Package ‘catenelson’

July 2, 2020

Type Package

Title Cate-Nelson Analysis

Version 1.1.1

Description The package allows to perform a Cate-Nelson analysis, developped to partition a set of concentrations of a particular soil nutrient, on the basis of yield. The package extend the function cateNelson from the rcompanion package.

License GPL-3

Copyright Research and development institute for agri-environment (IRDA)

URL https://github.com/irda-rd/catenelson, https://www.irda.qc.ca/en/

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports dplyr,
ggplot2,
gtools,
rcompanion,
testthat,
utils,
methods,
stats

R topics documented:

`catenelson-package` *catenelson: Cate-Nelson Analysis*

Description

The package allows to perform a Cate-Nelson analysis, developped to partition a set of concentrations of a particular soil nutrient, on the basis of yield. The package extend the function `cateNelson` from the `rcompanion` package.

Details

In particular, the function `cateNelson` from the `rcompanion` package has been rewritten to: (1) be quicker, (2) allow to divide data into more than two groups (as proposed in Cate and Nelson (1971)) and (3) accept constraints on the minimal number of different values in each group. (4) accept constraints on the minimum and maximum critical values in `x` and `y`.

The package also corrects for minor inconvenients of the original function such as the inability to handle repeated measures on extremities of the `x` vector and also provides the graph as a `ggplot2` object, which can be further manipulated. Most functionalities and output have been retained from the original function; names of parameters and output might however differ, and now follow the underscore separated naming convention.

The function `cate_nelson` is the wrapper function to be used to perform the analysis. It's documentation provides details on the method and its implementation, as well as an example.

Author(s)

Maintainer: Alexandre Leblanc <`alexandre.leblanc@irda.qc.ca`>

Other contributors:

- Christine Landry <`christine.landry@irda.qc.ca`> [research team head]
- Salvatore Mangiafico (creator of the rcompanion package, from which the analysis approach and some code behaviour were used in this package) [contributor]
- Research and development institute for agri-environment (IRDA) [copyright holder]

References

Cate RB, Nelson LA. 1971. A simple statistical procedure for partitioning soil test correlation data into two classes. Soil. Sci. Soc. Amer. Proc. 35: 658-660.

See Also

Useful links:

- https://github.com/irda-rd/catenelson
- https://www.irda.qc.ca/en/

---

cate_nelson *Cate-Nelson analysis*

Description

Perform a Cate-Nelson analysis, that partition a set of concentrations of a particular soil nutrient (x), on the basis of yield (y). The package documentation (catenelson) describes the aspects that were improved from the function cateNelson of the rcompanion package.

Usage

```
cate_nelson(x, y, label = NULL, n_group, crit_x_index = 1,
  crit_y_index = 1, trend = "positive", min_group_x = 2,
  min_group_y = 1, min_crit_x = NULL, min_crit_y = NULL,
  max_crit_x = NULL, max_crit_y = NULL, details = TRUE,
  details_prop = 1, x_lab = "X", y_lab = "Y", legend = "bottom")

cate_nelson_x(x, y, n_group = 2, min_group = 2, min_crit = NULL,
  max_crit = NULL)

cate_nelson_y(y, group_x, min_group = 1, min_crit = NULL,
  max_crit = NULL, trend = "positive")
```

Arguments

x	numeric vector of a predictor variable (e.g. nutrient concentration).
y	numeric vector of the associated response variable (e.g. yield, relative yield, ...).
label	character characterizing the point (e.g. the site names where the sample were collected). It serves to automatically set color of points on the graph; if label = NULL (by default), black and white are used.
n_group	integer, the number of groups in which to partition data in x and in y (possible values: between 2 and 10).
crit_x_index, crit_y_index	integer, the index of the partition to select. The default value (crit_x_index = 1 or crit_y_index = 1) correspond to the best partition, see the details section.
trend	character, the expected trend of the data, either positive or negative.
min_group_x, min_group_y	integer, the minimum number of different values in each group for the x and y partitioning; min_group_x must be at least two, min_group_y can be one.

min_crit_x, min_crit_y, max_crit_x, max_crit_y
: numeric vectors of length n_group-1, optional inequality constraints on each critical values in x and y (min <= x <= max). The default value is NULL and represent no constraint.

details
: logical indicating if details about partitioning in x and y should be provided in the output.

details_prop
: numeric, indicating the proportion of the total partition to show in the output (between 0 and 1), used if details = TRUE.

x_lab, y_lab, legend
: character specifying graphical options, repectively: the name of the x-axis, the name of the y-axis and the position of the legend (none to remove, look into ggplot2 options).

