

# Package ‘downscale’

May 19, 2015

**Type** Package

**Title** Downscaling Species Occupancy

**Version** 1.0

**Date** 2015-05-18

**Author** Charles Marsh [aut, cre]

**Maintainer** Charles Marsh <charliem2003@gmail.com>

**Description** A set of functions that downscales species occupancy at coarse grain sizes to predict species occupancy at fine grain sizes.

**License** GPL-2

**Depends** raster

**Imports** minpack.lm,  
Rmpfr,  
cubature,  
sp

**LazyData** true

**Collate** 'DataInput.R'  
'downscale.R'  
'OptimiseParameters.R'  
'OptimiseParametersGNB.R'  
'OptimiseParametersFNB.R'  
'OptimiseParametersLogis.R'  
'OptimiseParametersThomas.R'  
'PredictFunctions.R'  
'ResidFunctions.R'  
'StartingParams.R'  
'ThomasFunctions.R'  
'plot.predict.downscale.R'  
'predict.downscale.R'  
'ensemble.downscale.R'  
'upgrain.R'  
'ExtendRaster.R'  
'hui.downscale.R'  
'HuiFunctions.R'  
'upgrain.threshold.R'

R topics documented:

downscale-package . . . . .	2
downscale . . . . .	4
ensemble.downscale . . . . .	7
hui.downscale . . . . .	10
plot.predict.downscale . . . . .	12
predict.downscale . . . . .	13
upgrain . . . . .	16
upgrain.threshold . . . . .	19
<b>Index</b>	<b>23</b>

---

downscale-package	<i>Downscaling Species Occupancy</i>
-------------------	--------------------------------------

---

Description

The package provides a set of functions that model the occupancy-area relationship (OAR) of known coarse scale data. The models are then extrapolated to predict the proportion of occupied area at finer grain sizes.

Details

Package: downscale  
Type: Package  
Version: 1.0  
Date: 2015-05-18  
License: GPL-2

Overview

The package provides three sets of functions for each stage of analysis:

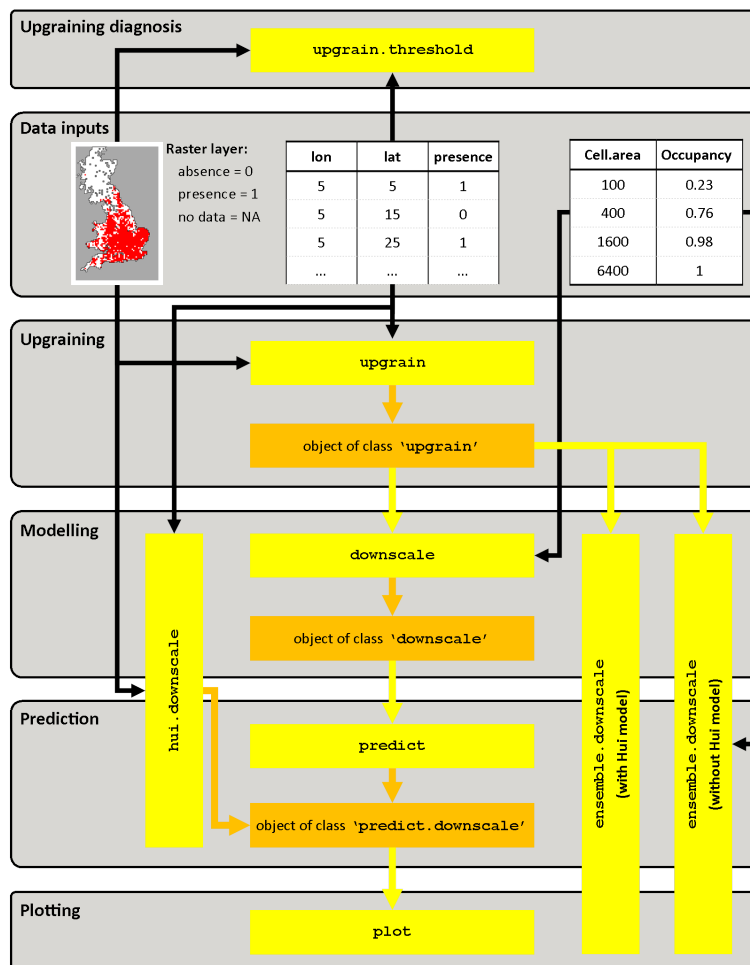
1) `upgrain` and `upgrain.threshold` prepare atlas data for downscaling.

2) `downscale` and `hui.downscale` model the OAR to the prepared data for one of ten possible downscaling models.

3) `predict.downscale` and `plot.predict.downscale` take the model outputs and predict occupancy at finer grains.

Finally, `ensemble.downscale` will run `downscale` and `predict.downscale` for a number of selected downscaling functions and calculate the mean predicted occupancies across all models.

The general flow of the package, and the inputs required for each function, is as follows:



Two vignettes are available to guide users. Both work through examples in code:

```
vignette("Downscaling", package = "downscale")
```

```
vignette("Upgraining", package = "downscale")
```

## Credits

This package was created as part of deliverable D3.2 of WP3 of the project: **EU-BON: Building the European Biodiversity Observation Network** - a 7th Framework Programme funded by the European Union under Contract No. 308454.

## Author(s)

Charles Marsh with input from Louise Barwell and Cang Hui.

Maintainer: Charles Marsh <charliem2003@gmail.com>

For reporting bugs or requesting information please include '`downscale`' in the subject line.

## References

Azaele, S., Cornell, S.J., & Kunin, W.E. (2012). Downscaling species occupancy from coarse spatial scales. *Ecological Applications* 22, 1004-1014.

Barwell, L.J., Azale, S., Kunin, W.E., & Isaac, N.J.B. (2014). Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* 20, 895-907.

