description

similar to flexsim() function

## Usage

```
fast.flexsim(scale = 2, island,dispersalscale = 51, allee = 1, T = 1000, initp,parcombination, spnum =
```

## Arguments

| | |
|---|---|
| scale | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. Specifically used in the spatial types. |
| island | number of patches in the simulation |
| dispersalscale | the scale value for reducing the dispersal rates between the neighbouring patches, allowing the simulation slow down or speed up. |
| allee | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species was removed from that patch |
| T | simulation time/steps |
| parcombination | the matrix listing out all combinations of parameter setting |
| initp | initial population size for a species that will be released at the source patch at each time step |
| spnum | species number in the model |
| sourcetype | model configuration for the source supply methods, currently there are three types of source species supplying modes: "constant": constant population size of each species will be released at the source patch for each simulation step "flexible": random population size of each species will be released (all species should typically be assigned different values) based on a normal distribution with mean=initp, Variance=initp/2, at the source patch for each simulation step "cochange": random population size (but all species will be assigned a same value) at the source patch for each simulation step |
| type | a model configuration vector describing the spatial patterns of each of the parameters, for example, a vector like this, c("decrease","decrease","decrease","increase","increase","increase", can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source patch to sink patches, so source patch the parameter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the parameter for the species there will be assigned a high value |
| path | local disk file name or folder to save the simulated outputs |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flexsim, sim.coarse

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (scale = 2, island,dispersalscale = 51, allee = 1, T = 1000,
    initp, parcombination, spnum = 2, sourcetype = "constant",
    type, path = NULL)
{
    parnum = spnum * 3
    parcomb <- parcombination
    outcome <- list()
    length(outcome) <- dim(parcomb)[1]
    resource <- matrix(0, ncol = island, nrow = spnum)
    grow <- resource
    comp <- resource
    spvector <- resource
    for (i in 1:spnum) {
        resource[i, ] <- parsetting(island, initp, scale, type[i])
    }
    outcomefile <- filename.check(path)
    comnum <- dim(parcomb)[1]
    colnames(parcomb) <- c(paste("r", c(1:spnum), sep = ""),
        paste("dis", c(1:spnum), sep = ""), paste("com", c(1:spnum),
            sep = ""))
    dismat <- list()
    length(dismat) <- spnum
    for (each in 1:comnum) {
        typenum <- spnum
        for (i in 1:spnum) {
            grow[i, ] <- parsetting(island, rate = parcomb[each,
                i], scale, type[typenum + i])
        }
        typenum <- spnum + typenum
        for (i in 1:spnum) {
            dismat[[i]] <- dispvar(island, rate = parcomb[each,
                i + spnum]/dispersalscale, scale, type[typenum +
                i])
        }
        typenum <- spnum + typenum
        for (i in 1:spnum) {
            comp[i, ] <- parsetting(island, rate = parcomb[each,
```

```
                i + spnum * 2], scale, type[typenum + i])
        }
        for (i in 1:spnum) {
            spvector[i, ] <- spabundance(island, 1000)
        }
        for (j in 1:T) {
            spvector <- flex.competition(spvector, resource,
                grow, comp, allee)
            spvector <- flex.dispersal(spvector, dismat, allee,
                type = sourcetype)
        }
        outcome[[each]] <- spvector
        write.table(outcome[[each]], file = outcomefile, sep = "\t",
            append = TRUE)
        outcome[[each]] <- list(abund = outcome[[each]], pars = parcomb[each,
            ])
    }
    return(outcome)
}
```

---

`filename.check` *open,check and create a new folder if not existed*

---

## Description

an internal function; please note that the default folder used for the simulation (if you don't point it out clearly, the package will allocate a path as "c://outcome")

## Usage

```
filename.check(path = NULL, return = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | local file path |
| `return` | if return=TRUE, it will return a checked new filename with adding the sign as "xxxx00yh0.xxxx.dat" format |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (path = NULL, return = TRUE)
{
    randnum <- runif(1)
    if (length(path) != 0) {
        pos <- unlist(gregexpr("/", path))
        dot <- unlist(gregexpr(".", path, fixed = T))
        dd <- unlist(gregexpr(":", path, fixed = T))
        special <- unlist(gregexpr("00yh", path))
        if (length(pos) >= 2 & dot[1] != -1 & dd[1] != -1 & special[1] !=
            -1) {
            folder <- substr(path, 1, pos[length(pos)] - 1)
            dir.create(folder, showWarnings = F)
            fname <- path
        }
        if (length(pos) >= 2 & dot[1] != -1 & dd[1] != -1 & special[1] ==
            -1) {
            folder <- substr(path, 1, pos[length(pos)] - 1)
            dir.create(folder, showWarnings = F)
            fname <- paste(substr(path, 1, dot[length(dot)] -
                1), "00yh", substr(path, dot[length(dot)], nchar(path)),
                sep = "")
        }
        if (dot[1] == -1 & dd[1] != -1 & special[1] == -1) {
            dir.create(path, showWarnings = F)
            fname <- paste(path, "/", randnum, "00yh", ".dat",
                sep = "")
        }
        if (dot[1] == -1 & dd[1] == -1 & special[1] == -1) {
            path <- paste("c://", path, sep = "")
            dir.create(path, showWarnings = F)
            fname <- paste(path, "/", randnum, "00yh", ".dat",
                sep = "")
        }
    }
    if (length(path) == 0) {
        if (length(folder) == 0) {
            folder = "c://outcome"
        }
        fname <- paste(folder, "/", randnum, "00yh", ".dat",
            sep = "")
        dir.create(folder, showWarnings = F)
    }
    if (return == T) {
        return(fname)
    }
  }
```

`flex.competition` *perform flexible competition analysis allowing multiple species*

## Description

this function is used to characterize species' competition (i.e., species population gain/loss due to competition) across the patches for each time step for multiple-species modeling

## Usage