Details

The analysis is divided in two parts: the partitioning in x, followed by the partitioning in y. Partitioning in x follows the procedure described in Cate and Nelson (1971), in which:

1. Data is sorted according to x.
2. Data is partitioned into groups of contiguous points in x. At least two points in each groups must be chosen, duplicated data in x must not be separated and count as one value. Associated critical values in x are computed from the mid values between the two adjacent points belonging to different groups (not in the original paper).
3. The partition that maximize R2 is selected. The model would correspond to a step function with the average value for each group.

Partitioning in y was mentioned as part of an earlier graphical method in the original paper of Cate and Nelson (1971), but no algorithm was suggested. The function reproduces the approach used in the rcompanion package, in which:

1. Data is sorted according to y.
2. Given the partition in x, the partition in y that maximise the number of point on diagonal quadrats, either positive or negative depending on the data trend, is selected.
3. Cramer's V and Ficher's p values associated with the contingency table formed by the quadrats are also calculated as supplementary information. The actual package sort partition in decreasing order of Cramer's V for a same number of point in the diagonal quadrats.

As allowed in the rcompanion package, one can select the ith best partition in x (crit_x_index) and in y (crit_y_index). In both partitioning, one can impose additional constraint on the minimum number of distinct values per group through min_group_x and min_group_y. Additional constraints can also be defined on the range of critical values in x or y through the parameters min_crit_x, min_crit_y, max_crit_x, max_crit_y.

Quadrat names are defined as ij with i the index along the x-axis and j the index along the y-axis. The number of groups has been restricted to a maximum of 10 to avoid ambiguities with the quadrat names, and its repercussion when counting the number of points. The code might be improve in the future to allow a greater number of groups, but the constraint is not likely to be reached because covering all possible partitions would then require a considerable time to compute.

The function cate_nelson is a wrapper around the function cate_nelson_x and cate_nelson_y, which perform the partitioning respectively in x and y.

Value

The function return a `list` with the following elements:

- `x_partition` the partitions in `x` and their statistics (`data.frame`), sorted by R2, in proportion defined by `details_prop`.
- `y_partition` the partitions in `y` and their statistics (`data.frame`), sorted by `p_pred` then `cramer_V`, in proportion defined by `details_prop`.
- `model` the selected partitions in x and y and their statistics (`data.frame`).
- `group` the group in x and y, to which each point belong according to the `model`'s partition (`data.frame`).
- `graph` a `ggplot2` object representing the points and the quadrats, delimited by the critical values of the `model`. Empty circles correspond to points falling outside the diagonal quadrats, while full circles represent points within. Color can be added through `label`.
- `param` a `list` of the entry parameters that characterize the analysis.

References

Cate RB, Nelson LA. 1971. A simple statistical procedure for partitioning soil test correlation data into two classes. Soil. Sci. Soc. Amer. Proc. 35: 658-660.

Examples

```
#Generate data
n = 30
x <- rnorm(n-2)
x <- c(x, rep(min(x), 2))
y <- x + rnorm(n)
label <- LETTERS[(seq_len(n)-1)%%4+1] #Alternative for black and white: label = NULL

#Call the function
CN <- cate_nelson(x, y, label = label, n_group = 3, trend = "positive")

#Investigate the output
CN$x_partition
CN$y_partition
CN$model
CN$group
CN$graph
CN$param
```

`cate_nelson_graph` *cate_nelson_graph*

Description

Produce a graph associated to the Cate-Nelson analysis.

Usage

```
cate_nelson_graph(x, y, df_x, df_y, quadrat, quadrat_name_pred,
  quadrat_name_err, x_lab = "X", y_lab = "Y", label = NULL,
  legend = "bottom")
```

Arguments

x	numeric vector of a predictor variable (e.g. nutrient concentration).
y	numeric vector of the associated response variable (e.g. yield, relative yield, ...).
df_x	data.frame corresponding to the df element of an object produced by cate_nelson_x selected for a single partition in x.
df_y	data.frame corresponding to the df element of an object produced by cate_nelson_y selected for a single partition in y.
quadrat	character, vector of the same length as x and y defining the quadrats in which points belong.
quadrat_name_pred	character, name of quadrats that correspond to the diagonal specified by trend.
quadrat_name_err	character, name of quadrats that does not correspond to the diagonal specified by trend.
x_lab, y_lab, legend	character specifying graphical options, repectively the name of the x-axis, the name of the y-axis and the position of the legend (none to remove, look into ggplot2 options).
label	character characterizing the point (e.g. the site where the sample was collected). It serves to automatically set color of points on the produced graph; if label = NULL (by default), black and white are used.

