Package: ohenery (via r-universe)

September 5, 2024

Maintainer Steven E. Pav <shabbychef@gmail.com>

Version 0.1.1

Date 2019-10-14

License LGPL-3

Title Modeling of Ordinal Random Variables via Softmax Regression

BugReports https://github.com/shabbychef/ohenery/issues

Description Supports the modeling of ordinal random variables, like the outcomes of races, via Softmax regression, under the Harville <doi:10.1080/01621459.1973.10482425> and Henery <doi:10.1111/j.2517-6161.1981.tb01153.x> models.

Depends R (>= 3.0.2)

Imports Rcpp (>= 0.12.3), maxLik, magrittr, methods, dplyr

LinkingTo Rcpp

Suggests rlang, tidyr, forcats, microbenchmark, testthat, numDeriv, ggplot2, scales, knitr

Encoding UTF-8

URL https://github.com/shabbychef/ohenery

Collate 'data.r' 'harsm.r' 'hensm.r' 'linodds.r' 'ohenery.r' 'RcppExports.R' 'rsm.r' 'utils.r'

RoxygenNote 6.1.1

Repository https://shabbychef.r-universe.dev

RemoteUrl https://github.com/shabbychef/ohenery

RemoteRef HEAD

RemoteSha 4b7613bc9ceb495adc3d888d93c87384c9a612e9

Contents

`as.linodds` *An object for modeling linear odds.*

Description

A model for odds linear in some feature.

Usage

```
as.linodds(object, formula, beta)

## S3 method for class 'linodds'
predict(object, newdata, type = c("eta", "mu",
  "erank"), na.action = na.pass, group = NULL, ...)

## S3 method for class 'linodds'
coef(object, ...)
```

Arguments

`object`	some list-like object.
`formula`	an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`beta`	the fit coefficients.
`newdata`	a `data.frame` from which we can extract a model frame via the formula of the `object`.
`type`	indicates which prediction should be returned: `eta` The odds.

- `mu` The probability.
- `erank` The expected rank.

`na.action` How to deal with missing values in `y`, `g`, `X`, `wt`, `eta0`.

`group` the string name of the group variable in the data, or a bare character with the group name. The group indices need not be integers, but that is more efficient. They need not be sorted.

`...` other arguments.

Details

An object which holds a formula, some fit coefficients β which fit in that formula to generate odds in odds space. The odds can then be converted, via `predict.linodds` to probabilities, or to expected ranks under the Harville model. Both `harsm` and `hensm` return objects of class `linodds`.

We think of linear odds as $\eta = x^{\top}\beta$, for independent variables x. The odds, η are converted to probabilities, μ via $\mu = c \exp \eta$, where the constant c is chosen so the μ for a given matching sum to one.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

`harsm`, `hensm`.

`smax`, `harsm_invlink`.

`best_picture` *Oscar Award Best Picture Data*

Description

Historical data on the Best Picture nominees and winners from 1934 through 2014.

Usage

```
data(best_picture)
```

Format

A `data.frame` object with 484 observations and 19 columns. The columns are defined as follows:

`year` The integer year of the nomination. These span from 1934 through 2014. Note that the number of films nominated per year varies from 5 to 12.

`film` The title of the film.

`winner` A logical for whether the film won the Oscar for Best Picture. There is exactly one winning film per year.

`nominated_for_Writing` A logical indicating whether the film was also nominated for a Writing award that year.

`nominated_for_BestDirector` A logical indicating whether the film was also nominated for Best Director award that year.

`nominated_for_BestActress` A logical indicating whether the film was also nominated for at least one Best Actress award that year.

`nominated_for_BestActor` A logical indicating whether the film was also nominated for at least one Best Actor award that year.

`nominated_for_BestFilmEditing` A logical indicating whether the film was also nominated for at least one Best Film Editing award that year.