Hui, C. (2009). On the scaling patterns of species spatial distribution and association. *Journal of Theoretical Biology* 261, 481-487.

Hui, C., McGeoch, M.A., & Warren, M. (2006). A spatially explicit approach to estimating species occupancy and spatial correlation. *Journal of Animal Ecology* 7, 140-147.

---

downscale

---

*Model area of occupancy against grain size for downscaling*


---

## Description

Fits the log observed proportion of occupancies against grain size for coarse-scale data (typically atlas data) for nine commonly used downscaling models (see Azale et al. 2012 and Barwell et al. 2014). See [hui.downscale](#) for downscaling using the Hui model. The parameters of the fitted models may then be used to estimate the area of occupancy at finer grain sizes than the observed data using [predict.downscale](#). Presence-absence atlas data can be prepared for downscaling using [upgrain](#).

## Usage

```
downscale(occupancies, model, extent = NULL, tolerance = 1e-06,
          starting_params = NULL)
```

## Arguments

occupancies	Either a data frame containing two columns or an object of class "upgrain" from the <a href="#">upgrain</a> function. If using a data frame the first column must be the grain sizes (cell area in squared units e.g. km <sup>2</sup> ). The second column is the proportion of occupancies at each grain size.
model	selected downscaling model, chosen from one of "Nachman", "PL", "Logis", "Poisson", "NB", "GNB", "INB", "FNB", "Thomas". See Details below for model descriptions.
extent	total area in same units as occupancy. If using an object of class "upgrain", this is automatically inputted.
tolerance	only applicable for the Thomas model. The tolerance used during integration in the Thomas model during optimisation of parameters. Lower numbers allow for greater accuracy but require longer processing times (default = 1e-6).
starting_params	a list of starting values for model parameters. Useful if the default values are not converging during optimisation. The parameter names must be the same for the default values (see Details for information on the parameters).

## Details

Nine downscaling models are available. area is the grain size (cell area) and extent the total area in the same units:

Model	Code	R code	Model equation
Nachman	"Nachman"	$\log(1 - \exp(-C * \text{area}^z))$	$1 - e^{-cA^z}$
Power law	"PL" "	$\log(C * \text{area}^z)$	$cA^z$
Logistic	"Logis"	$\log((C * (\text{area}^z)) / (1 + (C * (\text{area}^z))))$	$\frac{cA^z}{1 + cA^z}$
Poisson	"Poisson"	$\log(1 - (\exp(-\text{lambda} * \text{area})))$	$1 - e^{-\gamma A}$
Negative binomial	"NB"	$\log(1 - (1 + (C * \text{area}) / k)^{-k})$	$1 - \left(1 + \frac{\gamma A}{k}\right)^{-k}$
Generalised negative binomial	"GNB"	$\log(1 - (1 + (C * \text{area}^z) / k)^{-k})$	$1 - \left(1 + \frac{cA^z}{k}\right)^{-k}$
Improved negative binomial	"INB"	$\log(1 - ((C * \text{area}^{(b-1)})^{(r * \text{area})} / (1 - C * \text{area}^{(b-1))}))$	$1 - [c(\gamma A)^{b-1}]^{1 - c(\gamma A)^{b-1}}$
Finite negative binomial	"FNB"	$\log(1 - ((\text{gamma}(W + ((\text{extent} * k) / \text{area}) - k) * \text{gamma}(\text{extent} * k) / \text{area}) / (\text{gamma}(W + ((\text{extent} * k) / \text{area})) * \text{gamma}(((\text{extent} * k) / \text{area}) - k)))$	$1 - \frac{\Gamma\left(N + \frac{A_0 k}{A} - k\right) \Gamma\left(\frac{A_0 k}{A}\right)}{\Gamma\left(N + \frac{A_0 k}{A}\right) \Gamma\left(\frac{A_0 k}{A} - k\right)}$
Thomas	"Thomas"	see below	see below

The finite negative binomial model ("FNB") incorporates several gamma functions. This may result in integers larger than is possible to store in R. Therefore multiple precision floating point numbers ([mpfr](#) function in package **Rmpfr**) are used to make calculations possible.

The Thomas model incorporates spatial point processes in order to estimate species aggregations. This involves multi-dimensional integration which may be time-consuming. Users can alter the tolerance value during the integration process - a smaller value will give more accurate estimates but longer processing times.

The optimisation procedure requires initial starting values for all model parameters. In most cases the default values should work, however if the model is not converging adequately it is possible to input the starting parameters. The parameters must be in the form of a list with the same parameter names as in the table below. For example for the Nachman model the code would be `starting_params = list("C" = 0.1, "z" = 0.01)`. Please take particular note of capitals. The default starting parameters are:

"Nachman"	C = 0.01	z = 0.01	
"PL" "	C = 0.01	z = 0.01	
"Logis"	C = 0.01	z = 0.01	
"Poisson"	lambda = 1e-8		
"NB"	C = 0.01		
"GNB"	C = 0.00001	z = 0.1	k = 0.01
"INB"	C = 1	r = 0.01	b = 0.1
"FNB"	W = 10	k = 10	
"Thomas"	rho = 1e-8	mu = 10	sigma = 1

NOTE: for downscaling it is important to set occupancies above the scale of saturation (the grain size at which all cells are occupied) and the scale of saturation (the grain size where only a single cell is occupied) are not included for modelling. The downscale functions will automatically set these occupancies to NA.

## Value

downscale returns an object of class "downscale" containing four objects:

model	Downscaling model selected.	
pars	Estimated parameters for the downscaling model.	
observed	Data frame containing two columns:	
	Cell.area	Grain sizes for which occupancy have been observed.
	Occupancy	Observed area of occupancy for each grain size.
extent	Only for FNB and Thomas models.	