```
flex.competition(spvector, resource, grow, comp, allee = 1)
```

## Arguments

| | |
|---|---|
| `spvector` | species-patch abundance matrix prior to dispersal in this time step |
| `resource` | carrying capability of the species at each patches, thus it is a matrix in terms of species-patches |
| `grow` | growth rate matrix for all species (rows) across patches (columns) |
| `comp` | competition coefficient matrix for the species (rows) across patches (columns) |
| `allee` | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species will be removed from that patch |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`competition`, `compvar`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (spvector, resource, grow, comp, allee = 1)
{
    spnum <- dim(spvector)[1]
    islandnum <- dim(spvector)[2]
```

```
    for (i in 1:islandnum) {
        s <- spvector[, i]
        for (sp in 1:spnum) {
            spvector[sp, i] = s[sp] + (1 - comp[sp, i] * s[sp]/resource[sp,
                i] - (1 - comp[sp, i]) * sum(s[-sp])/resource[sp,
                i]) * s[sp] * grow[sp, i]
            if (spvector[sp, i] < allee) {
                spvector[sp, i] = 0
            }
        }
    }
    return(spvector)
}
```

---

`flex.dispersal` *perform species-specific dispersal and fluctuating source analysis, for two or multiple species models, it's an internal function*

---

## Description

perform species-specific dispersal and fluctuating source analysis, for two or multiple species models, it's an internal function

## Usage

```
flex.dispersal(spvector, initp,dismat, allee = 1,type = "constant")
```

## Arguments

| | |
|---|---|
| `spvector` | species-patch abundance matrix prior to dispersal in this time step |
| `initp` | initial population size for a species that will be released at the source patch at each time step |
| `dismat` | a list of dispersal matrix for each of the species. The dispersal matrix for the species indicated its dispersal ability and perference across the patches. |
| `allee` | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species will be removed from that patch |
| `type` | model configuration for the source supply methods, currently there are three types of source species supplying modes: "constant": constant population size of each species will be released at the source patch for each simulation step "flexible": random population size of each species will be released (all species should typically be assigned different values) based on a normal distribution with mean=initp, Variance=initp/2, at the source patch for each simulation step "cochange": random population size (but all species will be assigned a same value) at the source patch for each simulation step |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flex.competition, dispersal

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (spvector, initp,dismat, allee = 1, type = "constant")
{
    if (type == "constant") {
        spnum <- length(dismat)
        for (i in 1:spnum) {
            spvector[i, ] <- spvector[i, ] %*% dismat[[i]]
            spvector[i, 1] = initp
        }
    }
    if (type == "flexible") {
        spnum <- length(dismat)
        for (i in 1:spnum) {
            spvector[i, ] <- spvector[i, ] %*% dismat[[i]]
            spvector[i, 1] = rnorm(1, mean = initp, sd = initp/10)
        }
    }
    if (type == "cochange") {
        spnum <- length(dismat)
        newresource <- rnorm(1, mean = initp, sd = initp/10)
        for (i in 1:spnum) {
            spvector[i, ] <- spvector[i, ] %*% dismat[[i]]
            spvector[i, 1] = newresource
        }
    }
    spvector[which(spvector < allee)] = 0
    spvector[which(spvector < 0)] = 0
    return(spvector)
  }
```

flexsim *species coexistence simulation for multiple-species modeling (>=2)*

## Description

all combinations of parameter setting can be input as in the argument item "prange"

## Usage

