

Package ‘stepseg’

May 1, 2023

Title Stepwise Segmented Regression Analysis

Version 0.9.1

Description This package facilitates stepwise segmented regression analysis, an approach for breakpoint detection and evaluation that is especially useful for data with high error variance (e.g., longitudinal behavioral data). If you use this package, please cite the original publication: Britt, B. C. (2015). Stepwise segmented regression analysis: An iterative statistical algorithm to detect and quantify evolutionary and revolutionary transformations in longitudinal data. In S. A. Matei, M. G. Russell, & E. Bertino (Eds.), Transparency in social media: Tools, methods, and algorithms for mediating online interactions (pp. 125-144). Heidelberg, Germany: Springer.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Imports Compositional,
DirichletReg,
Formula,
maxLik,
stats,
utils

R topics documented:

| | |
|--------------------------------|----|
| DirichReg | 2 |
| get_p_from_model | 3 |
| get_t_from_model | 3 |
| print.stepseg_output | 4 |
| stepseg | 4 |
| stepsegcomp | 10 |
| stepsegdir | 15 |

| | |
|--------------|-----------|
| Index | 20 |
|--------------|-----------|

Description

This is a modified form of the [DirichReg](#) function, adjusted to avoid "model frame and formula mismatch in model.matrix()" errors that frequently result when the formula is long and the original function is called from within another function.

Usage

```
DirichReg(  
  formula,  
  data,  
  model = c("common", "alternative"),  
  subset,  
  sub.comp,  
  base,  
  weights,  
  control,  
  verbosity = 0  
)
```

Arguments

| | |
|-----------|--|
| formula | The formula specifying the regression model |
| data | The data set from which the model is constructed |
| model | The parameterization of the model |
| subset | Not used |
| sub.comp | Not used |
| base | Not used |
| weights | Not used |
| control | A list of variables to control the convergence process |
| verbosity | Not used |

Value

A Dirichlet regression model as specified by [DirichReg](#)

| | |
|------------------|------------------------------|
| get_p_from_model | Retrieve Smallest p -value |
|------------------|------------------------------|

Description

This function obtains the smallest p -value among all p -value for a given breakpoint in a regression model prepared by prepared by [comp.reg](#).

Usage

```
get_p_from_model(  
  model,  
  ord,  
  breakpoints,  
  i,  
  allow_interactions,  
  ivs,  
  parameterization  
)
```

Arguments

| | |
|--------------------|---|
| model | The regression model outputted from DirichReg |
| ord | A non-negative numeric atomic vector indicating the maximum exponent that was applied to coefficients added to the regression model |
| breakpoints | A vector of all breakpoints used in model |
| i | The numeric index of the breakpoint for which to obtain the t -statistic |
| allow_interactions | A boolean value indicating whether interactions were created between independent variables |
| ivs | A data.frame representing values of the independent variables |
| parameterization | The specific parameterization used to create model |

Value

A numeric value representing the smallest p -value corresponding to the specified breakpoint

| | |
|------------------|---------------------------------|
| get_t_from_model | Retrieve Largest t -statistic |
|------------------|---------------------------------|

Description

This function obtains the largest absolute value among all t -statistics for a given breakpoint in a regression model summary prepared by [comp.reg](#).

Usage

```
get_t_from_model(model, ord, i, allow_interactions, ivs)
```

Arguments

| | |
|---------------------------------|---|
| <code>model</code> | The regression model outputted from comp.reg |
| <code>ord</code> | A non-negative numeric atomic vector indicating the maximum exponent that was applied to coefficients added to the regression model |
| <code>i</code> | The numeric index of the breakpoint for which to obtain the t -statistic |
| <code>allow_interactions</code> | A boolean value indicating whether interactions were created between independent variables |
| <code>ivs</code> | A <code>data.frame</code> representing values of the independent variables |

Value

A numeric value representing the largest absolute value among all t -statistics corresponding to the specified breakpoint

```
print.stepseg_output
```

Print

Description

This method prints the output element of an object with the `stepseg_output` class, such as output obtained from the [stepseg](#) function.

Usage

```
## S3 method for class 'stepseg_output'
print(x, ...)
```

Arguments

| | |
|------------------|--|
| <code>x</code> | An object with the <code>stepseg_output</code> class |
| <code>...</code> | Other arguments inherited from the generic <code>print</code> function |

```
stepseg
```

Stepwise Segmented Regression Analysis

Description

This function performs stepwise segmented regression analysis, as described by Britt (2015).

Usage

```
stepseg(
  dv,
  ivs = data.frame(1:nrow(data.frame(dv))),
  start_formula = NULL,
  add = 0.15,
  remove = 0.2,
  add_mode = "mse",
  family = stats::gaussian(link = "identity"),
  order = 1,
  allow_interactions = FALSE,
  update = 0,
  verbose = FALSE
)
```