### Author(s)

Charles Marsh <<charliem2003@gmail.com>> with input from Louise Barwell.

### References

Azaele, S., Cornell, S.J., & Kunin, W.E. (2012). Downscaling species occupancy from coarse spatial scales. *Ecological Applications* 22, 1004-1014.

Barwell, L.J., Azaele, S., Kunin, W.E., & Isaac, N.J.B. (2014). Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* 20, 895-907.

### See Also

See [upgrain](#) for the preparation of presence-absence atlas data to occupancy data at several spatial scales.

The function output may be used as the input for [predict.downscale](#) for extrapolating downscaling functions to smaller grain sizes using the estimated parameters from the downscale output.

See [hui.downscale](#) for downscaling using the Hui model.

### Examples

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
head(atlas.data)

## explore thresholds using upgrain.threshold
thresh <- upgrain.threshold(atlas.data = atlas.data,
                           cell.width = 10,
                           scales = 3,
                           thresholds = seq(0, 1, 0.01))

## upgrain data (using All Presences threshold)
occupancy <- upgrain(atlas.data,
                    cell.width = 10,
                    scales = 3,
                    method = "All_Presences")

## Logistic model
(logis <- downscale(occupancies = occupancy,
                   model = "Logis"))
```

```
## Improved Negative Binomial model
(inb <- downscale(occupancies = occupancy,
                  model = "INB"))

## Specifying the starting parameters (gives a poorer fit in this case)
new.params <- list("C" = 0.1, "r" = 0.00001, "b" = 0.1)
(inb.new <- downscale(occupancies = occupancy,
                      model = "INB",
                      starting_params = new.params))

## plot the predictions of two FNB models using predict.downscale
predict(inb,
        new.areas = c(1, 2, 5, 25, 100, 400, 1600, 6400),
        plot = TRUE)
predict(inb.new,
        new.areas = c(1, 2, 5, 25, 100, 400, 1600, 6400),
        plot = TRUE)
```

---

ensemble.downscale      *Ensemble modelling of multiple downscaling functions*

---

## Description

Predict area of occupancy at fine grain sizes for multiple downscaling methods using [downscale](#) and [predict.downscale](#). Occupancies are converted to area of occupancy (AOO) by multiplying by the total extent. The mean of the logged predicted occupancies of all models is then calculated.

## Usage

```
ensemble.downscale(occupancies, new.areas, extent, cell.width = NULL,
                   models = "all", tolerance_mod = 1e-6, tolerance_pred = 1e-6,
                   tolerance_hui = 1e-6, starting_params = NULL, plot = TRUE,
                   verbose = TRUE)
```

## Arguments

occupancies	Either a data frame containing two columns or an object of class "upgrain" from the upgrain function. If using the Hui model (or model = "all") occupancies must be of class "upgrain". If using a data frame the first column must be the grain sizes (cell area in squared units e.g. km <sup>2</sup> ). The second column is the proportion of occupancies at each grain size.
new.areas	vector of grain sizes (in squared units e.g. km <sup>2</sup> ) for which area of occupancy will be predicted.
extent	total area in same units as occupancy. If using an object of class "upgrain", this is automatically inputted.
cell.width	the cell width of the atlas data. Only required for Hui model.
tolerance_mod	only applicable for the Thomas model. The tolerance used during integration in the Thomas model during optimisation of parameters. Lower numbers allow for greater accuracy but require longer processing times (default = 1e-6).

tolerance_pred	only applicable for the Thomas model. The tolerance used during the prediction stage.
tolerance_hui	only applicable for the Hui model. The tolerance used during integration in the Thomas model during optimisation of parameters. Lower numbers allow for greater accuracy but require longer processing times (default = 1e-6).
models	vector of chosen downscaling models. Default models = "all" runs all available models. See <a href="#">downscale</a> for list of available models.
starting_params	Starting values for model parameters if a model is not converging sufficiently. A list where each model to be specified is a list of parameter values. The parameter names must be the same for the default values (see Details for information on the parameters and how to specify them).
plot	if TRUE predictions of all models (red) are plotted against grain size along with the mean of all models (grey) and observed occupancies (black).
verbose	if TRUE prints updates on modelling status.

### Details

Ten downscaling models are available: "Nachman", "PL", "Logis", "Poisson", "NB", "GNB", "INB", "FNB", "Thomas" and "Hui". They can be input in any order, or all models run through models = "all". If the Hui model is included the input data must be an object of class "upgrain" generated through [upgrain](#). See [downscale](#) and [hui.downscale](#) for more details of the available models.

The optimisation procedure requires initial starting values for all model parameters. In most cases the default values should work, however if one or more models are not converging adequately it is possible to input the starting parameters. For each model we wish to specify, the parameters must be in the form of a list with the same parameter names as in the table below. starting\_params is then a list of these lists - the names of the lists are the same as the models. For example, if we wish to specify the starting parameters for the Nachman model and the Generalised Negative Binomial model the code would be:

```
starting_params = list(Nachman = list("C" = 0.1, "z" = 0.01),
GNB = list("C" = 0.1, "z" = 1, "k" = 0.01))
```

Please take particular note of capitals. The default starting parameters are:

"Nachman"	C = 0.01	z = 0.01	
"PL"	C = 0.01	z = 0.01	
"Logis"	C = 0.01	z = 0.01	
"Poisson"	lambda = 1e-8		
"NB"	C = 0.01		
"GNB"	C = 0.00001	z = 0.1	k = 0.01
"INB"	C = 1	r = 0.01	b = 0.1
"FNB"	W = 10	k = 10	
"Thomas"	rho = 1e-8	mu = 10	sigma = 1

### Value

Returns a list of two data frames: Occupancy = proportion of occupancies; A00 = occupancies converted to area of occupancy.

In each data frame the first column cell.area are the grain sizes used for predictions. The final column Means are the mean of the logged predictions of all models for each grain size. Intermediate columns are the predicted occupancies for the selected downscaling models.



**Author(s)**

Charles Marsh <<charliem2003@gmail.com>> with input from Louise Barwell.

**References**

Azaele, S., Cornell, S.J., & Kunin, W.E. (2012). Downscaling species occupancy from coarse spatial scales. *Ecological Applications* 22, 1004-1014.

Barwell, L.J., Azaele, S., Kunin, W.E., & Isaac, N.J.B. (2014). Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* 20, 895-907.

**See Also**

See [upgrain](#) for creating objects of class "upgrain". See [downscale](#) and [predict.downscale](#), and [link{hui.downscale}](#) for downscaling models individually.

**Examples**

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
head(atlas.data)

## upgrain data (using All Presences threshold)
occupancy <- upgrain(atlas.data,
                     cell.width = 10,
                     scales = 3,
                     method = "All_Presences")

## ensemble downscaling with an object of class upgrain
ensemble.downscale(occupancies = occupancy,
                   new.areas = c(1, 2, 5, 15, 50, 100, 400, 1600, 6400),
                   cell.width = 10,
                   models = c("Nachman", "PL", "Logis", "GNB", "FNB", "Hui"),
                   plot = TRUE)

## With all models:
## tolerance of the Thomas model has been specified to increase processing time
ensemble.downscale(occupancies = occupancy,
                   new.areas = c(1, 2, 5, 15, 50, 100, 400, 1600, 6400),
                   cell.width = 10,
                   models = "all",
                   tolerance_mod = 1e-3,
                   plot = TRUE)

## Specifying starting parameters for Nachman and GNB models
new.params <- list(Nachman = list("C" = 0.1, "z" = 0.01),
                  GNB = list("C" = 0.1, "z" = 1, "k" = 0.01))

ensemble.downscale(occupancies = occupancy,
                   new.areas = c(1, 2, 5, 15, 50, 100, 400, 1600, 6400),
                   cell.width = 10,
                   models = c("Nachman", "PL", "Logis", "GNB", "FNB", "Hui"),
```

```

starting_params = new.params,
plot = TRUE)

## ensemble modelling with data frame of occupancies (not applicable for Hui
## model) with hypothetical species occupancy data
occupancy.dd <- data.frame(Cell.areas = c(100, 400, 1600, 6400),
                           Occupancy = c(0.16, 0.36, 0.59, 0.86))

## now extent must be specified (but cell.width not needed)
ensemble.downscale(occupancies = occupancy.dd,
                   new.areas = c(1, 2, 5, 15, 50, 100, 400, 1600, 6400),
                   extent = 384000,
                   models = c("Nachman", "PL", "Logis", "GNB", "FNB"),
                   plot = TRUE)

```

---

hui.downscale	<i>Predict occupancy at fine grain sizes using the Hui model</i>
---------------	--

---

## Description

Predict area of occupancy at fine grain sizes from atlas data using the Hui model. There is also a simple plot function. Essentially the function is equivalent to [downscale](#) and [predict.downscale](#) combined.

## Usage

```

hui.downscale(atlas.data, cell.width, new.areas, extent = NULL,
              tolerance = 1e-6, plot = FALSE)

```

## Arguments

atlas.data	either an object of class "upgrain"; or a raster file where 1 = presence and 0 = absence; or a data frame containing these columns: presence = presence - absence data; lon = easting coordinates of cells; lat = northing coordinates of cells.
cell.width	the cell width of the atlas data.
new.areas	vector of grain sizes (in same units as cell.width but squared e.g. km <sup>2</sup> ) at fine scales for model prediction.
extent	the extent of the atlas data in same units as cell.areas. If the input data is of class "upgrain" this can be left as NULL.
tolerance	tolerance for root solving to estimate probability of absence at the fine scale (default = 1e-6).
plot	if plot = TRUE (default = FALSE) plots observed and predicted occupancies against grain size on a log-log plot.

## Details

The function estimates fine-scale occupancy from atlas-scale data using the model developed by Hui. The model requires presence-absence data only at a single scale (the atlas scale) and calculates the probability of occupancy at a fine scale through the conditional probability that at the atlas scale

a randomly chosen cell adjacent to an occupied cell is also occupied. Plotting can be called directly from `hui.downscale` or from `plot.predict.downscale`.

NOTE: if comparing occupancy predictions from the Hui model with those from other models using `downscale` it is important that the atlas data used here is the standardised atlas data whose extent is the same as the largest grain size used in the other models. This ensures that all models are modelling the proportion of occupancy of the same extent. This standardised atlas raster can be obtained through the `upgrain` function (see `vignette("Upgraining", package = "downscale")` for more details.

## Value

Returns an object of class 'predict.downscale' with three objects:

model	Downscaling model used (Hui model in this case).	
predicted	Data frame containing two columns:	
	Cell.area	Grain sizes for which occupancy have been estimated
	Occupancy	Predicted area of occupancy for each grain size
observed	Data frame containing two columns:	
	Cell.area	Grain sizes for which occupancy have been observed
	Occupancy	Observed area of occupancy for each grain size

## Author(s)

Charles Marsh <<charliem2003@gmail.com>> with input from Louise Barwell and Cang Hui.

## References

- Hui, C. (2009). On the scaling patterns of species spatial distribution and association. *Journal of Theoretical Biology* 261, 481-487.
- Hui, C., McGeoch, M.A., & Warren, M. (2006). A spatially explicit approach to estimating species occupancy and spatial correlation. *Journal of Animal Ecology* 7, 140-147.
- Azaele, S., Cornell, S.J., & Kunin, W.E. (2012). Downscaling species occupancy from coarse spatial scales. *Ecological Applications* 22, 1004-1014.
- Barwell, L.J., Azaele, S., Kunin, W.E., & Isaac, N.J.B. (2014). Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* 20, 895-907.