Value

Return a ggplot2 object representing points and the quadrats, delimited by the critical values of the model. Empty circles correspond to point falling outside the diagonal quadrats, while full circles represent point within. Color can be added through label.

---

cn_rss *Residual sum of squares (grouped data)*

Description

Compute the residual sum of squares for grouped data (i.e. the within group ss). The groups are defined by selection matrices.

Usage

```
cn_rss(selection_matrices, y)
```

Arguments

selection_matrices	list of matrix which elements, named after the group, represent if the element belong to the group (1) or not (0). The rows of the matrices are associated to elements of y; while column represent different partitions.
y	numeric, vector on which calculate the sum of squares

Details

The residual sum of squares is computed through matrix multiplication, which considerably fasten its calculation when the number of partition is high. This is an alternative to looping on regression and anova for each partition.

Value

Return a (`numeric`) vector of residual sum of squares of length corresponding to the number of partition (possible groupings), determined by the `selection_matrices`.

Examples

```
group <- matrix(c(rep(1L,4), rep(2L,6), rep(1L,6), rep(2L,4)), ncol = 2)
S <- group_selection_matrices(group)
y <- rnorm(10)
cn_rss(S, y)
```

---

cn_rss_group *Residual sum of squares (grouped data, one group)*

Description

Compute the residual sum of squares for grouped data (i.e. the within group ss) for one group only. Group belonging is defined by a selection matrix.

Usage

```
cn_rss_group(selection_matrix, y)
```

Arguments

`selection_matrix`
: `matrix` which elements represent if the element belong to the group (1) or not (0). The rows of the matrices are associated to elements of `y`; while column represent different partitions.

`y`
: `numeric`, vector on which calculate the sum of squares

Details

The residual sum of squares is computed through matrix multiplication, which considerably fasten its calculation when the number of partition is high. This is an alternative to looping on regression and anova for each partition.

Value

Return a (`numeric`) vector of residual sum of squares of length corresponding to the number of partition, determined by the `selection_matrices`.

Examples

```
group <- matrix(c(rep(1L,4), rep(2L,6), rep(1L,6), rep(2L,4)), ncol = 2)
S <- group_selection_matrices(group)
y <- rnorm(10)
cn_rss(S, y)
```

---

cn_tss *Total sum of squares*

Description

Compute the total sum of squares.

Usage

```
cn_tss(y)
```

Arguments

`y`	`numeric` vector on which to calculate the sum of squares.

Details

Computed through matrix multiplication.

Value

Return the sum of squares (`numerical`).

Examples

```
y <- rnorm(10)
cn_tss(y)
```

---

group_crit *Critical values*

Description

Generate critical values associated with divisions.

Usage

```
group_crit(x, division)
```

Arguments

`x`	`numeric`, an ordered vector of elements.
`division`	`matrix` of position that define a new group (rows) for different partitions (columns), as generated by the function `group_division`.

Value

Return a `matrix` of the same dimension as `division`, containing the average values of `x` for the position defined in the `division` and the position preceeding it.

Examples

```
#General example
x <- sample(1:100, 10)
division <- group_division(x, n_group = 3, min_group = 2)
group_crit(x, division)
```

`group_division` *Group division*

Description

Find the possible ways to separate an ordered numerical vector in groups.

Usage

```
group_division(x, n_group, min_group, min_crit = NULL, max_crit = NULL)
```

Arguments

`x` — `numeric`, an ordered vector of elements.

`n_group` — `integer`, the number of groups (minimum 2).

`min_group` — `integer`, the minimum number of values in each group (not elements).

`min_crit, max_crit` — `numeric` vectors of length corresponding to the number of group (number of rows of `division`) minus one, representing inequality constraints on each critical values in `x` (`min <= x <= max`). The default value is `NULL` and represent no constraint.

Details

Repeated values are kept in the same group and only count as one value with respect to the `min_group` constraint. An error is thrown if no group division meet the conditions imposed.

Value

Return a `matrix` of position that define new groups (divisions). Rows represent the `n_group -1` divisions for a given partition, while columns represent different partitions.

Examples

```
#General example
x <- sample(1:100, 10)
group_division(x, n_group = 2, min_group = 2)

#Example with a repeated value
x <- 100*c(1:3, 3, 3:5)
group_division(x, n_group = 2, min_group = 2)
```

group_division_bound *Group division bound*

Description

Evaluate column index of a division object that respect bounding restriction on their corresponding critical values.