`Adventure` A double computed as a 0/1 indicator of whether “Adventure” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Biography` A double computed as a 0/1 indicator of whether “Biography” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Comedy` A double computed as a 0/1 indicator of whether “Comedy” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Crime` A double computed as a 0/1 indicator of whether “Crime” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Drama` A double computed as a 0/1 indicator of whether “Drama” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`History` A double computed as a 0/1 indicator of whether “History” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Musical` A double computed as a 0/1 indicator of whether “Musical” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Romance` A double computed as a 0/1 indicator of whether “Romance” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Thriller` A double computed as a 0/1 indicator of whether “Thriller” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`War` A double computed as a 0/1 indicator of whether “War” was one of the genres tagged for the film in IMDb, divided by the total count of genres tagged for the film.

`Other` A double computed as 1 minus the sum of the other genre indicators. Effectively this is is the sum of indicators for “Mystery”, “Family”, “Fantasy”, “Action”, “Western”, “Music”, “Sport”, “Sci Fi”, “Film-Noir”, “Animation”, and “Horror” divided by the total count of genres tagged for the film.

Note

“Oscar” is a copyright property of the Academy of Motion Picture Arts and Sciences. IMDb is owned by Amazon.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

Source

Awards data were sourced from Wikipedia, while genre data were sourced from IMDb. Any errors in transcription are the fault of the package author.

Examples

```
library(dplyr)
data(best_picture)

best_picture %>%
  group_by(nominated_for_BestDirector) %>%
  summarize(propwin=mean(winner)) %>%
  ungroup()
best_picture %>%
  group_by(nominated_for_BestActor) %>%
  summarize(propwin=mean(winner)) %>%
  ungroup()
# hmmmm.
best_picture %>%
  group_by(nominated_for_BestActress) %>%
  summarize(propwin=mean(winner)) %>%
 ungroup()
```

diving *Olympic Diving Data*

Description

One hundred years of Men's Olympic Platform Diving records.

Usage

```
data(diving)
```

Format

A `data.frame` object with 695 observations and 13 columns.

The columns are defined as follows:

`Name` The participant's name.

`Age` The age of the participant at the time of the Olympics. Some values missing.

`Height` The height of the participant at the time of the Olympics, in centimeters. Many values missing.

`Weight` The height of the participant at the time of the Olympics, in kilograms. Many values missing.

`Team` The string name of the team (country) which the participant represented.

NOC The string name of the National Olympic Committee which the participant represented. This is a three character code.

Games The string name of the Olympic games, including a year.

Year The integer year of the Olympics. These range from 1906 through 2016.

City The string name of the host city.

Medal A string of “Gold”, “Silver”, “Bronze” or NA.

EventId A unique integer ID for each Olympics.

AthleteId A unique integer ID for each participant.

HOST_NOC The string name of the National Olympic Committee of the nation hosting the Olympics. This is a three character code.

Note

The author makes no guarantees regarding correctness of this data.

Please attribute this data to the upstream harvester.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

Source

Data were collected by Randi Griffin from the website “sports-reference.com”, and staged on Kaggle at https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results.

Examples

```
library(dplyr)
library(forcats)
data(diving)

fitdat <- diving %>%
  mutate(Finish=case_when(grepl('Gold',Medal)   ~ 1,
                          grepl('Silver',Medal) ~ 2,
                          grepl('Bronze',Medal) ~ 3,
                          TRUE ~ 4)) %>%
  mutate(weight=ifelse(Finish <= 3,1,0)) %>%
 mutate(cut_age=cut(coalesce(Age,22.0),c(12,19.5,21.5,22.5,25.5,99),include.lowest=TRUE)) %>%
  mutate(country=forcats::fct_relevel(forcats::fct_lump(factor(NOC),n=5),'Other')) %>%
  mutate(home_advantage=NOC==HOST_NOC)

hensm(Finish ~ cut_age + country + home_advantage,data=fitdat,weights=weight,group=EventId,ngamma=3)
```

erank *Expected rank under the Harville model.*

Description

Compute the expected rank of a bunch of entries based on their probability of winning under the Harville model.

Usage

```
erank(mu)
```

Arguments

`mu`	a vector giving the probabilities. Should sum to one.

Details

Given the vector μ, we compute the expected rank of each entry, under the Harville model, where tail probabilities of winning remain proportional.