## See Also

See `downscale` for estimating parameters of a downscaling function from observed occupancies at coarse grain sizes using other downscaling models.

See `upgrain` for creating extent-standardised atlas data as an input.

See `ensemble.downscale` for ensemble modelling of multiple downscaling models.

## Examples

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
```

```

atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
head(atlas.data)

## Fit Hui model to atlas.data
(hui <- hui.downscale(atlas.data,
                      cell.width = 10,
                      extent = 228900,
                      new.areas = c(1, 2, 5, 15, 50,75),
                      plot = TRUE))

## Fit Hui model to standardised atlas data for comparison with other models
## First, upgrain data (using All Presences threshold)
occupancy <- upgrain(atlas.data,
                     cell.width = 10,
                     scales = 3,
                     method = "All_Presences")

## the "upgrain" object can be used as input for the Hui model
(hui.stand <- hui.downscale(occupancy,
                           cell.width = 10,
                           new.areas = c(1, 2, 5, 15, 50,75),
                           plot = TRUE))

## compare the area of occupancy (A00) predictions of the two models
hui$predicted
hui.stand$predicted

```

---

plot.predict.downscale

*Plotting of downscaled occupancy at fine grain sizes*

---

## Description

A simple plotting function of predict.downscale objects. Occupancy is plotted against grain size (cell area) in log-log space. Observed occupancy at large grain sizes are plotted in black, and occupancies predicted through [predict.downscale](#) plotted in red.

## Usage

```

## S3 method for class 'predict.downscale'
plot(x, xlim = NULL, ylim = NULL,
     xlab = NULL, ylab = NULL, main = NULL, lwd.obs = NULL, lwd.pred = NULL,
     col.obs = NULL, col.pred = NULL, ...)

```

## Arguments

x	Output object from predict.downscale and of class predict.downscale
xlim, ylim	limits of axes. Defaults to minimum and maximum of data.
xlab, ylab, main	axis labels and title.

```

lwd.obs, lwd.pred      line width of observed and predicted occupancies (default = 2).
col.obs, col.pred      line and point colours of observed (default = black) and predicted (default =
                        red) occupancies.
...                    arguments, including graphical parameters passed to other methods.

```

**Author(s)**

Charles Marsh <<charliem2003@gmail.com>>.

**See Also**

See [predict.downscale](#) and [hui.downscale](#) for generating predict.downscale objects.

**Examples**

```

## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## upgrain data (using All Presences threshold)
occupancy <- upgrain(atlas.data,
                    cell.width = 10,
                    scales = 3,
                    method = "All_Presences")

## Logistic model
logis <- downscale(occupancies = occupancy,
                  model = "Logis")

## predict occupancies at fine scales
logis.pred <- predict(logis,
                    new.areas = c(1, 5, 25, 100, 400, 1600, 6400))

## plot predictions
plot(logis.pred)

## change some of the plotting arguments
plot(logis.pred,
     col.obs = "blue",
     pch = 16,
     ylim = c(0.07, 0.7))

```

---

predict.downscale	<i>Predict occupancy at fine grain sizes</i>
-------------------	--

---

**Description**

Predict proportion of occupancy at fine grain sizes using parameters from an object of class `downscale` estimated from coarse grain sizes using [downscale](#). Proportion of occupancy is converted to area of occupancy (AOO) by multiplying by the extent. There is also a simple plot function.

**Usage**

```
## S3 method for class 'downscale'
predict(object, new.areas, tolerance = 1e-6, plot = FALSE, ...)
```

**Arguments**

object	a fitted object of class 'downscale'.
new.areas	vector of grain sizes (in squared units e.g. km <sup>2</sup> ) for which area of occupancy will be predicted.
tolerance	only applicable for the Thomas model. The tolerance used during integration in the Thomas model during optimisation of parameters. Lower numbers allow for greater accuracy but require longer processing times (default = 1e-6).
plot	if plot = TRUE (default = FALSE) plots observed and predicted occupancies against grain size on a log-log plot using <a href="#">plot.predict.downscale</a> .
...	arguments, including graphical parameters for <a href="#">plot.predict.downscale</a> , passed to other methods.

**Details**

The function takes the parameters for a downscaling model estimated through [downscale](#) and uses the model to predict area of occupancy at finer grain sizes. See [downscale](#) for details on the downscaling models and their parameterisation. Plotting can be called directly from [predict.downscale](#) or from [plot.predict.downscale](#).

For predictions using the Thomas model, if the tolerance value is not sufficiently low it may lead to inaccurate results. Typically, this will be indicated by fine grain sizes with higher predicted occupancies than those at larger grain sizes. In these cases try a lower tolerance value in the arguments.

**Value**

predict returns an object of class 'predict.downscale' with three objects:

model	Downscaling model used.	
predicted	Data frame containing two columns:	
Cell.area	Grain sizes for which occupancy have been estimated	
Occupancy	Predicted proportion of occupancy for each grain size	
A00	Predicted area of occupancy (proportion of area occupancy multiplied by extent)	
observed	Data frame containing two columns:	
	Cell.area	Grain sizes for which occupancy have been observed
	Occupancy	Observed area of occupancy for each grain size

**Author(s)**

Charles Marsh <<charliem2003@gmail.com>> with input from Louise Barwell.

## References

Azaele, S., Cornell, S.J., & Kunin, W.E. (2012). Downscaling species occupancy from coarse spatial scales. *Ecological Applications* 22, 1004-1014.

Barwell, L.J., Azaele, S., Kunin, W.E., & Isaac, N.J.B. (2014). Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* 20, 895-907.

## See Also

See [downscale](#) for estimating parameters of a downscaling function from observed occupancies at coarse grain sizes.

## Examples

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
head(atlas.data)

## explore thresholds using upgrain.threshold
thresh <- upgrain.threshold(atlas.data = atlas.data,
                           cell.width = 10,
                           scales = 3,
                           thresholds = seq(0, 1, 0.01))

## upgrain data (using All Presences threshold)
occupancy <- upgrain(atlas.data,
                    cell.width = 10,
                    scales = 3,
                    method = "All_Presences")

## Logistic model
(logis <- downscale(occupancies = occupancy,
                   model = "Logis"))

## Improved Negative Binomial model
(inb <- downscale(occupancies = occupancy,
                 model = "INB"))

## Specifying the starting parameters (gives a poorer fit in this case)
new.params <- list("C" = 1, "r" = 0.00001, "b" = 0.1)
(inb.new <- downscale(occupancies = occupancy,
                    model = "INB",
                    starting_params = new.params))

## plot the predictions of two FNB models using predict.downscale
predict(inb,
       new.areas = c(1, 2, 5, 25, 100, 400, 1600, 6400),
       plot = TRUE)
predict(inb.new,
       new.areas = c(1, 2, 5, 25, 100, 400, 1600, 6400),
       plot = TRUE)
```