Arguments

| | |
|---------------------------------|---|
| <code>dv</code> | A vector or data.frame representing values of the dependent variable, which is coerced to a data.frame if not provided as one |
| <code>ivs</code> | A data.frame representing values of the independent variables, which is coerced to a data.frame if not provided as one (default = <code>c(1:nrow(dv))</code>), which implicitly treats <code>dv</code> as sequentially ordered data and attempts to detect breakpoints in that sequence) |
| <code>start_formula</code> | A formula to be used for the regression model prior to the first stepwise model selection iteration, with all listed coefficients permanently retained in the model (default = <code>NULL</code> , which adds an intercept and a linear coefficient for each independent variable to the regression model) |
| <code>add</code> | A numeric atomic vector between 0 and 1 indicating the p -value threshold to add coefficients to the model during each forward selection iteration, which must be less than or equal to the <code>remove</code> argument in order to avoid infinite loops (default = <code>.15</code> , as recommended by Britt, 2015) |
| <code>remove</code> | A numeric atomic vector between 0 and 1 indicating the p -value threshold to remove coefficients from the model during each backward selection iteration, which must be greater than or equal to the <code>add</code> argument in order to avoid infinite loops (default = <code>.20</code> , as recommended by Britt, 2015) |
| <code>add_mode</code> | A character atomic vector (either "p" or "mse") indicating what criterion should be used to select the best candidate block during each forward selection iteration (default = "mse") |
| <code>family</code> | A family object specifying the distribution and link function to be used in the general linear model (default = <code>stats::gaussian(link = "identity")</code> , which is appropriate for normally-distributed data and which facilitates a simple or multiple linear regression) |
| <code>order</code> | A non-negative numeric atomic vector indicating the maximum exponent that will be applied to coefficients added to the regression model (default = 1; it is generally recommended to use either <code>order = 0</code> or <code>order = 1</code>) |
| <code>allow_interactions</code> | A boolean value indicating whether interactions may be created between indicator functions representing breakpoints along one independent variable with other independent variables in the model (default = <code>FALSE</code> , which restricts each indicator function to only interact with the independent variable along which its breakpoint was defined) |

| | |
|---------|--|
| update | A numeric value indicating how many loop iterations should elapse between progress updates (default = 0, which suppresses output) |
| verbose | A boolean value indicating whether the current regression model and results should be outputted as part of the progress report every update iterations (default = FALSE; note that if update = 0, no updates will be printed even if verbose = TRUE) |

Details

When using stepwise segmented regression analysis, coefficients are added to and removed from the model in "blocks," with a given block consisting of the indicator function representing a given breakpoint location as well as all interaction terms between that indicator function and independent variables that are eligible to be added to the model.

The order and allow_interactions arguments jointly indicate what interaction terms are valid. order indicates the maximum exponent that can be applied to the independent variables, while allow_interactions indicates whether interaction terms can be created between a given independent variable and an indicator function that is defined by values of a different independent variable.

All interaction terms that are considered valid, with exponents ranging from 0 to order, are included in a given block. Consider, for instance, an analysis in which ivs, the data.frame representing the independent variables, has three columns signifying three variables. If order = 1 and allow_interactions = FALSE, which are their default values in stepseg, then the block corresponding to a potential breakpoint at `ivs[, 1] = 10` would include two coefficients:

- `I(ivs[, 1] = 10)`
- `I(ivs[, 1] = 10):(ivs[, 1]^1)`

Setting order = 2 would retain both of the aforementioned coefficients and also add

- `I(ivs[, 1] = 10):(ivs[, 1]^2)`

to the block. Maintaining order = 2 and setting allow_interactions = TRUE would result in the block containing a total of seven coefficients:

- `ivs[, 1] = 10`
- `I(ivs[, 1] = 10):(ivs[, 1]^1)`
- `I(ivs[, 1] = 10):(ivs[, 1]^2)`
- `I(ivs[, 1] = 10):(ivs[, 2]^1)`
- `I(ivs[, 1] = 10):(ivs[, 2]^2)`
- `I(ivs[, 1] = 10):(ivs[, 3]^1)`
- `I(ivs[, 1] = 10):(ivs[, 3]^2)`

Note that if order = 0, then each block will only include the indicator function itself (in this example, `I(ivs[, 1] = 10)`) regardless of the value of allow_interactions.

During each forward selection iteration, the algorithm selects the "best" block of coefficients that can be added. If add_mode = "p", then all possible candidate blocks that could be added to the model are evaluated, and whichever block has a *p*-value less than or equal to all *p*-values in all other blocks is treated as the "best" block. As long as that *p*-value is less than the threshold specified by the add argument, all coefficients included in the block are added to the model. If add_mode = "mse", then the candidate block whose coefficients would jointly reduce the model mean squared error by the greatest amount is treated as the best block, and all of its coefficients are added to the model if the resulting *p*-value for at least one of those coefficients would be less than add.

During each backward selection iteration, all coefficients listed in `start_formula` are retained in the regression model. Among all remaining coefficients, any blocks whose p -values are all greater than or equal to `remove` are removed from the model. If at least one p -value contained in a given block is less than `remove`, then none of the coefficients in the block are removed from the model.

When `order = 0`, `add_mode = "p"` is generally acceptable to follow common conventions of stepwise model selection. When `order > 0`, however, `add_mode = "p"` is more likely to result in spurious breakpoints being added to the model due to intercept and higher-order terms competing with one another, and in rare cases the algorithm may entirely fail to converge. `add_mode = "mse"` is more robust against these issues and is strongly recommended whenever `order > 0`.

Note that this function uses Type II sums of squares (via [drop1](#)) for parameter estimation, as Type I sums of squares (the default in [glm](#)) can result in an infinite loop. If Type I sums of squares for the final model are desired, the corresponding `glm` object can be retrieved via the `raw_glm` element of the output from `stepseg`.

The `start_formula` argument can be used to provide a regression model whose coefficients will be inputted prior to the first stepwise model selection iteration. However, only first-order independent variables should be listed in this formula. Higher-order terms are not outputted in a predictable manner by some of the external functions used called by `stepseg` and may lead to invalid results. If an interaction term is desired, you should create a new column in the data.frame submitted as the `ivs` argument to represent that interaction, and that column can subsequently be used in the analysis. Similarly, listing indicator functions representing breakpoints in this argument may lead to unexpected behavior during backward selection iterations of the `stepseg` function, so forcibly adding them to the model via the `start_formula` argument is not advised. For most use cases, this argument can and should be omitted.