```
flexsim(scale = 2, dispersalscale = 51, allee = 1, T = 1000, prange, initp,spnum = 2, island=10, source
```

## Arguments

| | |
|---|---|
| scale | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. Specifically used in the spatial types. |
| dispersalscale | the scale value for reducing the dispersal rates between the neighbouring patches, allowing the simulation slow down or speed up. |
| allee | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species was removed from that patch |
| T | simulation time/steps |
| prange | the matrix listing out all combinations of parameter setting |
| initp | initial population size for a species that will be released at the source patch at each time step |
| spnum | number of species |
| island | number of patches |
| sourcetype | model configuration for the source supply methods, currently there are three types of source species supplying modes: "constant": constant population size of each species will be released at the source patch for each simulation step "flexible": random population size of each species will be released (all species should typically be assigned different values) based on a normal distribution with mean=initp, Variance=initp/2, at the source patch for each simulation step "cochange": random population size (but all species will be assigned a same value) at the source patch for each simulation step |
| type | a model configuration vector describing the spatial patterns of each of the parameters, for example, a vector like this, c("decrease","decrease","decrease","increase","increase","increase", can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source |

| | |
|---|---|
| | patch to sink patches, so source patch the parameter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the parameter for the species there will be assigned a high value |
| path | local disk file name or folder to save the simulated outputs |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sim.coarse, fast.flexsim

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (scale = 2, dispersalscale = 51, allee = 1, T = 1000,
    prange, initp,spnum = 2, island=10,sourcetype = "constant", type,
    path = NULL)
{
    parnum = spnum * 3
    outcome <- list()
    if (!is.list(prange)) {
        length(outcome) <- length(prange)^parnum
    }
    else {
        parlen <- vector()
        length(parlen) <- length(prange)
        for (i in 1:length(prange)) {
            parlen[i] <- length(prange[[i]])
        }
        length(outcome) <- prod(parlen)
    }
    resource <- matrix(0, ncol = island, nrow = spnum)
    grow <- resource
    comp <- resource
    for (i in 1:spnum) {
        resource[i, ] <- parsetting(island, initp, scale, type[i])
    }
    outcomefile = filename.check(path)
    if (!is.list(prange)) {
        parcomb <- comblist(prange, parnum)
```

```
    }
    if (is.list(prange)) {
        parcomb <- comblist2(prange)
    }
    comnum <- dim(parcomb)[1]
    colnames(parcomb) <- c(paste("r", c(1:spnum), sep = ""),
        paste("dis", c(1:spnum), sep = ""), paste("com", c(1:spnum),
            sep = ""))
    dismat <- list()
    length(dismat) <- spnum
    for (each in 1:comnum) {
        typenum <- spnum
        for (i in 1:spnum) {
            grow[i, ] <- parsetting(island, rate = parcomb[each,
                i], scale, type[typenum + i])
        }
        typenum <- spnum + typenum
        for (i in 1:spnum) {
            dismat[[i]] <- dispvar(island, rate = parcomb[each,
                i + spnum]/dispersalscale, scale, type[typenum +
                i])
        }
        typenum <- spnum + typenum
        for (i in 1:spnum) {
            comp[i, ] <- parsetting(island, rate = parcomb[each,
                i + spnum * 2], scale, type[typenum + i])
        }
        spvector <- rbind(spabundance(island, 1000), spabundance(island,
            1000))
        for (j in 1:T) {
            spvector <- flex.competition(spvector, resource,
                grow, comp, allee)
            spvector <- flex.dispersal(spvector, dismat, allee,
                type = sourcetype)
        }
        outcome[[each]] <- spvector
        write.table(outcome[[each]], file = outcomefile, sep = "\t",
            append = TRUE)
        outcome[[each]] <- list(abund = outcome[[each]], pars = parcomb[each,
            ])
    }
    return(outcome)
}
```

---

make.heatmap *make a heatmap based on matrix values*

---

## Description

make a heatmap based on matrix values

## Usage

```
make.heatmap(mat, type = "gray", xname = "x", yname = "y", xlab = NULL, ylab = NULL, title = NULL)
```

## Arguments

| | |
|---|---|
| mat | the coexistence density matrix under the two parameter value combination (x and y) |
| type | currently only support the "gray" density heatmap |
| xname | name for parameter x at the horizontal axis |
| yname | name for parameter y at the vertical axis |
| xlab | parameter x's sampling point vector, for example, c(.1,.3,.6,.9), something like that.. |
| ylab | parameter y's sampling point vector, for example, c(.1,.3,.6,.9), something like that.. |
| title | main text for the figure |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.pdf.pairpar, batch.pdf.onepar

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (mat, type = "gray", xname = "x", yname = "y", xlab = NULL,
    ylab = NULL, title = NULL)
{
    xrange <- dim(mat)[2] + 1
    yrange <- dim(mat)[1] + 1
    if (xrange >= yrange) {
        maxrange <- xrange
        xscale = 1
        yscale <- yrange/xrange
    }
    if (yrange > xrange) {
        maxrange <- yrange
        xscale <- xrange/yrange
```

```
        yscale = 1
    }
    maxnum = ceiling(max(as.vector(mat)))
    tmat <- t(mat)
    plot((1:maxrange) * xscale, (1:maxrange) * yscale, type = "n",
        xlab = xname, ylab = yname, axes = F)
    if (length(xlab) != 0) {
        axis(1, at = c(1:(xrange - 1)) + 0.5, labels = xlab)
    }
    else {
        axis(1, at = c(1:(xrange - 1)) + 0.5, labels = c(1:(xrange -
            1)))
    }
    if (length(ylab) != 0) {
        axis(2, at = c(1:(yrange - 1)) + 0.5, labels = ylab)
    }
    else {
        axis(2, at = c(1:(yrange - 1)) + 0.5, labels = c(1:(yrange -
            1)))
    }
    if (length(title) != 0) {
        mtext(title)
    }
    for (i in 1:(xrange - 1)) {
        for (j in 1:(yrange - 1)) {
            value = tmat[i, j]
            x1 = i
            x2 = i + 1
            y1 = j
            y2 = j + 1
            if (type == "gray") {
                rect(x1, y1, x2, y2, col = gray((maxnum - value)/maxnum),
                  border = gray((maxnum - value)/maxnum))
            }
        }
    }
}
```

---

make.parcomb *make the parameter combination index matrix*

---

## Description

make the parameter combination index matrix, this matrix is the index matrix that can be used in the read.data() function or batch.read() functions to

## Usage

```
make.parcomb(prange, parnum, path = NULL)
```

## Arguments

| | |
|---|---|
| prange | sampling points' vector for the parameter space interval[0~1], for example c(0.1,.3,.6,.9) |
| parnum | number of parameters in the model |
| path | local filen/path to save the parameter combination matrix |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

read.data, batch.read, read.patchdata

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (prange, parnum, path = NULL)
{
    if (!is.list(prange)) {
        parcomb <- comblist(prange, parnum)
    }
    if (is.list(prange)) {
        parcomb <- comblist2(prange)
    }
    if (length(path) != 0) {
        path = filename.check(path)
        write.table(parcomb, path, sep = "\t")
    }
    return(parcomb)
  }
```