upgrain

*Upgraining of atlas data to larger grain sizes***Description**

Takes presence-absence atlas data and aggregates data to larger grain sizes, returning occupancy at each grain size for use in [downscale](#) modelling. Atlas data may be in the form of raster data or as a data frame with presence-absence information and cell coordinates.

The extent for all scales is standardised to that of the largest grain size by applying a threshold for the proportion of unsampled atlas cells allowed within a cell at the largest grain size. The threshold can be chosen by the user or one of four threshold selections methods apply. See [upgrain.threshold](#) for a visualisation of the thresholds.

The function outputs a data frame of occupancies suitable as input for [downscale](#), and will also plot the original atlas data along with the standardised data for each upgrained scale. In all plots presence = red, absence = white, and NA = grey.

**Usage**

```
upgrain(atlas.data, cell.width = NULL, scales, threshold = NULL,
        method = "Gain_Equals_Loss")
```

**Arguments**

atlas.data	either a raster file of presence-absence atlas data or a data frame of sampled cells. If a data frame columns must be in the following order: longitude, latitude, presence-absence.
cell.width	if data is a data frame, the cell widths of sampled cells. If data is a raster then leave as default (= NULL)
scales	the number of cells to upgrain. Upgraining will happen by factors of 2 - ie if scales = 3, the atlas data will be aggregated in 2x2 cells, 4x4 cells and 8x8 cells.
threshold	default = NULL. A user defined threshold for the proportion of unsampled atlas cells allowed within a cell at the largest grain size. Note: if a method is selected then threshold must be NULL.
method	one of "All_Sampled", "All_Presences", "Gain_Equals_Loss" or "Sampled_Only" (default = "Gain_Equals_Loss"). If the user wishes to define their own threshold then method must equal NULL. See details and <a href="#">upgrain.threshold</a> for descriptions of the different methods.

**Details**

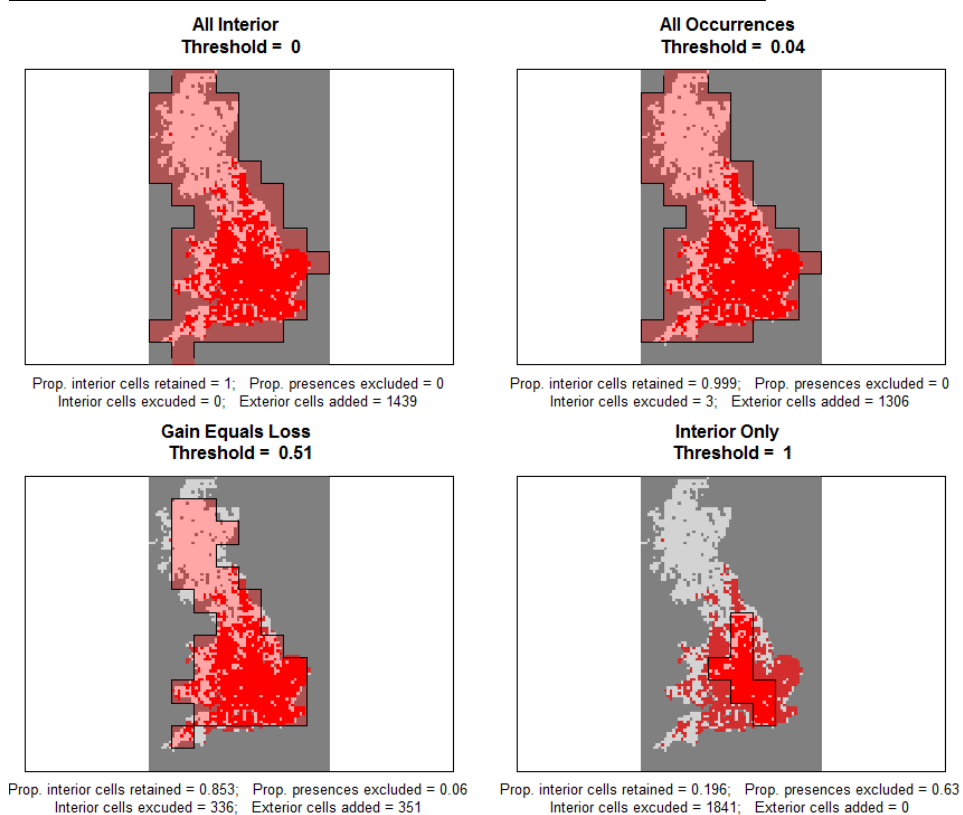
When aggregating data to larger grain sizes, the extent of all grain sizes is set to the extent of the largest grain size. At the atlas scale, unsampled NA cells that fall within this extent are assigned as absences. This ensures that there are subsequently equal extents across all scales and therefore consistency of occupancy estimates. Therefore, it is necessary to apply a threshold whereby only those cells at the largest grain size are retained if a certain proportion of them are sampled at the atlas scale. However, there is a trade-off between assigning unsampled cells as absences, and discarding sampled cells and presences.