The larger the value of `order`, the more likely spurious breakpoints are to emerge. As with other regression contexts, you should only increase the complexity of your model (such as by increasing the value of `order`) when you have a clear reason to do so. Moreover, whenever `order > 0`, singularities may occur that render some coefficients inestimable. For instance, if `order = 3` and a pair of breakpoints is identified along the same independent variable, with those two breakpoints occurring two data points apart, there will be insufficient data between the breakpoints to estimate the standard error of the interaction between the indicator function and a cubic term, so it will be reported as NA. This is normal, expected behavior in stepwise segmented regression analysis. In such cases, those NA coefficients can be treated as though they were absent from the model. The `cleaned_output`, `cleaned_formula`, and `cleaned_factors` elements of the output from `stepseg` remove any NA coefficients accordingly, while `cleaned_raw_glm` uses `cleaned_formula` in a [glm](#) function call that, like `raw_glm`, uses Type I sums of squares.

However, if `family = "binomial"` or `family = "poisson"`, any singularities will generate an error. This is due to a known limitation in [drop1](#), which `stepseg` calls to generate statistics using Type II sums of squares for general linear models. A warning is generated when either of these two distribution families are used as input arguments.

Finally, use caution when setting `order > 1`, as increasingly high-order terms may compete with lower-order terms in the model and yield unpredictable results. Refer to Britt (2015) for more information.

Value

An object with the `stepseg_output` class that contains the following list elements:

`output`: The summary of the final regression model that resulted from the stepwise segmented regression analysis, constructed using [glm](#) and passed through [drop1](#)) to obtain results using Type II sums of squares

`raw_glm`: The raw output for the final regression model using [glm](#), without passing it through [drop1](#), thereby using Type I sums of squares rather than Type II

`cleaned_output`: The summary of the final regression model that resulted from the stepwise segmented regression analysis, removing any coefficients that had NA p -values due to singularities or other issues, constructed using `glm` and passed through `drop1`) to obtain results using Type II sums of squares

`cleaned_raw_glm`: The raw output for the final regression model using `glm`, removing any coefficients that had NA p -values due to singularities or other issues, but without passing it through `drop1`, thereby using Type I sums of squares rather than Type II

`start_formula`: The formula for the regression model prior to the first iteration of the stepwise segmented regression analysis

`final_formula`: The final formula for the regression model that resulted from the stepwise segmented regression analysis

`cleaned_formula`: The final formula for the regression model that resulted from the stepwise segmented regression analysis, removing any coefficients that had NA p -values due to singularities or other issues

`start_factors`: The coefficients used in the regression model prior to the first iteration of the stepwise segmented regression analysis

`final_factors`: The coefficients used in the regression model that resulted from the stepwise segmented regression analysis

`cleaned_factors`: The coefficients used in the regression model that resulted from the stepwise segmented regression analysis, removing any coefficients that had NA p -values due to singularities or other issues

`add_iterations`: The number of forward selection iterations elapsed

`remove_iterations`: The number of backward selection iterations elapsed

`model_iterations`: A list showing how the regression model appeared at the end of each forward selection and backward selection iteration

`dv`: The value set for the `dv` argument

`ivs`: The value set for the `ivs` argument

`add`: The value set for the `add` argument

`remove`: The value set for the `remove` argument

`add_mode`: The value set for the `add_mode` argument

`family`: The value set for the `family` argument

`order`: The value set for the `order` argument

`allow_interactions`: The value set for the `allow_interactions` argument

`update`: The value set for the `update` argument

`verbose`: The value set for the `verbose` argument

References

Britt, B. C. (2015). Stepwise segmented regression analysis: An iterative statistical algorithm to detect and quantify evolutionary and revolutionary transformations in longitudinal data. In S. A. Matei, M. G. Russell, & E. Bertino (Eds.), *Transparency in social media: Tools, methods, and algorithms for mediating online interactions* (pp. 125-144). Heidelberg, Germany: Springer.

Examples

```

set.seed(123)
dv <- c(11:30, seq(70, 12, by = -2)) + (rnorm(50)/10)
results1 <- stepseg(dv)
print(results1)
# Single term deletions
#
# Model:
# dv ~ 1 + I(ivs[, 1]^1) + I(ivs[, 1] > 20) + I(ivs[, 1] > 20):I(ivs[,
#   1]^1)
#
#               Df Deviance      AIC F value    Pr(>F)
# <none>                0.4 -89.11
# I(ivs[, 1]^1)          1    663.3 279.15    75604 < 2.2e-16 ***
# I(ivs[, 1] > 20)        1  12818.4 427.22 1461966 < 2.2e-16 ***
# I(ivs[, 1]^1):I(ivs[, 1] > 20) 1    4607.1 376.06  525415 < 2.2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

ivs <- cbind(c(1:10, 21:30, 1:30), c(1:40, 50:41))
results2 <- stepseg(dv, ivs, order = 1, allow_interactions = TRUE, update = 2)
# "Beginning the stepwise procedure..."
# "Completed iteration #2"
# "Completed iteration #4"

print(results2)
# Single term deletions
#
# Model:
# dv ~ 1 + I(ivs[, 1]^1) + I(ivs[, 2]^1) + I(ivs[, 2] > 20) + I(ivs[,
#   2] > 20):I(ivs[, 1]^1) + I(ivs[, 2] > 20):I(ivs[, 2]^1) +
#   I(ivs[, 1] > 25) + I(ivs[, 1] > 25):I(ivs[, 1]^1) + I(ivs[,
#   1] > 25):I(ivs[, 2]^1)
#
#               Df Deviance      AIC    F value    Pr(>F)
# <none>                0.31 -92.709
# I(ivs[, 1]^1)          1     0.32 -92.585 1.7790e+00  0.189631
# I(ivs[, 2]^1)          1    28.33 131.492 3.7394e+03 < 2.2e-16 ***
# I(ivs[, 2] > 20)        1  1104.78 314.663 1.4737e+05 < 2.2e-16 ***
# I(ivs[, 1] > 25)        1     0.32 -92.518 1.8364e+00  0.182794
# I(ivs[, 1]^1):I(ivs[, 2] > 20) 1    182.43 224.610 2.4301e+04 < 2.2e-16 ***
# I(ivs[, 2]^1):I(ivs[, 2] > 20) 1    21.79 118.370 2.8668e+03 < 2.2e-16 ***
# I(ivs[, 1]^1):I(ivs[, 1] > 25) 1     0.31 -93.556 9.5640e-01  0.333834
# I(ivs[, 2]^1):I(ivs[, 1] > 25) 1     0.37 -85.378 8.4115e+00  0.005969 **
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

results3 <- stepseg(dv, ivs, order = 2, allow_interactions = FALSE,
  family = "poisson", update = 10) #Generates warnings due to non-Poisson DV
# "Beginning the stepwise procedure..."
# There were 50 or more warnings (use warnings() to see the first 50)

print(results3)
# Single term deletions
#
# Model:
# dv ~ 1 + I(ivs[, 1]^1) + I(ivs[, 1]^2) + I(ivs[, 2]^1) + I(ivs[,
#   2]^2) + I(ivs[, 2] > 20) + I(ivs[, 2] > 20):I(ivs[, 2]^1) +