---

# parsetting *rate vector for each species at each island, called widely by other functions*

---

## Description

set up rate matrix of species-patches for the simulation

## Usage

```
parsetting(island, rate = 1, scale = 2, type = "decrease")
```

## Arguments

| | |
|---|---|
| island | number of patches |
| rate | basal rate for setting up the matrix |
| scale | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. Specifically used in the spatial types. |
| type | a model configuration vector describing the spatial patterns of each of the paramters (default is "decrease"), for example, a vector like this, c("decrease","decrease","decrease","increase","incr can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source patch to sink patches, so source patch the paramter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the paramter for the species there will be assigned a high value |

## Details

will return a vector for recording species parameter value at each patch

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sim.coarse, flexsim

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (island, rate = 1, scale = 2, type = "decrease")
```

```
{
    parset <- vector()
    length(parset) <- island
    if (type == "decrease") {
        parset[1:as.integer(island/2)] = rate
        parset[(as.integer(island/2) + 1):island] = rate/scale
    }
    if (type == "increase") {
        parset[1:as.integer(island/2)] = rate/scale
        parset[(as.integer(island/2) + 1):island] = rate
    }
    if (type == "constant") {
        parset[1:island] = rate
    }
    if (type == "mosaiclow") {
        for (i in 1:island) {
            if (i%%2 == 0) {
                parset[i] = rate
            }
            else {
                parset[i] = rate/scale
            }
        }
    }
    if (type == "mosaichigh") {
        for (i in 1:island) {
            if (i%%2 == 0) {
                parset[i] = rate/scale
            }
            else {
                parset[i] = rate
            }
        }
    }
    return(parset)
  }
```

---

plot_n2n *plot distribution of niche and neutral coexistence patterns and generate pdf graphics*

---

## Description

make barplots to compare the model outcomes under niche and nearly-neutrality parameter settings

## Usage

```
plot_n2n(resultlist, island, pagesetup = c(1, 1), path = NULL)
```

## Arguments

| | |
|---|---|
| resultlist | output from batch.n2n() function |
| island | number of patches in the model |
| pagesetup | how many figures will be generated in one page |
| path | local disk filename for saving the pdf graphics; if path==NULL, a random filename will be generated under "c://outcome" folder |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.n2n, batch.coexistence, batch.mcoexistence

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (resultlist, island = island, pagesetup = c(1, 1), path = NULL)
{
    prop <- list()
    length(prop) <- length(resultlist)
    scenarionum <- length(resultlist)
    temp <- matrix(0, ncol = 2, nrow = island - 1)
    for (i in 1:scenarionum) {
        temp[, 1] = rev(resultlist[[i]][1:9, 1]/resultlist[[i]][island,
            1])
        temp[, 2] = rev(resultlist[[i]][1:9, 2]/resultlist[[i]][island,
            2])
        prop[[i]] <- temp
    }
    if (length(path) != 0) {
        randnum <- runif(1)
        pos <- unlist(gregexpr("/", path))
        folder <- substr(path, 1, pos[length(pos)] - 1)
        dir.create(folder, showWarnings = F)
        filename = paste(path, "00yh", randnum, ".dat", sep = "")
    }
    else {
        randnum <- runif(1)
        dir.create(folder, showWarnings = F)
```

```
        filename = paste(folder, "n2nbarplot", randnum, ".dat",
            sep = "")
    }
    pdf(filename)
    par(mfrow = pagesetup)
    for (i in 1:scenarionum) {
        nnplot <- barplot(prop[[i]], beside = T, axisnames = F,
            col = 1)
        y <- max(prop[[i]])
        text(nnplot[5, 1], y - 0.05, "Neutrality", font = 2,
            cex = 2)
        text(nnplot[5, 2], y - 0.05, "Niche", font = 2, cex = 2)
        axis(1, at = nnplot[, 1], labels = c(1, 2, 3, 4, 5, 6,
            7, 8, 9))
        axis(1, at = nnplot[, 2], labels = c(1, 2, 3, 4, 5, 6,
            7, 8, 9))
        mtext(paste("Model", i, sep = "-"))
    }
    dev.off()
}
```

---

`read.data` *read data with paramter combination index file*

---

## Description

read the data from one model/scenario output

## Usage

```
read.data(path = NULL, index = NULL, spnum = 2, islandnum=10)
```

## Arguments

| | |
|---|---|
| `path` | local filename for the model output |
| `index` | local fielname/path for recording all the paramter combination |
| `spnum` | species number in the model, default is 2 |
| `islandnum` | patch number in the model,default is 10 |

## Details

return a list of data, for each list member there is a paramater combination item in a vector form and an abundance matrix of species-patches

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.read, read.patchdata

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (path = NULL, index = NULL, spnum = 2, islandnum=10)
{
    if (is.character(index)) {
        indmat <- read.table(index, header = T)
        index = indmat
    }
    if (length(path) != 0) {
        raw <- scan(path, what = character(), sep = "\t")
        fileline <- length(count.fields(path))
        outlist <- list()
        length(outlist) <- fileline/(spnum + 1)
        sp <- matrix(0, nrow = spnum, ncol = islandnum)
        count = 0
        for (i in 1:length(raw)) {
            if (raw[i] == "V1") {
                count = count + 1
                for (j in 1:spnum) {
                  sp[j, ] = as.numeric(raw[(i + 10 + j + (j -
                    1) * islandnum):(i + 10 + j - 1 + j * islandnum)])
                }
                par <- index[count, ]
                names(par) <- c(paste("r", 1:spnum, sep = ""),
                  paste("dis", 1:spnum, sep = ""), paste("c",
                    1:spnum, sep = ""))
                outlist[[count]] <- list(abund = sp, pars = par)
            }
        }
    }
    return(outlist)
  }
```

---

read.patchdata *read species abundance data from the patches, an internal function*

---

## Description

read species abundance data from the patches, an internal function used by other functions

## Usage