It is highly advisable before selecting a threshold to explore this trade-off on a case-by-case basis with [upgrain.threshold](#) and to read the help file therein as well as referring to: `vignette("Upgraining", package = "downscale")`.



The user may input their own threshold or use one of four suggested threshold criteria:

Threshold	Method	Description
0	All_Sampled	All of the original atlas data is included.
Species specific	All_Occurrences	The threshold where no occurrences in the atlas data are excluded.
Atlas specific	Gain_Equals_Loss	The threshold where the number of sampled atlas cells reclassified as No Data equals the number of unsampled exterior cells reclassified as absence. In this threshold the new standardised extent also equals the extent of the original atlas data.
1	Sampled_Only	Only cells that contain 100% sampled atlas data are included.



## Value

Returns a list of class "upgrain" that can be used as a direct input to [downscale](#) and [hui.downscale](#). The list contains five objects:

- `threshold`      The proportion of unsampled atlas cells allowed within a cell at the largest grain size, either defined by the user through threshold or calculated according to method.
- `extent.stand`    The standardised extent after upgraining (equal to the extent of the largest grain size)
- `occupancy.stand`    Occupancy for each grain size where extent has been standardised. The data

frame contains three columns:

Cell.area	Grain sizes for for each upgrained scale.
Extent	Extent for each grain size.
Occupancy	Observed area of occupancy for each grain size.

occupancy.orig Original occupancies for each grain size before extent has been standardised.

The data frame contains three columns:

Cell.area	Grain sizes for for each upgrained scale.
Extent	Extent for each grain size.
Occupancy	Observed area of occupancy for each grain size.

atlas.raster.stand

A raster layer of the extent-standardised atlas data

### Author(s)

Charles Marsh <<charliem2003@gmail.com>>

### Examples

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
head(atlas.data)

## explore thresholds using upgrain.threshold
thresh <- upgrain.threshold(atlas.data = atlas.data,
                           cell.width = 10,
                           scales = 3,
                           thresholds = seq(0, 1, 0.01))

## use a specified threshold - method must equal NULL
upgrain(atlas.data = atlas.data,
        cell.width= 10,
        scales = 3,
        threshold = 0.15,
        method = NULL)

## use one of the suggested methods - threshold must equal NULL
gain_equal_loss <- upgrain(atlas.data = atlas.data,
                          cell.width= 10,
                          scales = 3,
                          threshold = NULL,
                          method = "Gain_Equals_Loss")

## input data for downscale for Gain Equals Loss threshold
gain_equal_loss$occupancy.stand
```

```
## and the original occupancies (note that extent varies with scale)
gain_equal_loss$occupancy.orig
```

---

upgrain.threshold	<i>Exploration of trade-offs in threshold selection for upgraining</i>
-------------------	--

---

## Description

Explores the NoData threshold selection for upgraining whilst keeping a constant extent across scales. The thresholds are the quantity of unsampled cells at the atlas scale allowed within each cell at the largest grain size. A low threshold means that many unsampled cells will be assigned as absences, whereas a high threshold will mean that many sampled cells and many presence records will be excluded. These trade-offs are plotted, and four possible threshold choices are suggested and their maps presented.

## Usage