```

```

#      I(ivs[, 2] > 20):I(ivs[, 2]^2)
#                                     Df Deviance AIC      LRT Pr(>Chi)
# <none>                                0.8352 Inf
# I(ivs[, 1]^1)                        1   6.3978 Inf   5.5627 0.0183474 *
# I(ivs[, 1]^2)                        1  13.0428 Inf  12.2076 0.0004759 ***
# I(ivs[, 2]^1)                        1   1.7744 Inf   0.9392 0.3324749
# I(ivs[, 2]^2)                        1   5.8674 Inf   5.0322 0.0248799 *
# I(ivs[, 2] > 20)                     1  16.5465 Inf  15.7114 7.378e-05 ***
# I(ivs[, 2]^1):I(ivs[, 2] > 20)      1  12.8583 Inf  12.0231 0.0005255 ***
# I(ivs[, 2]^2):I(ivs[, 2] > 20)      1   3.7525 Inf   2.9174 0.0876314 .
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

stepsegcomp

Stepwise Segmented Compositional Regression Analysis

Description

This function performs stepwise segmented compositional regression analysis, a modified form of the procedure described by Britt (2015).

Usage

```

stepsegcomp(
  dv,
  ivs = data.frame(1:nrow(data.frame(dv))),
  start_breakpoints = numeric(0),
  retain_start_breakpoints = TRUE,
  add = 0.15,
  remove = 0.2,
  bonferroni = TRUE,
  add_mode = "mse",
  ord = 1,
  type = "classical",
  allow_interactions = FALSE,
  update = 0,
  verbose = FALSE
)

```

Arguments

| | |
|---------------------------------------|---|
| <code>dv</code> | A <code>data.frame</code> representing values of the dependent variable |
| <code>ivs</code> | A <code>data.frame</code> representing values of the independent variables, coerced to a <code>data.frame</code> if not provided as one (default = <code>data.frame(1:nrow(data.frame(dv)))</code>), which implicitly treats <code>dv</code> as sequentially ordered data and attempts to detect breakpoints in that sequence) |
| <code>start_breakpoints</code> | A numeric vector of breakpoints to be added to the regression model before the first iteration of the stepwise procedure |
| <code>retain_start_breakpoints</code> | If <code>TRUE</code> , then the breakpoints specified in <code>start_breakpoints</code> can never be removed from the model |

| | |
|--------------------|---|
| add | A numeric atomic vector between 0 and 1 indicating the p -value threshold to add coefficients to the model during each forward selection iteration, which must be less than or equal to the remove argument in order to avoid infinite loops (default = .15, as recommended by Britt, 2015) |
| remove | A numeric atomic vector between 0 and 1 indicating the p -value threshold to remove coefficients from the model during each backward selection iteration, which must be greater than or equal to the add argument in order to avoid infinite loops (default = .20, as recommended by Britt, 2015) |
| bonferroni | If TRUE, then the p -values specified in add and remove are divided by the number of terms added to the model for each breakpoint, including all categories of the dependent variable, all independent variables (including any interactions), and all orders of those independent variables (0:ord) |
| add_mode | A character atomic vector (either "p" or "mse") indicating what criterion should be used to select the best candidate block during each forward selection iteration (default = "mse") |
| ord | A non-negative numeric atomic vector indicating the maximum exponent that will be applied to coefficients added to the regression model (default = 1; it is generally recommended to use either order = 0 or order = 1) |
| type | The method used to conduct compositional regression; since <code>comp.reg</code> does not return standard errors for "lmfit" or "spatial", at present, this must be set to "classical" |
| allow_interactions | A boolean value indicating whether interactions may be created between indicator functions representing breakpoints along one independent variable with other independent variables in the model (default = FALSE, which restricts each indicator function to only interact with the independent variable along which its breakpoint was defined) |
| update | A numeric value indicating how many loop iterations should elapse between progress updates (default = 0, which suppresses output) |
| verbose | A boolean value indicating whether the current regression model and results should be outputted as part of the progress report after every step of the stepwise procedure (default = FALSE) |

Details

When using any form of stepwise segmented regression analysis, coefficients are added to and removed from the model in "blocks," with a given block consisting of the indicator function representing a given breakpoint location as well as all interaction terms between that indicator function and independent variables that are eligible to be added to the model.

The `ord` and `allow_interactions` arguments jointly indicate what interaction terms are valid. `ord` indicates the maximum exponent that can be applied to the independent variables, while `allow_interactions` indicates whether interaction terms can be created between a given independent variable and an indicator function that is defined by values of a different independent variable.

All interaction terms that are considered valid, with exponents ranging from 0 to `ord`, are included in a given block. Consider, for instance, an analysis in which `ivs`, the `data.frame` representing the independent variables, has three columns signifying three variables. If `ord = 1` and `allow_interactions = FALSE`, which are their default values in `stepseg`, then the block corresponding to a potential breakpoint at `ivs[, 1] = 10` would include two coefficients:

- `I(ivs[, 1] = 10)`

- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^1)$

Setting `ord = 2` would retain both of the aforementioned coefficients and also add

- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^2)$

to the block. Maintaining `ord = 2` and setting `allow_interactions = TRUE` would result in the block containing a total of seven coefficients:

- $\text{ivs[, 1]} = 10$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^2)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 2]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 2]}^2)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 3]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 3]}^2)$

Note that if `ord = 0`, then each block will only include the indicator function itself (in this example, $I(\text{ivs[, 1]} = 10)$) regardless of the value of `allow_interactions`.

During each forward selection iteration, the algorithm selects the "best" block of coefficients that can be added. If `add_mode = "p"`, then all possible candidate blocks that could be added to the model are evaluated, and whichever block has a p -value less than or equal to all p -values in all other blocks is treated as the "best" block. As long as that p -value is less than the threshold specified by the `add` argument, all coefficients included in the block are added to the model. If `add_mode = "mse"`, then the candidate block whose coefficients would jointly reduce the model mean squared error by the greatest amount is treated as the best block, and all of its coefficients are added to the model if the resulting p -value for at least one of those coefficients would be less than `add`.

During each backward selection iteration, all coefficients listed in `start_formula` are retained in the regression model. Among all remaining coefficients, any blocks whose p -values are all greater than or equal to `remove` are removed from the model. If at least one p -value contained in a given block is less than `remove`, then none of the coefficients in the block are removed from the model.

When `ord = 0`, `add_mode = "p"` is generally acceptable to follow common conventions of stepwise model selection. When `ord > 0`, however, `add_mode = "p"` is more likely to result in spurious breakpoints being added to the model due to intercept and higher-order terms competing with one another, and in rare cases the algorithm may entirely fail to converge. `add_mode = "mse"` is more robust against these issues and is generally recommended as whenever `ord > 0`.

As a caveat, since compositional data are multidimensional in nature, there may be rare cases in which the most suitable breakpoint selected based on MSE does not correspond to a statistically significant change in any individual category of the dependent variable. This can cause the model to prematurely terminate. If an inspection of the final model suggests that such an issue has occurred, then setting `add_mode = "p"` may allow that problem to be overcome, regardless of other weaknesses in that option.

The larger the value of `ord`, the more likely spurious breakpoints are to emerge. As with other regression contexts, you should only increase the complexity of your model (such as by increasing the value of `ord`) when you have a clear reason to do so. Moreover, whenever `ord > 0`, singularities may occur that render some coefficients inestimable. For instance, if `ord = 3` and a pair of breakpoints is identified along the same independent variable, with those two breakpoints occurring two data points apart, there will be insufficient data between the breakpoints to estimate the standard error of the interaction between the indicator function and a cubic term, so it will be reported as NA. This is normal, expected behavior in stepwise segmented regression analysis. In such cases, those NA coefficients can be treated as though they were absent from the model.

Finally, use caution when setting $\text{ord} > 1$, as increasingly high-order terms may compete with lower-order terms in the model and yield unpredictable results. Refer to Britt (2015) for more information.

Value

The final compositional regression model outputted from [comp.reg](#)

References

Britt, B. C. (2015). Stepwise segmented regression analysis: An iterative statistical algorithm to detect and quantify evolutionary and revolutionary transformations in longitudinal data. In S. A. Matei, M. G. Russell, & E. Bertino (Eds.), *Transparency in social media: Tools, methods, and algorithms for mediating online interactions* (pp. 125-144). Heidelberg, Germany: Springer.

Examples

```
dv <- rbind(c(0.058, 0.050, 0.010, 0.221, 0.186, 0.068, 0.407),
            c(0.054, 0.065, 0.009, 0.262, 0.196, 0.069, 0.345),
            c(0.040, 0.114, 0.004, 0.235, 0.191, 0.084, 0.332),
            c(0.044, 0.070, 0.011, 0.356, 0.172, 0.049, 0.298),
            c(0.048, 0.123, 0.007, 0.277, 0.175, 0.053, 0.317),
            c(0.046, 0.101, 0.013, 0.240, 0.212, 0.105, 0.283),
            c(0.058, 0.125, 0.010, 0.257, 0.191, 0.050, 0.309),
            c(0.048, 0.065, 0.010, 0.264, 0.247, 0.083, 0.283),
            c(0.072, 0.067, 0.006, 0.237, 0.197, 0.097, 0.324),
            c(0.040, 0.192, 0.002, 0.243, 0.174, 0.069, 0.280),
            c(0.041, 0.175, 0.005, 0.208, 0.174, 0.075, 0.322),
            c(0.046, 0.098, 0.004, 0.274, 0.206, 0.058, 0.314),
            c(0.037, 0.087, 0.004, 0.237, 0.222, 0.092, 0.321),
            c(0.035, 0.084, 0.004, 0.287, 0.196, 0.067, 0.327),
            c(0.040, 0.061, 0.004, 0.271, 0.226, 0.059, 0.339),
            c(0.038, 0.239, 0.006, 0.228, 0.148, 0.052, 0.289),
            c(0.051, 0.141, 0.004, 0.243, 0.180, 0.068, 0.313),
            c(0.035, 0.156, 0.007, 0.227, 0.199, 0.063, 0.313),
            c(0.037, 0.188, 0.008, 0.258, 0.197, 0.048, 0.264),
            c(0.037, 0.126, 0.002, 0.309, 0.180, 0.032, 0.314),
            c(0.035, 0.092, 0.005, 0.237, 0.242, 0.059, 0.330),
            c(0.043, 0.114, 0.005, 0.225, 0.245, 0.057, 0.311),
            c(0.038, 0.139, 0.002, 0.270, 0.184, 0.053, 0.314),
            c(0.049, 0.080, 0.010, 0.243, 0.215, 0.068, 0.335),
            c(0.041, 0.081, 0.005, 0.219, 0.193, 0.053, 0.408),
            c(0.045, 0.115, 0.001, 0.266, 0.181, 0.052, 0.340),
            c(0.047, 0.140, 0.003, 0.253, 0.170, 0.066, 0.321),
            c(0.049, 0.226, 0.003, 0.239, 0.170, 0.047, 0.266),
            c(0.041, 0.119, 0.002, 0.358, 0.153, 0.035, 0.292),
            c(0.030, 0.165, 0.010, 0.391, 0.090, 0.030, 0.284))