Usage

```
group_division_bound(x, division, min_crit = NULL, max_crit = NULL)
```

Arguments

x	`numeric`, an ordered vector of elements.
division	`matrix` of position that define a new group (rows) for different partitions (columns), as generated by the function group_division; alternatively a `numeric` vector representing a single partition.
min_crit, max_crit	`numeric` vectors of length corresponding to the number of group (number of rows of `division`) minus one, representing inequality constraints on each critical values in `x` (`min <= x <= max`). The default value is `NULL` and represent no constraint.

Value

Return a `logical` vector indication which column of `division` that respect the constraints.

Examples

```
#General example
x <- c(9,11, sample(12:19, 3), 20,22)
division <- group_division(x, n_group = 3, min_group = 1)
group_division_bound(x, division, min_crit = c(11,11), max_crit = c(20,20))
```

group_division_check *Group division check*

Description

Check that a division vector or matrix is appropriate.

Usage

```
group_division_check(n_row, division)
```

Arguments

n_row — integer, the number of row of the matrix to build, which should equal the length of the vector to be classified.

division — matrix of position that define a new group (rows) for different partitions (columns), as generated by the function group_division; alternatively a numeric vector representing a single partition.

Value

Return TRUE if properties of division are appropriate and an error otherwise.

Examples

```
#General example
##Data
x <- sample(1:100, 10)
division <- group_division(x, n_group = 2, min_group = 2)

##Case that return TRUE, select n_row = 5 for an example that return an error.
group_division_check(n_row = length(x), division)
group_division_check(n_row = length(x), division[,1])
```

---

group_matrix *Group matrix*

Description

Generate a matrix of group, from a division matrix.

Usage

```
group_matrix(n_row, division)
```

Arguments

n_row — integer, the number of row of the matrix to build, which should equal the length of the vector to be classified.

division — matrix of position that define a new group (rows) for different partitions (columns), as generated by the function group_division.

Details

Division is checked for validity before generating matrix.

Value

Return a matrix of integer representing the group to which elements of a vector might belong.

Examples

```
#General example
x <- sample(1:100, 10)
division <- group_division(x, n_group = 2, min_group = 2)
group_matrix(n_row = length(x), division)
```

---

group_nb *Number of group*

Description

Provides the number of group in a group matrix.

Usage

```
group_nb(group)
```

Arguments

group	`matrix` of `integer` indicating the group elements belong (rows) for various partition (columns), as generated by the function `group_matrix`.

Examples

```
group <- t(gtools::permutations(3,3,1:3))
group_nb(group)
```

---

group_selection_matrices *Selection matrices*

Description

Generate a list of selection matrices from a group matrix.

Usage

```
group_selection_matrices(group)
```

Arguments

group	`matrix` of `integer` indicating the group elements belong (rows) for various partition (columns), as generated by the function `group_matrix`.

Details

The groups in the `group` matrix must be integers, incremented from 1.

Value

Return a list of matrix which elements, named after the group, represent if the element belong to the group (1) or not (0).

Examples

```
group <- t(gtools::permutations(3,3,1:3))
group_selection_matrices(group)
```

`group_vector` *Group matrix*

Description

Generate a vector of group, from a division vector.

Usage

```
group_vector(n_row, division_vector)
```

Arguments

`n_row` `integer`, the number of row of the matrix to build, which should equal the length of the vector to be classified.

`division_vector` `integer`, vector of position that define a new group (row).

Details

Division is checked for validity before generating the vector.

Value

Return a vector of `integer` representing the group to which elements of a vector might belong.

Examples

```
#General example
x <- sample(1:100, 10)
division <- group_division(x, n_group = 2, min_group = 2)
group_vector(n_row = length(x), division[,1])
```

quadrat_contingency *Quadrat contingency*

Description

Form a contingency table (matrix) from quadrat counts.

Usage

```
quadrat_contingency(Q)
```

Arguments

Q	numeric vector corresponding to counts of quadrats.

Details

The order of elements in Q are assumed to increment by i first, then j (e.g. 11, 12, 12, 21, 22, ...), with i designating quadrat indexes along the the x-axis and j the y-axis. As a supplementary constraint, i and j must range over the same values and therefore Q must have a square number of elements (quadrats).

Value

Return the corresponding contingency matrix.

Examples

```
Q <- rnorm(9)
quadrat_contingency(Q)
```

quadrat_count *Quadrat count*

Description

Count the number of points that is present in quadrat defined by group_x in x and group_y in y.

Usage

```
quadrat_count(group_x, group_y)
```

Arguments

group_x	group matrix containing only one partition (column).
group_y	group matrix containing one or multiple partitions (column).