Under the Harville model, the probability that the ith element is assigned value 1 is

$$\pi_{1,i} = \frac{\mu_i}{\sum_j \mu_j}.$$

Once an element has been assigned a 1, the Harville procedure removes it from the set and iterates. If there are k elements in μ, then the ith element can be assigned any place between 1 and k. This function computes the expected value of that random variable.

While a naive implementation of this function would take time factorial in k, this implementation takes time quadratic in k, since it can be shown that the expected rank of the ith element takes value

$$e_i = k + \frac{1}{2} - \sum_j \frac{\mu_i}{\mu_i + \mu_j}.$$

Value

The expected ranks, a vector.

Note

we should have the sum of ranks equal to the sum of `1:length(mu)`.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

Examples

```
# a garbage example
set.seed(12345)
mus <- runif(12)
mus <- mus / sum(mus)
erank(mus)

# confirm the expected rank via simulation
set.seed(123)
mus <- runif(6,min=0,max=2)
mus <- mus / sum(mus)
set.seed(101)
emp <- rowMeans(replicate(200,rhenery(mu=mus,gamma=rep(1,length(mus)-1))))
(emp - erank(mus)) / emp


if (require(microbenchmark)) {
  p10 <- 1:10 / sum(1:10)
  p16 <- 1:16 / sum(1:16)
  p24 <- 1:24 / sum(1:24)
  microbenchmark(erank(p10), erank(p16), erank(p24))
}
```

---

harsm *Friendly interface to softmax regression under Harville model.*

Description

A user friendly interface to the softmax regression under the Harville model.

Usage

```
harsm(formula, data, group = NULL, weights = NULL,
  na.action = na.omit)

## S3 method for class 'harsm'
vcov(object, ...)

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

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. The details of model specification are given under ‘Details’.

`data` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.

`group` the string name of the group variable in the data, or a bare character with the group name. The group indices need not be integers, but that is more efficient. They need not be sorted.

`weights` an optional vector of weights, or the string or bare name of the weights in the `data` for use in the fitting process. The weights are attached to the outcomes, not the participant. Set to `NULL` for none.

`na.action` How to deal with missing values in the outcomes, groups, weights, etc.

`object` an object of class `harsm`.

`...` additional arguments to be passed to the low level regression fitting functions (see below).

`x` logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.

Details

Performs a softmax regression by groups, via Maximum Likelihood Estimation, under the Harville model. We fit β where odds are $\eta = x^{\top}\beta$ for independent variables x. The probability of taking first place is then $\mu = c \exp \eta$, where the c is chosen so the μ sum to one. Under the Harville model, conditional on the first place finisher having been observed, the probability model for second (and successive) places with the probabilities of the remaining participants renormalized.

The `print` method of the `harsm` object includes a display of the R-squared. This measures the improvement in squared errors of the expected rank from the model over the null model which posits that all odds are equal. When the formula includes an offset, a ‘delta R-squared’ is also output. This is the improvement in predicted ranks over the model based on the offset term. Note that the expected ranks are only easy to produce under the Harville model; under the Henery model, the summary R-squared statistics are not produced. Note that this computation uses weighted sums of squares, as the weights contribute to the likelihood term. However, the square sum computation does not take into account the standard error of the rank, and so unlike in linear regression, the softmax regression does not always give positive R-squareds, and the statistic is otherwise hard to interpret.

Value

An object of class `harsm`, but also of `maxLik` with the fit.

Note

Since version 0.1.0 of this package, the normalization of weights used in this function have changed under the hood. This is to give correct inference in the case where zero weights are used to signify finishing places were not observed. If in doubt, please confirm inference by simulations, taking as example the simulations in the README.

This regression may give odd results when the outcomes are tied, imposing an arbitrary order on the tied outcomes. Moreover, no warning may be issued in this case. In future releases, ties may be dealt with differently, perhaps in analogy to how ties are treated in the Cox Proportional Hazards regression, using the methods of Breslow or Efron.

To avoid incorrect inference when only the top performers are recorded, and all others are effectively tied, one should use weighting. Set the weights to zero for participants who are tied non-winners, and one for the rest So for example, if you observe the Gold, Silver, and Bronze medal winners of an Olympic event that had a starting field of 12 participants, set weights to 1 for the medal winners, and 0 for the others. Note that the weights do not attach to the participants, they attach to the place they took.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

harsmfit, harsmlik.

Examples