```
read.patchdata(path = NULL, spnum = 2, islandnum = 10)
```

## Arguments

| | |
|---|---|
| path | local filename for the model output |
| spnum | species number in the model, default is 2 |
| islandnum | patch number in the model,default is 10 |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

batch.read, read.data,

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (path = NULL, spnum = 2, islandnum = island)
{
    if (length(path) != 0) {
        raw <- scan(path, what = character(), sep = "\t")
        fileline <- length(count.fields(path))
        outlist <- list()
        length(outlist) <- fileline/(spnum + 1)
        sp <- matrix(0, nrow = spnum, ncol = islandnum)
        count = 0
        for (i in 1:length(raw)) {
            if (raw[i] == "V1") {
                count = count + 1
                for (j in 1:spnum) {
                  sp[j, ] = as.numeric(raw[(i + 10 + j + (j -
                    1) * islandnum):(i + 10 + j - 1 + j * islandnum)])
                }
                outlist[[count]] <- list(abund = sp)
            }
```

```
        }
    }
    return(outlist)
  }
```

---

`sim.coarse` *2-species coexistence modeling based on the matrix listing out all combinations of parameter setting*

---

## Description

all combinations of parameter setting can be input as in the argument item "prange"

## Usage

```
sim.coarse(island=10,scale = 2, dispersalscale = 51, allee = 1, T = 1000, prange, type, initp, path = N
```

## Arguments

| | |
|---|---|
| `island` | number of patches |
| `scale` | controlling the parameter's low value=parameter original value/scale; while parameter's high value=parameter original value*scale. Specifically used in the spatial types. |
| `dispersalscale` | the scale value for reducing the connectivity probability between the neighbouring patches, allowing the simulation slow down or speed up. |
| `allee` | allee effect for the species, the minimum viable population in a local patch, default=1, indicating that if the population size in a patch for a species is less than 1, then the species was removed from that patch |
| `T` | simulation time/steps |
| `prange` | the matrix listing out all combinations of parameter setting |
| `initp` | initial population size for a species that will be released at the source patch at each time step |
| `type` | a model configuration vector describing the spatial patterns of each of the parameters, for example, a vector like this, c("decrease","decrease","decrease","increase","increase","increase", can be used as the input. There are 5 simple spatial types currently for the package: "decrease",the parameter will have one-time decreasing transition from source patch to other sink patches in the middle of the patches "increase",the parameter will have one-time increasing transition from source patch to other sink patches in the middle of the patches "constant",the parameter will keep in a constant value across the patches during the simulation "mosaiclow",the parameter will switch from a low value to a high value one-by-one from the source patch to sink patches, so source patch the parameter for the species there will be assigned a low value "mosaichigh",the parameter will switch from a high value to a low value one-by-one from the source patch to sink patches,so source patch the parameter for the species there will be assigned a high value |
| `path` | local disk file name or folder to save the simulated outputs |

## Value

a list of output, each list member has a parameter combination vector, a matrix of species-site abundance

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flexsim

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (scale = 2, dispersalscale = 51, allee = 1, T = 1000,
    prange = parspace, type = typevector, path = NULL)
{
    parnum = 5
    parlen <- length(prange)
    outcome <- list()
    length(outcome) <- parlen^parnum
    outindex <- matrix(0, nrow = parlen^parnum, ncol = parnum)
    colnames(outindex) <- c("r1", "r2", "dis", "c11", "c22")
    habitat1 <- parsetting(island, good, scale, type[1])
    habitat2 <- parsetting(island, good, scale, type[2])
    resource <- rbind(habitat1, habitat2)
    count = 0
    outcomefile = filename.check(path)
    for (i1 in 1:parlen) {
        for (i2 in 1:parlen) {
            for (i3 in 1:parlen) {
                for (i4 in 1:parlen) {
                  for (i5 in 1:parlen) {
                    grow1 <- parsetting(island, rate = prange[i1],
                      scale, type[3])
                    grow2 <- parsetting(island, rate = prange[i2],
                      scale, type[4])
                    grow <- rbind(grow1, grow2)
                    dismat <- dispvar(island, rate = prange[i3]/dispersalscale,
                      scale, type[5])
                    comp1 <- compvar(island, rate = prange[i4],
                      scale, type[6])
```

```
                    comp2 <- compvar(island, rate = prange[i5],
                      scale, type[7])
                    spvector <- rbind(spabundance(island, 1000),
                      spabundance(island, 1000))
                    count = count + 1
                    outindex[count, ] <- c(prange[i1], prange[i2],
                      prange[i3], prange[i4], prange[i5])
                    for (j in 1:T) {
                      spvector <- competition(spvector, resource,
                        comp1, comp2, grow, allee)
                      spvector <- dispersal(spvector, dismat,
                        allee)
                    }
                    outcome[[count]] <- spvector
                    write.table(outcome[[count]], file = outcomefile,
                      sep = "\t", append = TRUE)
                    outcome[[count]] <- list(abund = outcome[[count]],
                      pars = outindex[count, ])
                  }
                }
              }
            }
          }
          return(outcome)
        }
```

---

spabundance *initialization of species' abundance across the patches/islands*

---

## Description

initialization of species' abundance across the patches/islands prior to dispersal/competition interactions

## Usage