```
upgrain.threshold(atlas.data, cell.width = NULL, scales,
  thresholds = seq(0, 1, 0.01))
```

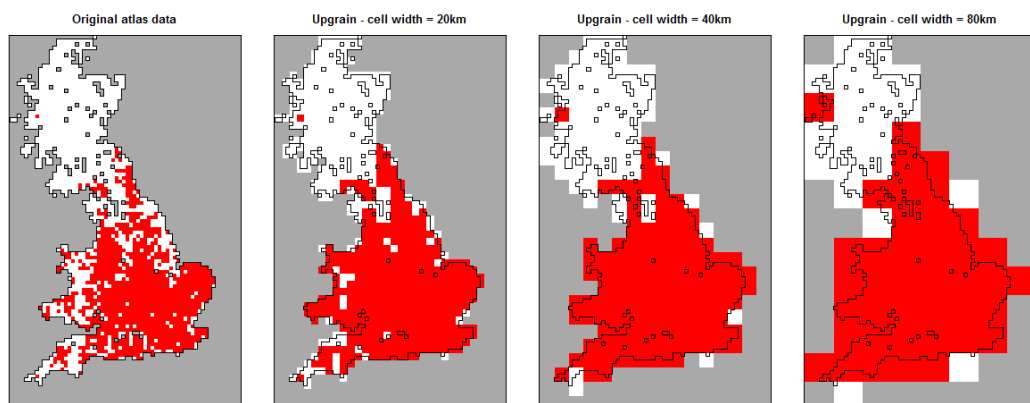
## Arguments

atlas.data	either a raster file of presence-absence atlas data or a data frame of sampled cells. If a data frame columns must be in the following order with these names: lon, lat, presence.
cell.width	if data is a data frame, the cell widths of sampled cells. If data is a raster then leave as default (= NULL)
scales	the number of cells to upgrain. Upgraining will happen by factors of 2 - ie if scales = 3, the atlas data will be aggregated in 2x2 cells, 4x4 cells and 8x8 cells.
thresholds	a vector of thresholds between and including 0 and 1 for the quantity of unsampled NA cells that can be included.

## Details

A more detailed description is available at `vignette("Upgraining", package = "downscale")`.

In order to [downscale](#) we need to [upgrain](#) our atlas data across several scales. However, if the atlas data is not rectangular, as we aggregate cells during upgraining then the extent also increases.



Instead we must ensure the extent is constant across all scales by fixing the extent at all grain sizes to the extent of the largest grain size and convert our proportion of occupied cells back to area of occupancy by using the standardised extent (not the original atlas data extent).

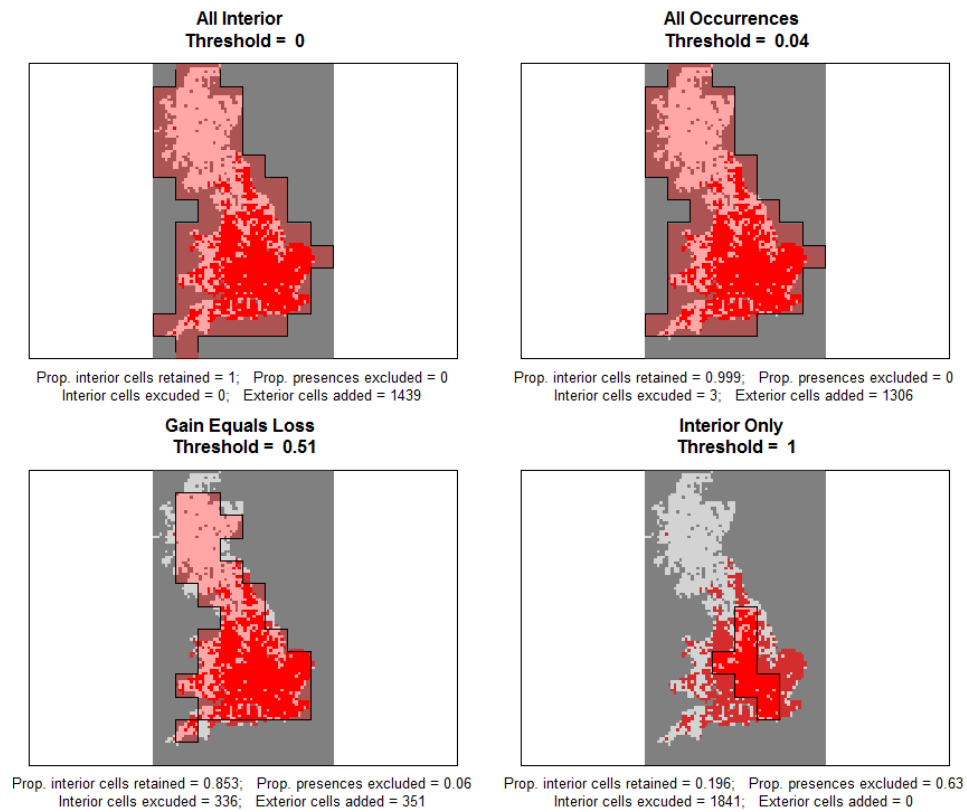
However, if we fix the extent there is trade-off between assigning large areas of unsampled areas as absence, and discarding sampled areas and known presences. The `upgrain.threshold` function allows visualisations of this trade-off at the atlas scale through four plots:

- a) The total standardised extent;
- b) The number of unsampled cells added and assigned as absences, and the number of sampled cells excluded and assigned as No Data;
- c) The proportion of the original atlas data retained;
- d) The proportion of known presences excluded.

The final choice of threshold is up to the user on a case-by-case basis but we propose four threshold criteria in this function:

Threshold	Method	Description
0	All_Sampled	All of the original atlas data is included.
Species specific	All_Occurrences	The threshold where no occurrences in the atlas data are excluded.
Atlas specific	Gain_Equals_Loss	The threshold where the number of sampled atlas cells reclassified as No Data equals the number of unsampled exterior cells reclassified as absence. In this threshold the new standardised extent also equals the extent of the original atlas data.
1	Sampled_Only	Only cells that contain 100% sampled atlas data are included.

The function also creates maps for each of these four thresholds. In the example case this clearly demonstrates the trade-off between generating assumptions about unsampled areas, and losing data (and presences) for the sampled atlas data.



## Value

Returns a list containing two objects:

**Thresholds**            the threshold values for the four default threshold selections.

**Data**                    Data frame containing six columns:

Thresholds	Thresholds tested.
SampledExcluded	Number of sampled cells excluded.
SampledIncluded	Number of sampled cells included.
UnsampledAdded	Number of unsampled NoData cells added.
Extent	Total number of cells included.
PresencesExcluded	Number of cells with presence records excluded.

## Author(s)

Charles Marsh <<charliem2003@gmail.com>>

## Examples

```
## example species data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## if the input data is a data frame it must have the columns "lon", "lat"
## and "presence"
```

```
head(atlas.data)

thresh <- upgrain.threshold(atlas.data = atlas.data,
                           cell.width = 10,
                           scales = 3,
                           thresholds = seq(0, 1, 0.01))

## the four optional thresholds
thresh$Thresholds
head(thresh$Data)
```

# Index

downscale, [2](#), [4](#), [7–11](#), [13–17](#), [19](#)  
downscale-package, [2](#)  
  
ensemble.downscale, [2](#), [7](#), [11](#)  
  
hui.downscale, [2](#), [4](#), [6](#), [8](#), [10](#), [11](#), [13](#), [17](#)  
  
mpfr, [5](#)  
  
plot.predict.downscale, [2](#), [11](#), [12](#), [14](#)  
predict.downscale, [2](#), [4](#), [6](#), [7](#), [9](#), [10](#), [12](#), [13](#),  
[13](#), [14](#)  
  
upgrain, [2](#), [4](#), [6](#), [8](#), [9](#), [11](#), [16](#), [19](#)  
upgrain.threshold, [2](#), [16](#), [19](#)