```
nfeat <- 5
set.seed(1234)
g <- ceiling(seq(0.1,1000,by=0.1))
X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
beta <- rnorm(nfeat)
eta <- X %*% beta
y <- rsm(eta,g)
# now the pretty frontend
data <- cbind(data.frame(outcome=y,race=g),as.data.frame(X))

fmla <- outcome ~ V1 + V2 + V3 + V4 + V5
fitm <- harsm(fmla,data,group=race)

eta0 <- rowMeans(X)
data <- cbind(data.frame(outcome=y,race=g,eta0=eta0),as.data.frame(X))
fmla <- outcome ~ offset(eta0) + V1 + V2 + V3 + V4 + V5
fitm <- harsm(fmla,data,group=race)

# with weights
data <- cbind(data.frame(outcome=y,race=g,eta0=eta0),as.data.frame(X))
data$wts <- runif(nrow(data),min=1,max=2)
fmla <- outcome ~ offset(eta0) + V1 + V2 + V3 + V4 + V5
fitm <- harsm(fmla,data,group=race,weights=wts)

# softmax on the Best Picture data
data(best_picture)
df <- best_picture
df$place <- ifelse(df$winner,1,2)
df$weight <- ifelse(df$winner,1,0)

fmla <- place ~ nominated_for_BestDirector + nominated_for_BestActor + Drama

harsm(fmla,data=df,group=year,weights=weight)
```

```
# test against logistic regression
if (require(dplyr)) {
nevent <- 10000
set.seed(1234)
adf <- data_frame(eventnum=floor(seq(1,nevent + 0.7,by=0.5))) %>%
  mutate(x=rnorm(n()),
         program_num=rep(c(1,2),nevent),
         intercept=as.numeric(program_num==1),
         eta=1.5 * x + 0.3 * intercept,
         place=ohenery::rsm(eta,g=eventnum))

# Harville model
modh <- harsm(place ~ intercept + x,data=adf,group=eventnum)

# the collapsed data.frame for glm
ddf <- adf %>%
  arrange(eventnum,program_num) %>%
  group_by(eventnum) %>%
    summarize(resu=as.numeric(first(place)==1),
              delx=first(x) - last(x),
              deli=first(intercept) - last(intercept)) %>%
  ungroup()

# glm logistic fit
modg <- glm(resu ~ delx + 1,data=ddf,family=binomial(link='logit'))

all.equal(as.numeric(coef(modh)),as.numeric(coef(modg)),tolerance=1e-4)
all.equal(as.numeric(vcov(modh)),as.numeric(vcov(modg)),tolerance=1e-4)
}
```

harsmfit *Experts only softmax regression under Harville model.*

Description

An “experts only” softmax fitting function for the Harville model.

Usage