Value

Return a `data.frame` with columns named `ij`, after the quadrat, with i the quadrat index on the x-axis and j the quadrat index on the y-axis. Each line correspond to a partition of `group_y`.

Examples

```
#Generate data
##Data
n = 10
x <- rnorm(n)
y <- rnorm(n)
##Group x, only one partition
division_x  <- group_division(y, 2, min_group = 2)
group_x <- group_matrix(n, division_x[,1 , drop = FALSE])

##Group y, multiple partition
division  <- group_division(y, 2, min_group = 2)
group_y <- group_matrix(n, division)

#Compute the number of elements in each quadrat
quadrat_count(group_x, group_y)
```

`quadrat_count_pred` *Quadrat count prediction*

Description

Count the number of points in quadrats on the diagonal specified by `trend`, from existing an data count.

Usage

```
quadrat_count_pred(Q, trend)
```

Arguments

`Q`	`data.frame` of quadrat counts, as generated by the function `quadrat_count`. Must possesses a squared number of elements.
`trend`	`character`, either `positive` or `negative`.

Value

Return a `data.frame` with columns named `ij`, after the quadrat, with i the quadrat index on the x-axis and j the quadrat index on the y-axis. Each line correspond to a partition of `group_y`.

Examples

```
#Generate data
##Data
n = 10
x <- rnorm(n)
y <- rnorm(n)
##Group x, only one partition
division_x  <- group_division(y, 2, min_group = 2)
group_x <- group_matrix(n, division_x[,1 , drop = FALSE])

##Group y, multiple partition
division  <- group_division(y, 2, min_group = 2)
group_y <- group_matrix(n, division)

#Compute the number of elements in each quadrat
Q <- quadrat_count(group_x, group_y)

#Compute the number of element on the positive diagonal
quadrat_count_pred(Q = Q, trend = "positive")
```

quadrat_name *Quadrat name*

Description

Generate names of quadrats.

Usage

```
quadrat_name(n_group)
```

Arguments

n_group	`integer`, the number of groups.

Details

The order of elements are incremented by `i` first, then `j` (e.g. 11, 12, 12, 21, 22, ...), with i designating quadrat indexes along the the x-axis and j the y-axis. As a supplementary constraint, i and j range from 1 to the `n_group`, representing a squared number of quadrats. Names are non ambiguous for n_group smaller or equal to 10

Value

Return a vector of name for the quadrats (`character`).

Examples

```
n_group = 10
quadrat_name(n_group)
```

quadrat_name_err *Quadrat name error*

Description

Generate name of quadrats that does not correspond to the diagonal specified by `trend`. Complement of the function `quadrat_name_pred`.

Usage

```
quadrat_name_err(n_group, trend)
```

Arguments

n_group	`integer`, the number of groups.
trend	`character`, either `positive` or `negative`.

Value

Return a vector of quadrat names.

Examples

```
quadrat_name_pred(n_group = 3, trend = "positive")
quadrat_name_err(n_group = 3, trend = "positive")
```

quadrat_name_pred *Quadrat name prediction*

Description

Generate name of quadrats that correspond to the diagonal specified by `trend`.

Usage

```
quadrat_name_pred(n_group, trend)
```

Arguments

n_group	`integer`, the number of groups.
trend	`character`, either `positive` or `negative`.

Value

Return a vector of quadrat names.

Examples

```
quadrat_name_pred(n_group = 3, trend = "positive")
quadrat_name_pred(n_group = 3, trend = "negative")
```

quadrat_stat *Quadrat statistics*

Description

Compute the Cramer's V and Fisher p values of the contingency matrix made by the quadrats.

Usage

```
quadrat_stat(Q)
```

Arguments

Q	`numeric` vector corresponding to counts of quadrats.

Details

The order of elements in `Q` are assumed to increment by `i` first, then `j` (e.g. 11, 12, 12, 21, 22, ...), with i designating quadrat indexes along the the x-axis and j the y-axis. As a supplementary constraint, i and j must range over the same values and therefore Q must have a square number of elements (quadrats).

Examples

```
#Generate data
##Data
n = 10
x <- rnorm(n)
y <- rnorm(n)
##Group x, only one partition
division_x  <- group_division(y, 2, min_group = 2)
group_x <- group_matrix(n, division_x[,1 , drop = FALSE])

##Group y, multiple partition
division  <- group_division(y, 2, min_group = 2)
group_y <- group_matrix(n, division)

#Compute the number of elements in each quadrat
Q <- quadrat_count(group_x, group_y)
quadrat_stat(Q)
```

Index