ivs <- c(1:30)
model <- stepsegcomp(dv, ivs, verbose=TRUE)
# "Beginning the stepwise procedure..."
# "End of step 1 ~ Current Breakpoints: 23"
# "End of step 2 ~ Current Breakpoints: 4, 23"
# "No additional breakpoints can be added to the model."
model
# $runtime
#   user  system elapsed
#     0      0      0
#
```

```

# $be
#           [,1]      [,2]      [,3]      [,4]
# (Intercept) -0.2878712 -1.96059047  1.0820294  1.12821655
# xivs[,1]^1   0.2700093  0.06038650  0.2450410  0.08682669
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.7953717  0.19964276  0.4145045  0.09793102
# xI(ivs[,1]>4)*(ivs[,1]^1) -0.2344747 -0.08772856 -0.2252737 -0.06469739
# xI(ivs[,1]>23)*(ivs[,1]^0) -4.2608171 -1.32045504 -3.1891707  1.80206300
# xI(ivs[,1]>23)*(ivs[,1]^1)  0.1431590  0.05065426  0.1120019 -0.08473563
#           [,5]      [,6]
# (Intercept)  0.22781059  1.919198434
# xivs[,1]^1   0.03425128  0.015529618
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.23577716 -0.241304627
# xI(ivs[,1]>4)*(ivs[,1]^1) -0.03957347  0.005926647
# xI(ivs[,1]>23)*(ivs[,1]^0)  1.58830152  0.353969914
# xI(ivs[,1]>23)*(ivs[,1]^1) -0.06607795 -0.022286050
#
# $seb
#           [,1]      [,2]      [,3]      [,4]
# (Intercept)  0.50524000  0.7699158  0.24702365  0.22129975
# xivs[,1]^1   0.18448756  0.2811335  0.09020028  0.08080724
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.56810356  0.8657112  0.27775912  0.24883456
# xI(ivs[,1]>4)*(ivs[,1]^1)  0.18529495  0.2823639  0.09059504  0.08116089
# xI(ivs[,1]>23)*(ivs[,1]^0)  2.12661681  3.2406697  1.03975270  0.93147764
# xI(ivs[,1]>23)*(ivs[,1]^1)  0.07985208  0.1216835  0.03904155  0.03497594
#           [,5]      [,6]
# (Intercept)  0.32977623  0.21663586
# xivs[,1]^1   0.12041725  0.07910423
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.37080803  0.24359039
# xI(ivs[,1]>4)*(ivs[,1]^1)  0.12094425  0.07945042
# xI(ivs[,1]>23)*(ivs[,1]^0)  1.38806839  0.91184680
# xI(ivs[,1]>23)*(ivs[,1]^1)  0.05212042  0.03423883
#
# $est
# NULL
#
# $t
#           [,1]      [,2]      [,3]      [,4]
# (Intercept) -0.5697712 -2.5464998  4.380266  5.0981376
# xivs[,1]^1   1.4635635  0.2147965  2.716632  1.0744914
# xI(ivs[,1]>4)*(ivs[,1]^0)  1.4000470  0.2306113  1.492316  0.3935588
# xI(ivs[,1]>4)*(ivs[,1]^1) -1.2654136 -0.3106933 -2.486601 -0.7971499
# xI(ivs[,1]>23)*(ivs[,1]^0) -2.0035660 -0.4074636 -3.067240  1.9346283
# xI(ivs[,1]>23)*(ivs[,1]^1)  1.7928026  0.4162787  2.868788 -2.4226831
#           [,5]      [,6]
# (Intercept)  0.6908036  8.85909836
# xivs[,1]^1   0.2844383  0.19631842
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.6358470 -0.99061637
# xI(ivs[,1]>4)*(ivs[,1]^1) -0.3272042  0.07459553
# xI(ivs[,1]>23)*(ivs[,1]^0)  1.1442531  0.38819012
# xI(ivs[,1]>23)*(ivs[,1]^1) -1.2677940 -0.65089993
#
# $p
#           [,1]      [,2]      [,3]      [,4]
# (Intercept)  0.57412808  0.01771475  0.0002008412  3.242277e-05
# xivs[,1]^1   0.15628846  0.83174067  0.0120382387  2.932869e-01
# xI(ivs[,1]>4)*(ivs[,1]^0)  0.17429429  0.81957130  0.1486457861  6.973815e-01
# xI(ivs[,1]>4)*(ivs[,1]^1)  0.21787098  0.75871609  0.0202485998  4.331800e-01

```

```

# xI(ivs[,1]>23)*(ivs[,1]^0) 0.05653118 0.68727920 0.0052865773 6.490810e-02
# xI(ivs[,1]>23)*(ivs[,1]^1) 0.08561900 0.68090526 0.0084573677 2.332019e-02
#                               [,5]      [,6]
# (Intercept)                0.4963166 4.953975e-09
# xivs[,1]^1                  0.7785143 8.460138e-01
# xI(ivs[,1]>4)*(ivs[,1]^0) 0.5308933 3.317559e-01
# xI(ivs[,1]>4)*(ivs[,1]^1) 0.7463503 9.411548e-01
# xI(ivs[,1]>23)*(ivs[,1]^0) 0.2638031 7.012974e-01
# xI(ivs[,1]>23)*(ivs[,1]^1) 0.2170338 5.212948e-01

```

stepsegdir

Stepwise Segmented Dirichlet Regression Analysis

Description

This function performs stepwise segmented Dirichlet regression analysis, a modified form of the procedure described by Britt (2015).

Usage