```
harsmfit(y, g, X, wt = NULL, eta0 = NULL, normalize_wt = FALSE,
  method = c("BFGS", "NR", "CG", "NM"))
```

Arguments

y	a vector of the ranked outcomes within each group. Only the order within a group matters.
g	a vector giving the group indices. Need not be integers, but that is more efficient. Need not be sorted. Must be the same length as y.
X	a matrix of the independent variables. Must have as many rows as the length of y.
wt	an optional vector of the observation level weights. These must be non-negative, otherwise an error is thrown. Note that the weight of the last ranked outcome within a group is essentially ignored. Must be the same length as y.
eta0	an optional vector of the consensus odds. These are added to the fit odds in odds space before the likelihood caclulation. If given, then when the model is used to predict, similar consensus odds must be given. Must be the same length as y.
normalize_wt	if TRUE, we renormalize wt, if given, to have mean value 1. Note that the default value has changed since version 0.1.0 of this package. Moreover, non-normalized weights can lead to incorrect inference. Use with caution.
method	maximisation method, currently either "NR" (for Newton-Raphson), "BFGS" (for Broyden-Fletcher-Goldfarb-Shanno), "BFGSR" (for the BFGS algorithm implemented in R), "BHHH" (for Berndt-Hall-Hall-Hausman), "SANN" (for Simulated ANNealing), "CG" (for Conjugate Gradients), or "NM" (for Nelder-Mead). Lower-case letters (such as "nr" for Newton-Raphson) are allowed. If missing, a suitable method is selected automatically.

Details

Given a number of events, indexed by group, and a vector y of the ranks of each entry within that group, perform maximum likelihood estimation under the softmax and proportional probability model.

The user can optionally supply a vector of η_0, which are taken as the fixed, or ‘consensus’ odds. The estimation is then conditional on these fixed odds.

Weighted estimation is supported.

The code relies on the likelihood function of harsmlik, and MLE code from maxLik.

Value

An object of class harsm, maxLik, and linodds.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Harville, D. A. "Assigning probabilities to the outcomes of multi-entry competitions." Journal of the American Statistical Association 68, no. 342 (1973): 312-316. http://dx.doi.org/10.1080/01621459.1973.10482425

See Also

the likelihood function, harsmlik, and the expected rank function (the inverse link), erank.

Examples

```
nfeat <- 5
set.seed(1234)
g <- ceiling(seq(0.1,1000,by=0.1))
X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
beta <- rnorm(nfeat)
eta <- X %*% beta
y <- rsm(eta,g)

mod0 <- harsmfit(y=y,g=g,X=X)
summary(mod0)
# now upweight finishers 1-5
modw <- harsmfit(y=y,g=g,X=X,wt=1 + as.numeric(y < 6))
summary(modw)
```

harsm_invlink *The inverse link for the softmax.*

Description

The inverse link function for the softmax. This function takes the group-wise probabilities, μ, and computes the expected ranks within each group under the Harville model. That is, it is a groupwise computation of the erank function.

Usage

```
harsm_invlink(eta, mu = smax(eta, g), g = NULL)
```

Arguments

eta a vector of the odds. Must be the same length as g if g is given.

mu a vector of the probabilities. Should sum to one, at least per group. Should be the same size as g if given. If both mu and eta are given, a warning is issued, and the mu is used.

g a vector giving the group indices. If NULL, then we assume only one group is in consideration.

Value

a vector of the ranks.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

the ungrouped version of this, erank.

Examples

```
mus <- runif(12)
mus <- mus / sum(mus)
harsm_invlink(mus)

harsm_invlink(mus,c(rep(1,6),rep(2,6)))
```

hensm *Friendly interface to softmax regression under Henery model.*

Description

A user friendly interface to the softmax regression under the Henery model.

Usage

```
hensm(formula, data, group = NULL, weights = NULL, ngamma = 4,
  na.action = na.omit)

## S3 method for class 'hensm'
vcov(object, ...)

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

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. The details of model specification are given under ‘Details’.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
`group`	the string name of the group variable in the data, or a bare character with the group name. The group indices need not be integers, but that is more efficient. They need not be sorted.
`weights`	an optional vector of weights, or the string or bare name of the weights in the `data` for use in the fitting process. The weights are attached to the outcomes, not the participant. Set to `NULL` for none.
`ngamma`	The number of gammas to fit. Should be at least 2.
`na.action`	How to deal with missing values in `y`, `g`, `X`, `wt`, `eta0`.

`object` an object of class `hensm`.

`...` additional arguments to be passed to the low level regression fitting functions (see below).

`x` logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.

Details

Performs a softmax regression by groups, via Maximum Likelihood Estimation. It is assumed that successive sub-races maintain the proportional probability of the softmax, up to some gamma coefficients, $\gamma_2, \gamma_3, ..., \gamma_n$, which we fit. This model nests the Harville model fit by `harsm`, by fixing all the gammas equal to 1.

Value

An object of class `hensm`, but also of `maxLik` with the fit.