```
spabundance(island, abund = 1000)
```

## Arguments

| | |
|---|---|
| island | number of patches |
| abund | initial abundance of species at the source patch (leftmost) |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

flexsim, sim.coarse

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (island, abund = 1000)
{
    sabund <- vector()
    length(sabund) <- island
    sabund[1] = abund
    sabund[2:island] = 0
    return(sabund)
  }
```

---

# sta.coexistence *posterior coexistence analysis*

---

## Description

basic statistics for a single output scenario under 2-species model as follows: number of parameter combination that lead to s1 species survive across the patches, while s2 species was extinct; number of parameter combination that lead to s2 species survive across the patches, while s1 species was extinct; number of parameter combination that lead to coexsistence of both both species across the patches; The other five parameters were "r1","r2","disp","comp1","comp2" respectively, representing the growth rates, patch connectivity, and competition ability of species 1 and 2.

## Usage

```
sta.coexistence(outcome, island)
```

## Arguments

| | |
|---|---|
| outcome | A list of data for showing all species-site abundance matrices for each of all the combinations of varying parameters, just the direct output of fast.flexsim() or sim.coarse() functions. Alternatively, if the output of the simulation scenarios were stored in the local disk, you can use batch.read() or read.data() functions to get the data for sta.coexistence function |
| island | number of patches in the modeling |

## Details

can only handle two-species model, for multiple-species modeling, use sta.mcoexistence() function

## Value

return a matrix, of columns were the counting of patches that have allowed the survival of species 1 only, counting of patches that have allowed the survival of species 2 only, counting of patches that have allowed both species coexist; the others are values for each parameter combination

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.mcoexistence, batch.coexistence

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (outcome, island)
{
    sta <- matrix(0, ncol = 3 + length(outcome[[1]]$pars), nrow = length(outcome))
    for (i in 1:length(outcome)) {
        s12 = 0
        s1 = 0
        s2 = 0
        for (j in 1:island) {
            if (outcome[[i]]$abund[1, j] != 0 & outcome[[i]]$abund[2,
                j] != 0) {
                s12 = s12 + 1
            }
            if (outcome[[i]]$abund[1, j] != 0 & outcome[[i]]$abund[2,
                j] == 0) {
                s1 = s1 + 1
            }
            if (outcome[[i]]$abund[1, j] == 0 & outcome[[i]]$abund[2,
                j] != 0) {
                s2 = s2 + 1
            }
        }
        sta[i, 1] = s1
```

```
        sta[i, 2] = s2
        sta[i, 3] = s12 - 1
        sta[i, 4:dim(sta)[2]] = outcome[[i]]$pars
        colnames(sta) <- c("s1win", "s2win", "coexist", "r1",
            "r2", "disp", "comp1", "comp2")
    }
    return(sta)
}
```

---

`sta.comparison` *posterior different parameter rate comparison for a single scenario of 2-species modeling*

---

## Description

Analyze and compare the number of patches that have allowed the coexistence of both species for each of the sampling points of the single focused parameter (while partialling out the influence of other parameters by taking average).

## Usage

```
sta.comparison(coexistence, parameters,island)
```

## Arguments

| | |
|---|---|
| `coexistence` | the basic statistics generated from sta.coexistence() function |
| `parameters` | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the number of patches where coexistence emerged (coexisting species number>=2) under the cases when the parameter (for example, growth rate for species 1)=0.2,0.5 and 0.9 respectively. |
| `island` | number of patches in the model |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`sta.mcomparison`, `sta.coexistence`, `batch.coexistence`, `batch.onepar`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistence, parameters,island)
{
    comparisonlist <- list()
    length(comparisonlist) <- dim(coexistence)[2] - 3
    conum <- matrix(0, ncol = length(parameters) + 1, nrow = island)
    for (pars in 1:length(comparisonlist)) {
        for (i in 1:(island - 1)) {
            conum[i, 1] = island - i
            for (j in 1:length(parameters)) {
                conum[i, j + 1] <- length(which(coexistence[,
                  3] == island - i & coexistence[, pars + 3] ==
                  parameters[j]))
            }
        }
        for (j in 1:length(parameters)) {
            conum[island, j + 1] = length(which(coexistence[,
                pars + 3] == parameters[j]))
        }
        comparisonlist[[pars]] <- conum
    }
    names(comparisonlist) <- colnames(coexistence[, 4:dim(coexistence)[2]])
    return(comparisonlist)
  }
```

---

`sta.fitness` *fitness/abundance comparison for neutral versus niche cases for 2-species modeling*

---

## Description

using the output form sta.coexistence(), then divide the cases into two groups, one of which has the feature that both species have equal growth rates (neutrality or nearly-neutrality cases); and another group with unequal growth rates (niche cases). Then, the function checks the patch number where both species can coexist under each of the conflicting models (neutrality versus niche). For example, there are 5 patches in the simulation, under neutrality group, assuming we have 2000 parameter combination cases. Among the parameter combinations, 100 of which allowed both species coexist in all 5 patches, 300 of which allowed species coexist in 3 of 5 patches. Then, the data will be recorded as two rows. Output is a matrix, row number=patch number

## Usage

```
sta.fitness(coexistence,island)
```

### Arguments

| | |
|---|---|
| coexistence | using the output form sta.coexistence() function |
| island | number of patches in the modeling |

### Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

### References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

### Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistence)
{
    neutral <- coexistence[which(coexistence[, 4] == coexistence[,
        5]), ]
    niche <- coexistence[which(coexistence[, 4] != coexistence[,
        5]), ]
    neutral.num <- dim(neutral)[1]
    niche.num <- dim(niche)[1]
    conum <- matrix(0, ncol = 2, nrow = island)
    colnames(conum) <- c("neutral", "niche")
    for (i in 1:(island - 1)) {
        conum[i, 1] <- length(which(neutral[, 3] == island -
            i))
        conum[i, 2] <- length(which(niche[, 3] == island - i))
    }
    conum[island, 1] = neutral.num
    conum[island, 2] = niche.num
    return(conum)
  }
```

---

## sta.mcoexistence *basic posterior coexistence analysis for a single scenario of the multiple-species simulation*

---

### Description

The basic statistic for multiple-species (>=2) coexistence simulation, which counts the coexistence of different species in the patches, gathers all the combinations of varying parameters. Its counterpart is sta.coexistence() function, which is used to handle a single scenario of 2-species simulation

## Usage

```
sta.mcoexistence(outcome, island = 10, spnum)
```

## Arguments

| | |
|---|---|
| outcome | A list of data for showing all species-site abundance matrices for each of all the combinations of varying parameters, just the direct output of fast.flexsim() or sim.coarse() functions. Alternatively, if the output of the simulation scenarios were stored in the local disk, you can use batch.read() or read.data() functions to get the data for sta.mcoexistence function |
| island | number of patches used in the modeling, default is 10 |
| spnum | number of species considered (>=2) |

## Value

This function will return a matrix, of columns from left to right are the countings of patch that have allowed the survival of only a single species (each of the total species number), then followed by the countings of patch that have allowed the coexistence of 2 species, 3 species,... until the column that all the species can coexist. After these columns, the others are values for each parameter combination.

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.coexistence, batch.mcoexistence

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (outcome, island, spnum)
{
    conum <- spnum * 2
    sta <- matrix(0, ncol = conum + length(outcome[[1]]$pars),
        nrow = length(outcome))
    for (i in 1:length(outcome)) {
        for (j in 2:island) {
            num <- length(which(outcome[[i]]$abund[, j] != 0))
            if (num == 0) {
```

```
                sta[i, 1] = sta[i, 1] + 1
            }
            if (num == 1) {
                spindex <- which(outcome[[i]]$abund[, j] != 0)
                sta[i, spindex + 1] = sta[i, spindex + 1] + 1
            }
            if (num > 1) {
                sta[i, spnum + num] = sta[i, spnum + num] + 1
            }
        }
        sta[i, (conum + 1):dim(sta)[2]] = as.numeric(outcome[[i]]$pars)
    }
    colnames(sta) <- c(paste("s", c(0:spnum), sep = ""), paste("co",
        c(2:spnum), sep = ""), names(outcome[[i]]$pars))
    return(sta)
}
```

---

`sta.mcomparison` *posterior different parameter rate comparison for a single scenario of multiple-species modeling*

---

## Description

Analyze and compare the number of patches that allowed different coexistence cases (for example 2 species coexistence, 3 species coexistence and so on) for the sampling points of the focused parameter (while partialling out the influence of other parameters by taking average).

## Usage

```
sta.mcomparison(coexistence, coenum, island, spnum, parameters)
```

## Arguments

| | |
|---|---|
| `coexistence` | The output from sta.mcoexistence() function,which is a matrix, of columns from left to right are the countings of patch numbers that only allowed the survival of a single species (each of the total species number), then followed by the countings of patches that allowed the coexistence of 2 species, 3 species,... until the column that all the species can coexist. After these columns, the others are values for each parameter combination. |
| `coenum` | coexisting species number in a patch you want to explore. Should be >=2 and <=total species number |
| `island` | number of patches in the model |
| `spnum` | number of species in the model |
| `parameters` | a vector for showing sampling points for a parameter, for example c(.2,.5,.9), indicating three sampling points in a single parameter |

## Value

will return a list, each list member is a matrix for one parameter.

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.comparison, batch.monepar

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistence, coenum, island, spnum, parameters = parspace)
{
    comparisonlist <- list()
    length(comparisonlist) <- dim(coexistence)[2] - 2 * spnum
    conum <- matrix(0, ncol = length(parameters) + 1, nrow = island)
    colnames(conum) <- c("coe.num", paste("=", parameters, sep = ""))
    for (pars in 1:length(comparisonlist)) {
        for (i in 1:(island - 1)) {
            conum[i, 1] = island - i
            for (j in 1:length(parameters)) {
                conum[i, j + 1] <- length(which(coexistence[,
                  spnum + coenum] == island - i & coexistence[,
                  pars + 2 * spnum] == parameters[j]))
            }
        }
        for (j in 1:length(parameters)) {
            conum[island, j + 1] = length(which(coexistence[,
                spnum + coenum] == parameters[j]))
        }
        comparisonlist[[pars]] <- conum
    }
    names(comparisonlist) <- colnames(coexistence[, (2 * spnum +
        1):dim(coexistence)[2]])
    return(comparisonlist)
  }
```

`sta.mpaircomparison` *pairwise parameter comparison for multiple species with multiple parameter space*

## Description

explore coexistence patch number/density for a single scenario of a pair of parameters, for the case of multiple species modeling

## Usage

