

Package: chaid (via r-universe)

September 26, 2026

Title Chi-Squared Automatic Interaction Detection

Version 0.1-3

Date 2026-08-21

Description An implementation of CHAID, a type of decision tree technique for a nominal scaled dependent variable published in 1980 by Gordon V. Kass. CHAID stands for Chi-Squared Automatic Interaction Detection and detects interactions between categorized variables of a data set, one of which is the dependent variable. The remaining variables may or may not be ordered. An algorithm for recursive partitioning is implemented, based on maximizing the significance of a chi-squared statistic for cross-tabulations between the dependent variable and the predictors at each partition. The data are partitioned into mutually exclusive, exhaustive subsets that best describe the dependent variable. Multiway splits are used by default.

Depends R (>= 2.10.0), partykit

License GPL-2

LazyLoad yes

LazyData yes

URL <https://zeileis.codeberg.page/chaid/>

BugReports <https://codeberg.org/zeileis/chaid/issues>

Repository <https://zeileis.r-universe.dev>

Date/Publication 2026-08-21 22:24:04 UTC

RemoteUrl <https://codeberg.org/zeileis/chaid>

RemoteRef HEAD

RemoteSha 4ef45a6676dc2a6ebb782089acf03935d0ffaefb

Contents

chaid-package	2
chaid	3
USvote	5
Index	7

chaid-package	<i>Chi-Squared Automatic Interaction Detection</i>
---------------	--

Description

This package offers an implementation of CHAID, a type of decision tree technique for a nominal scaled dependent variable published in 1980 by Gordon V. Kass. CHAID stands for Chi-Squared Automatic Interaction Detection and detects interactions between categorized variables of a data set, one of which is the dependent variable. The remaining variables may or may not be ordered. An algorithm for recursive partitioning is implemented, based on maximizing the significance of a chi-squared statistic for cross-tabulations between the dependent variable and the predictors at each partition. The data are partitioned into mutually exclusive, exhaustive subsets that best describe the dependent variable. Multiway splits are used by default.

Details

Package: chaid
License: GPL-2

The CHAID algorithm is subdivided in five different steps. Step 1 calculates the cross-tabulates for each dependent variable in turn and processing step 2 and step 3. In the first instance they include the search of two variable values which show the least significant differing sub-table. This quest includes permutation tests, if necessary. If the p-value exceeds the critical value, the categories are merged. This step 2 will be repeated. Secondly the most significant binary split should be found for each connected category with more than two components (step 3). The split is carried into execution, if the significance level is beyond a critical value. The algorithm returns to step 2. Step 4 computes the significance of each optimally combined independent variable. Therefore the implementation adjusts the p-value: I.e. using the Bonferroni multipliers according to the characteristics of the predictor – ordered or nominal case (q.v. Kass 1980: 122). The highest significant variable causes the division of the data set, if the critical value is passed. In step 5 the algorithm jumps to step 1, for each partition of the data set, which has not been investigated. chaid returns an object of class constparty, see package [party](#).

The package was implemented by students participating in an advanced R programming course taught at the Department of Statistics (University of Munich), winter term 2008/2009.

Author(s)

The FoRt Student Project Team, with participants:

Philip Bleninger, Catharina Brockhaus, Martina Feilke, Veronika Fensterer, Armin Graf, Stefanie Grunow, Elizabeth Heller, Matthias Hunger, Torsten Hothorn, Monika Jelizarow, Sara Kleyer, Julia Kopf, Sarah Maierhofer, Lisa Moest, Holger Reulen, Adrian Richter, Gunther Schauburger, Julia Schiele, Micha Schneider, Judith Schwitulla, Stephanie Thiemichen, Xuelin Wang, Anett Wins

Maintainer: Torsten Hothorn <Torsten.Hothorn@R-project.org>

References

G. V. Kass (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Applied Statistics*, **29**(2), 119–127.

chaid	<i>Chi-Squared Automatic Interaction Detection</i>
-------	--

Description

Fits a classification tree by the CHAID algorithm.

Usage

```
chaid(formula, data, subset, weights, na.action = na.omit,
      control = chaid_control())
chaid_control(alpha2 = 0.05, alpha3 = -1, alpha4 = 0.05, minsplit = 20,
              minbucket = 7, minprob = 0.01, stump = FALSE, maxheight = -1)
```

Arguments

formula	an object of class formula (or one that can be coerced to that class): a symbolic description of the model to be fitted. Both response and all covariates are assumed to be categorical (either ordered or not).
data	an optional data frame containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>chaid</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
na.action	a function which indicates what should happen when the data contain NAs. The default is <code>na.omit</code> .
control	hyper parameters of the algorithm as returned by <code>chaid_control</code> .
alpha2	Level of significance used for merging of predictor categories (step 2).

alpha3	If set to a positive value < 1 , level of significance used for the the splitting of former merged categories of the predictor (step 3). Otherwise, step 3 is omitted (the default).
alpha4	Level of significance used for splitting of a node in the most significant predictor (step 5).
minsplit	Number of observations in splitted response at which no further split is desired.
minbucket	Minimum number of observations in terminal nodes.
minprob	Mininimum frequency of observations in terminal nodes.
stump	only root node splits are performed.
maxheight	Maximum height for the tree.