Note

This regression may give odd results when the outcomes are tied, imposing an arbitrary order on the tied outcomes. Moreover, no warning may be issued in this case. In future releases, ties may be dealt with differently, perhaps in analogy to how ties are treated in the Cox Proportional Hazards regression, using the methods of Breslow or Efron.

To avoid incorrect inference when only the top performers are recorded, and all others are effectively tied, one should use weighting. Set the weights to zero for participants who are tied non-winners, and one for the rest So for example, if you observe the Gold, Silver, and Bronze medal winners of an Olympic event that had a starting field of 12 participants, set weights to 1 for the medal winners, and 0 for the others. Note that the weights do not attach to the participants, they attach to the place they took.

Since version 0.1.0 of this package, the normalization of weights used in this function have changed under the hood. This is to give correct inference in the case where zero weights are used to signify finishing places were not observed. If in doubt, please confirm inference by simulations, taking as example the simulations in the README.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

`harsm`, `smlik`.

Examples

```
nfeat <- 5
set.seed(1234)
g <- ceiling(seq(0.1,1000,by=0.1))
X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
beta <- rnorm(nfeat)
```

```
eta <- X %*% beta
# 2FIX: do rhenery
y <- rsm(eta,g)
# now the pretty frontend
data <- cbind(data.frame(outcome=y,race=g),as.data.frame(X))

fmla <- outcome ~ V1 + V2 + V3 + V4 + V5
fitm <- hensm(fmla,data,group=race)

# with offset
eta0 <- rowMeans(X)
data <- cbind(data.frame(outcome=y,race=g,eta0=eta0),as.data.frame(X))
fmla <- outcome ~ offset(eta0) + V1 + V2 + V3 + V4 + V5
fitm <- hensm(fmla,data,group=race)

# on horse race data
library(dplyr)
data(race_data)
df <- race_data %>%
 group_by(EventId) %>%
  mutate(eta0=log(WN_pool / sum(WN_pool))) %>%
 ungroup() %>%
 mutate(weights=ifelse(!is.na(Finish),1,0)) %>%
 mutate(fac_age=cut(Age,c(0,3,5,7,Inf),include.lowest=TRUE))

# Henery Model with market efficiency
hensm(Finish ~ eta0,data=df,group=EventId,weights=weights,ngamma=3)

# look for age effect not captured by consensus odds.
hensm(Finish ~ offset(eta0) + fac_age,data=df,group=EventId,weights=weights,ngamma=2)
```

inv_smax *The inverse softmax function.*

Description

The inverse softmax function: take a logarithm and center.

Usage

```
inv_smax(mu, g = NULL)
```

Arguments

mu	a vector of the probablities. Must be the same length as g if g is given. If mu and eta are both given, we ignore eta and use mu.
g	a vector giving the group indices. If NULL, then we assume only one group is in consideration.

Details

This is the inverse of the softmax function. Given vector μ for a single group, finds vector η such that

$$\eta_i = \log \mu_i + c,$$

where c is chosen such that the η sum to zero:

$$c = \frac{-1}{n} \sum_i \log \mu_i.$$

Value

the centered log probabilities.

Note

This function can deal with overflow in a semi-coherent way.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

smax

Examples

```
# we can deal with large values:
set.seed(2345)
eta <- rnorm(12,sd=1000)
mu <- smax(eta)
eta0 <- inv_smax(mu)
```

normalize *Normalize a vector to sum to one.*

Description

Divide a vector by its sum, resulting in a vector with sum equal to one.

Usage

```
normalize(x)
```

Arguments

x	vector of input data.

Value

the input divided by its sum. For the row-wise version, each row is divided by its sum.

Note

This function will return `NA` when any elements of the input are `NA`. May return `Inf` if the elements sum to zero.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

`ohenery` *The 'ohenery' package.*

Description

Modeling of ordinal outcomes via the softmax function under the Harville and Henery models.

Harville and Henery models

The Harville and Henery models describe the probability of ordered outcomes in terms of some parameters. Typically the ordered outcomes are things like place in a race, or winner among a large number of contestants. The Harville model could be described as a softmax probability for the first place finish, with a recursive model on the remaining places. The Henery model generalizes that to adjust the remaining places with another parameter.