```

stepsegdir(
  dv,
  ivs = data.frame(1:nrow(data.frame(dv))),
  start_breakpoints = numeric(0),
  retain_start_breakpoints = TRUE,
  add = 0.15,
  remove = 0.2,
  bonferroni = TRUE,
  add_mode = "mse",
  ord = 1,
  parameterization = "common",
  allow_interactions = FALSE,
  iterlim = 1000,
  tol1 = 1e-05,
  tol2 = 1e-10,
  update = 0,
  verbose = FALSE
)

```

Arguments

| | |
|---------------------------------------|---|
| <code>dv</code> | A data.frame representing values of the dependent variable |
| <code>ivs</code> | A data.frame representing values of the independent variables, coerced to a data.frame if not provided as one (default = <code>data.frame(1:nrow(data.frame(dv)))</code>), which implicitly treats <code>dv</code> as sequentially ordered data and attempts to detect breakpoints in that sequence) |
| <code>start_breakpoints</code> | A numeric vector of breakpoints to be added to the regression model before the first iteration of the stepwise procedure |
| <code>retain_start_breakpoints</code> | If TRUE, then the breakpoints specified in <code>start_breakpoints</code> can never be removed from the model |

| | |
|--------------------|---|
| add | A numeric atomic vector between 0 and 1 indicating the p -value threshold to add coefficients to the model during each forward selection iteration, which must be less than or equal to the remove argument in order to avoid infinite loops (default = .15, as recommended by Britt, 2015) |
| remove | A numeric atomic vector between 0 and 1 indicating the p -value threshold to remove coefficients from the model during each backward selection iteration, which must be greater than or equal to the add argument in order to avoid infinite loops (default = .20, as recommended by Britt, 2015) |
| bonferroni | If TRUE, then the p -values specified in add and remove are divided by the number of terms added to the model for each breakpoint, including all categories of the dependent variable, all independent variables (including any interactions), and all orders of those independent variables (0:ord) |
| add_mode | A character atomic vector (either "p" or "mse") indicating what criterion should be used to select the best candidate block during each forward selection iteration (default = "mse") |
| ord | A non-negative numeric atomic vector indicating the maximum exponent that will be applied to coefficients added to the regression model (default = 1; it is generally recommended to use either order = 0 or order = 1) |
| parameterization | The parameterization used to perform Dirichlet regression; since DirichReg does not return standard errors for "lmfit" or "spatial", at present, this must be set to "classical" |
| allow_interactions | A boolean value indicating whether interactions may be created between indicator functions representing breakpoints along one independent variable with other independent variables in the model (default = FALSE, which restricts each indicator function to only interact with the independent variable along which its breakpoint was defined) |
| iterlim | The maximum number of iterations permitted for convergence in each regression, as conducted by DirichReg |
| tol1 | The convergence criterion for BFGS optimization in each regression, as conducted by DirichReg |
| tol2 | The convergence criterion for NR optimization in each regression, as conducted by DirichReg |
| update | A numeric value indicating how many loop iterations should elapse between progress updates (default = 0, which suppresses output) |
| verbose | A boolean value indicating whether the current regression model and results should be outputted as part of the progress report after every step of the stepwise procedure (default = FALSE) |

Details

When using any form of stepwise segmented regression analysis, coefficients are added to and removed from the model in "blocks," with a given block consisting of the indicator function representing a given breakpoint location as well as all interaction terms between that indicator function and independent variables that are eligible to be added to the model.

The ord and allow_interactions arguments jointly indicate what interaction terms are valid. ord indicates the maximum exponent that can be applied to the independent variables, while allow_interactions indicates whether interaction terms can be created between a given independent variable and an indicator function that is defined by values of a different independent variable.

All interaction terms that are considered valid, with exponents ranging from 0 to `ord`, are included in a given block. Consider, for instance, an analysis in which `ivs`, the data.frame representing the independent variables, has three columns signifying three variables. If `ord = 1` and `allow_interactions = FALSE`, which are their default values in `stepseg`, then the block corresponding to a potential breakpoint at `ivs[, 1] = 10` would include two coefficients:

- $I(\text{ivs[, 1]} = 10)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^1)$

Setting `ord = 2` would retain both of the aforementioned coefficients and also add

- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^2)$

to the block. Maintaining `ord = 2` and setting `allow_interactions = TRUE` would result in the block containing a total of seven coefficients:

- $\text{ivs[, 1]} = 10$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 1]}^2)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 2]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 2]}^2)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 3]}^1)$
- $I(\text{ivs[, 1]} = 10) * (\text{ivs[, 3]}^2)$

Note that if `ord = 0`, then each block will only include the indicator function itself (in this example, $I(\text{ivs[, 1]} = 10)$) regardless of the value of `allow_interactions`.

During each forward selection iteration, the algorithm selects the "best" block of coefficients that can be added. If `add_mode = "p"`, then all possible candidate blocks that could be added to the model are evaluated, and whichever block has a p -value less than or equal to all p -values in all other blocks is treated as the "best" block. As long as that p -value is less than the threshold specified by the `add` argument, all coefficients included in the block are added to the model. If `add_mode = "mse"`, then the candidate block whose coefficients would jointly reduce the model mean squared error by the greatest amount is treated as the best block, and all of its coefficients are added to the model if the resulting p -value for at least one of those coefficients would be less than `add`.

During each backward selection iteration, all coefficients listed in `start_formula` are retained in the regression model. Among all remaining coefficients, any blocks whose p -values are all greater than or equal to `remove` are removed from the model. If at least one p -value contained in a given block is less than `remove`, then none of the coefficients in the block are removed from the model.

When `ord = 0`, `add_mode = "p"` is generally acceptable to follow common conventions of stepwise model selection. When `ord > 0`, however, `add_mode = "p"` is more likely to result in spurious breakpoints being added to the model due to intercept and higher-order terms competing with one another, and in rare cases the algorithm may entirely fail to converge. `add_mode = "mse"` is more robust against these issues and is generally recommended as whenever `ord > 0`.

As a caveat, since the Dirichlet distribution is multidimensional in nature, there may be rare cases in which the most suitable breakpoint selected based on MSE does not correspond to a statistically significant change in any individual category of the dependent variable. This can cause the model to prematurely terminate. If an inspection of the final model suggests that such an issue has occurred, then setting `add_mode = "p"` may allow that problem to be overcome, regardless of other weaknesses in that option.

The larger the value of `ord`, the more likely spurious breakpoints are to emerge. As with other regression contexts, you should only increase the complexity of your model (such as by increasing

the value of `ord`) when you have a clear reason to do so. Moreover, whenever `ord > 0`, singularities may occur that render some coefficients inestimable. For instance, if `ord = 3` and a pair of breakpoints is identified along the same independent variable, with those two breakpoints occurring two data points apart, there will be insufficient data between the breakpoints to estimate the standard error of the interaction between the indicator function and a cubic term, so it will be reported as NA. This is normal, expected behavior in stepwise segmented regression analysis. In such cases, those NA coefficients can be treated as though they were absent from the model.

Finally, use caution when setting `ord > 1`, as increasingly high-order terms may compete with lower-order terms in the model and yield unpredictable results. Refer to Britt (2015) for more information.

Value

The final Dirichlet regression model outputted from [DirichReg](#)

References

Britt, B. C. (2015). Stepwise segmented regression analysis: An iterative statistical algorithm to detect and quantify evolutionary and revolutionary transformations in longitudinal data. In S. A. Matei, M. G. Russell, & E. Bertino (Eds.), *Transparency in social media: Tools, methods, and algorithms for mediating online interactions* (pp. 125-144). Heidelberg, Germany: Springer.

Examples