Details

The current implementation only accepts nominal or ordinal categorical predictors. When predictors are continuous, they have to be transformed into ordinal predictors before using the following algorithm.

Merging: For each predictor variable X in turn, merge non-significant categories. Each final category of X will result in one child node if X is used to split the node. The merging step also calculates the adjusted p-value that is to be used in the splitting step.

1. If X has 1 category only, stop and set the adjusted p-value to be 1.
2. If X has 2 categories, go to step 8.
3. Else, find the allowable pair of categories of X (an allowable pair of categories for ordinal predictor is two adjacent categories, and for nominal predictor is any two categories) that is least significantly different (i.e., most similar). The most similar pair is the pair whose test statistic gives the largest p-value with respect to the dependent variable Y . How to calculate p-value under various situations will be described in later sections.
4. For the pair having the largest p-value, check if its p-value is larger than a user-specified alpha-level α_2 . If it does, this pair is merged into a single compound category. Then a new set of categories of X is formed. If it does not, then go to step 7.
5. (Optional) If the newly formed compound category consists of three or more original categories, then find the best binary split within the compound category which p-value is the smallest. Perform this binary split if its p-value is not larger than an alpha-level α_3 .
6. Go to step 2.
7. (Optional) Any category having too few observations (as compared with a user-specified minimum segment size) is merged with the most similar other category as measured by the largest of the p-values.
8. The adjusted p-value is computed for the merged categories by applying Bonferroni adjustments that are to be discussed later.

Splitting: The best split for each predictor is found in the merging step. The splitting step selects which predictor to be used to best split the node. Selection is accomplished by comparing the adjusted p-value associated with each predictor. The adjusted p-value is obtained in the merging step. 1. Select the predictor that has the smallest adjusted p-value (i.e., most significant). 2. If this adjusted p-value is less than or equal to a user-specified alpha-level α_4 , split the node using this predictor. Else, do not split and the node is considered as a terminal node.

Stopping: The stopping step checks if the tree growing process should be stopped according to the following stopping rules. a) If a node becomes pure; that is, all cases in a node have identical values of the dependent variable, the node will not be split. b) If all cases in a node have identical values for each predictor, the node will not be split. c) If the current tree depth reaches the user specified maximum tree depth limit value, the tree growing process will stop. d) If the size of a node is less than the user-specified minimum node size value, the node will not be split. e) If the split of a node results in a child node whose node size is less than the user-specified minimum child node size value, child nodes that have too few cases (as compared with this minimum) will merge with the most similar child node as measured by the largest of the p-values. However, if the resulting number of child nodes is 1, the node will not be split. f) If the trees height is a positive value and equals the maxheight.

Value

An object of class `constparty`, see package [party](#).

References

G. V. Kass (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Applied Statistics*, **29**(2), 119–127.

Examples

```
library("chaid")
data("USvote", package = "chaid")

### fit tree to subsample
set.seed(290875)
USvoteS <- USvote[sample(1:nrow(USvote), 1000),]

ctrl <- chaid_control(minsplit = 200, minprob = 0.1)
chaidUS <- chaid(vote3 ~ ., data = USvoteS, control = ctrl)

print(chaidUS)
plot(chaidUS)
```

USvote

U.S. 2000 Election Data

Description

Data from a post-election survey following the year 2000 U.S. presidential elections.

Usage

```
data("USvote")
```

Format

A data frame with 10645 observations on the following 6 variables.

vote3 candidate voted for Gore or Bush

gender gender, a factor with levels male and female

ager age group, an ordered factor with levels 18-24 < 25-34 < 35-44 < 45-54 < 55-64 < 65+

empstat status of employment, a factor with levels yes, no or retired

educr status of education, an ordered factor with levels <HS < HS < >HS < College < Post Coll

marstat status of living situation, a factor with levels married, widowed, divorced or never married

Details

The dataset is based on data from a post-election survey on persons who voted for either Bush or Gore in the 2000 U.S. election. The specific variables are related to the publication of Magidson and Vermunt (2005).

Further information (and datasets) about the 2000 U.S. election and other National Election Studies is available on the American National Election Studies Web site (<https://electionstudies.org/>).

Source

Latent GOLD 4.5: Sample Data Sets

https://web.archive.org/web/20090113231021/http://www.statisticalinnovations.com/products/latentgold_datasets.html#USselection2000reg

References

Jay Magidson and Jeroen K. Vermunt (2005). An Extension of the CHAID Tree-Based Segmentation Algorithm to Multiple Dependent Variables. In: C. Weihs & W. Gaul (Editors), Classification: The Ubiquitous Challenge, pages 176–183. Heidelberg: Springer-Verlag. <https://jeroenvermunt.nl/gfk12004b.pdf>

Examples

```
library("chaid")
data("USvote", package = "chaid")

### fit tree to subsample
set.seed(290875)
USvoteS <- USvote[sample(1:nrow(USvote), 2000),]
chaidUS <- chaid(vote3 ~ gender + ager + empstat + educr + marstat, data = USvoteS)

print(chaidUS)
plot(chaidUS)
```

Index

- * **datasets**

- USvote, [5](#)

- * **package**

- chaid-package, [2](#)

- * **tree**

- chaid, [3](#)

chaid, [3](#)

chaid-package, [2](#)

chaid_control (chaid), [3](#)

party, [2](#), [5](#)

USvote, [5](#)