```
sta.mpaircomparison(coexistence, coenum, spnum, parameters)
```

## Arguments

| | |
|---|---|
| `coexistence` | list of data generated by sta.mcoexistence() function |
| `coenum` | coexisting species number in a patch you want to explore across the scenarios. Should be >=2 and <=total species number |
| `spnum` | number of species in the model |
| `parameters` | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the number of patches where coexistence emerged (coexisting species number>=2) under the cases when each of the pairwise parameters (for example, growth rate and the competition ability of a species)=0.2,0.5 and 0.9 respectively. |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

`sta.mcomparison`, `batch.mpaircomp`, `sta.paircomparison`

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistence, coenum, spnum, parameters = parspace)
```

```
{
    if (coenum > spnum | coenum <= 1) {
        cat("wrong number", "\n")
        coenum = spnum
    }
    parnum <- dim(coexistence)[2] - 2 * spnum
    comparisonlist <- list()
    length(comparisonlist) <- parnum * (parnum - 1)/2
    varlist <- comparisonlist
    namesvector <- vector()
    length(namesvector) <- length(comparisonlist)
    count = 0
    for (p1 in 1:(parnum - 1)) {
        for (p2 in (p1 + 1):parnum) {
            if (!is.list(parameters)) {
                conum <- matrix(0, ncol = length(parameters),
                  nrow = length(parameters))
                colnames(conum) = paste("=", parameters, sep = "")
                rownames(conum) = paste("=", parameters, sep = "")
                varmat <- conum
                count = count + 1
                for (i in 1:length(parameters)) {
                  for (j in 1:length(parameters)) {
                    temp <- coexistence[which(coexistence[, 2 *
                      spnum + p1] == parameters[i] & coexistence[,
                      2 * spnum + p2] == parameters[j]), ]
                    if (length(temp) == 0) {
                      conum[i, j] = 0
                      varmat[i, j] = 0
                    }
                    else {
                      conum[i, j] <- mean(temp[, spnum + coenum])
                      varmat[i, j] <- var(temp[, spnum + coenum])
                    }
                  }
                }
            }
            else {
                conum <- matrix(0, ncol = length(parameters[[p2]]),
                  nrow = length(parameters[[p1]]))
                colnames(conum) = paste("=", parameters[[p2]],
                  sep = "")
                rownames(conum) = paste("=", parameters[[p1]],
                  sep = "")
                varmat <- conum
                count = count + 1
                for (i in 1:length(parameters[[p2]])) {
                  for (j in 1:length(parameters[[p1]])) {
                    temp <- coexistence[which(coexistence[, 2 *
                      spnum + p1] == parameters[[p2]][i] & coexistence[,
                      2 * spnum + p2] == parameters[[p1]][j]),
                      ]
                    if (length(temp) == 0) {
```

```
                    conum[i, j] = 0
                    varmat[i, j] = 0
                  }
                  else {
                    conum[i, j] <- mean(temp[, spnum + coenum])
                    varmat[i, j] <- var(temp[, spnum + coenum])
                  }
                }
              }
            }
            comparisonlist[[count]] <- conum
            namesvector[count] <- paste(colnames(coexistence)[2 *
                spnum + p1], colnames(coexistence)[2 * spnum +
                p2], sep = "-")
            varlist[[count]] <- varmat
        }
    }
    names(comparisonlist) <- namesvector
    names(varlist) <- namesvector
    return(list(mean = comparisonlist, var = varlist))
}
```

---

`sta.paircomparison` *pairwise parameter comparison for 2-species model*

---

## Description

explore coexistence patch number/density for a single scenario of a pair of parameters, for the case of 2-species modeling

## Usage

```
sta.paircomparison(coexistence, parnum, parameters)
```

## Arguments

| | |
|---|---|
| `coexistence` | list of data generated by sta.coexistence() function |
| `parnum` | number of parameters you want to choose and compare pairwisely in the model, should be less than the total number of parameters i.e.,, for two species model<=7. |
| `parameters` | a parameter sampling point vector,for example parameters=c(.2,.5,.9), indicating three sampling points in a single parameter. The function will thus compare the coexistence patch numbers under the cases when each of the pairwise parameters (for example, growth rate and the competition ability of a species)=0.2,0.5 and 0.9 respectively. |

## Author(s)

Youhua Chen <yhchen@zoology.ubc.ca>

## References

Chen YH (2012) coexist: an R package for performing species coexistence modeling and analysis under asymmetric dispersal and fluctuating source-sink dynamics. http://code.google.com/p/coexist.

## See Also

sta.paircomparison, sta.coexistence, batch.paircomp

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (coexistence, parnum = parnum, parameters = parspace)
{
    comparisonlist <- list()
    length(comparisonlist) <- parnum * (parnum - 1)/2
    varlist <- comparisonlist
    namesvector <- vector()
    length(namesvector) <- length(comparisonlist)
    count = 0
    for (p1 in 1:(parnum - 1)) {
        for (p2 in (p1 + 1):parnum) {
            conum <- matrix(0, ncol = length(parameters), nrow = length(parameters))
            varmat <- conum
            count = count + 1
            for (i in 1:length(parameters)) {
                for (j in 1:length(parameters)) {
                  temp <- coexistence[which(coexistence[, 3 +
                    p1] == parameters[i] & coexistence[, 3 +
                    p2] == parameters[j]), ]
                  conum[i, j] <- mean(temp[, 3])
                  varmat[i, j] <- var(temp[, 3])
                }
            }
            comparisonlist[[count]] <- conum
            namesvector[count] <- paste(colnames(coexistence)[3 +
                p1], colnames(coexistence)[3 + p2], sep = "-")
            varlist[[count]] <- varmat
        }
    }
    names(comparisonlist) <- namesvector
    names(varlist) <- namesvector
    return(list(mean = comparisonlist, var = varlist))
}
```

# Index