```
set.seed(2023)
dv <- rbind(DirichletReg::rdirichlet(12, c(8,8,32)),
            DirichletReg::rdirichlet(18, c(40,24,4)))
model <- stepsegdir(dv, add_mode="mse", update=10, verbose=TRUE)
# "Beginning the stepwise procedure..."
# "Testing observation 10"
# "Testing observation 20"
# "End of step 1 ~ Current Breakpoints: 10"
# "Testing observation 10"
# "Testing observation 20"
# "No additional breakpoints can be added to the model."
# Warning message:
# In min(abs(summary(model)$coef.mat[indices, 4]), na.rm = T) :
# no non-missing arguments to min; returning Inf
summary(model)
#
# Call:
# DirichReg(formula = as.formula(paste0("dv ~ ", paste(current_terms, collapse
# = " + "))), data = df, model = parameterization, control = list(iterlim =
# iterlim, tol1 = tol1, tol2 = tol2))
#
# Standardized Residuals:
#           Min           1Q       Median           3Q            Max
# V1 -2.1452  -0.6651  -0.2261   0.6674   2.1566
# V2 -1.7372  -1.1093   0.3604   0.8497   1.6465
# V3 -1.7572  -0.6547  -0.2742   0.6269   2.2728
#
# -----
# Beta-Coefficients for variable no. 1: V1
#
#               Estimate Std. Error z value Pr(>|z|)
# (Intercept)      4.57845    0.68780   6.657 2.8e-11 ***
# I(ivs[, 1]^1)     -0.22604    0.09559  -2.365  0.0180 *
# I(ivs[, 1] > 12)TRUE -0.54752    1.24725  -0.439  0.6607
```

```

# I(ivs[, 1]^1):I(ivs[, 1] > 12)TRUE  0.21705    0.10660    2.036    0.0417 *
# -----
# Beta-Coefficients for variable no. 2: V2
#
#               Estimate Std. Error z value Pr(>|z|)
# (Intercept)      4.35351    0.69899    6.228 4.72e-10 ***
# I(ivs[, 1]^1)     -0.18497    0.09834   -1.881    0.060 .
# I(ivs[, 1] > 12)TRUE -0.84702    1.24069   -0.683    0.495
# I(ivs[, 1]^1):I(ivs[, 1] > 12)TRUE  0.17734    0.10875    1.631    0.103
# -----
# Beta-Coefficients for variable no. 3: V3
#
#               Estimate Std. Error z value Pr(>|z|)
# (Intercept)      5.97326    0.68569    8.711 < 2e-16 ***
# I(ivs[, 1]^1)     -0.21326    0.09505   -2.244    0.02485 *
# I(ivs[, 1] > 12)TRUE -3.70315    1.21792   -3.041    0.00236 **
# I(ivs[, 1]^1):I(ivs[, 1] > 12)TRUE  0.18075    0.10538    1.715    0.08630 .
# -----
# Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Log-likelihood: 118.8 on 12 df (83 BFGS + 2 NR Iterations)
# AIC: -213.5, BIC: -196.7
# Number of Observations: 30
# Link: Log
# Parametrization: common

```

Index

`comp.reg`, [3](#), [4](#), [11](#), [13](#)

`DirichReg`, [2](#), [2](#), [3](#), [16](#), [18](#)

`drop1`, [7](#), [8](#)

`family`, [5](#)

`formula`, [5](#)

`get_p_from_model`, [3](#)

`get_t_from_model`, [3](#)

`glm`, [7](#), [8](#)

`print.stepseg_output`, [4](#)

`stepseg`, [4](#), [4](#)

`stepsegcomp`, [10](#)

`stepsegdir`, [15](#)