These are best illustrated with an example. Suppose you observe a race of 20 contestants. Contestant number 11 takes first place, number 6 takes second place, and 17 takes third place, while the fourth through twentieth places are not recorded or not of interest. Under the Harville model, the probability of this outcome can be expressed as

$$\frac{\mu_{11}}{\sum_i \mu_i} \frac{\mu_6}{\sum_{i \neq 11} \mu_i} \frac{\mu_{17}}{\sum_{i \neq 11, i \neq 6} \mu_i},$$

where $\mu_i = \exp \eta_i$. In a softmax regression under the Harville model, one expresses the odds as $\eta_i = x_i^{\top} \beta$, where x_i are independent variables, for some β to be fit by the regression.

Under the Henery model, one adds gammas, $\gamma_2, \gamma_3, ...$ such that the probability of the outcome above is

$$\frac{\mu_{11}}{\sum_i \mu_i} \frac{\mu_6^{\gamma_2}}{\sum_{i \neq 11} \mu_i^{\gamma_2}} \frac{\mu_{17}^{\gamma_3}}{\sum_{i \neq 11, i \neq 6} \mu_i^{\gamma_3}}.$$

There is no reason to model a γ_1 as anything but one, since it would be redundant. The Henery softmax regression estimates the β as well as the γ_j. To simplify the regression, the higher order gammas are assumed to equal the last fit value. That is, we usually model $\gamma_5 = \gamma_4 = \gamma_3$.

The regression supports weighted estimation as well. The weights are applied to the *places*, not to the participants. The weighted likelihood under the example above, for the Harville model is

$$\left(\frac{\mu_{11}}{\sum_i \mu_i}\right)^{w_1} \left(\frac{\mu_6}{\sum_{i \neq 11} \mu_i}\right)^{w_2} \left(\frac{\mu_{17}}{\sum_{i \neq 11, i \neq 6} \mu_i}\right)^{w_3}.$$

The weighting mechanism is how this package deals with unobserved places. Rather than marking all runners-up as tied for fourth place, in this case one sets the $w_i = 0$ for $i > 3$. The regression is then not asked to make distinctions between the tied runners-up.

Breaking Changes

This package is a work in progress. Expect breaking changes. Please file any bug reports or issues at https://github.com/shabbychef/ohenery/issues.

Legal Mumbo Jumbo

ohenery is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

Note

This package is maintained as a hobby.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

References

Harville, D. A. "Assigning probabilities to the outcomes of multi-entry competitions." Journal of the American Statistical Association 68, no. 342 (1973): 312-316. http://dx.doi.org/10.1080/01621459.1973.10482425

Henery, R. J. "Permutation probabilities as models for horse races." Journal of the Royal Statistical Society: Series B (Methodological) 43, no. 1 (1981): 86-91. http://dx.doi.org/10.1111/j.2517-6161.1981.tb01153.x

---

ohenery-NEWS *News for package 'ohenery':*

Description

News for package 'ohenery'

ohenery Initial Version 0.1.1 (2018-10-14)

- Change default in harsm and hensm to use unnormalized weights, correcting inference when not all finishes are observed.

ohenery Initial Version 0.1.0 (2018-10-01)

- first CRAN release.

race_data *Horse Race Data*

Description

Three weeks of horse race data from tracks worldwide.

Usage

```
data(race_data)
```

Format

A `data.frame` object with 36,418 observations and 19 columns.

The columns are defined as follows:

`EventId` An integer ID denoting the event (race). These range from 1 to 4486.

`TrackId` An integer ID number of the the track. There are 64 different tracks represented.

`Type` The type of event, one of "Thoroughbred" or "Harness".

`RaceNum` The integer race number within a group of races at a track on a given date.

`CorrectedPostTime` The 'corrected' post time of the race, in the form `%Y-%m-%d %H:%M:%S`, presumably in the PDT time zone. Has values like "2019-03-05 02:30:00".

`Yards` The length of the race, in yards.

`SurfaceText` A string, one of "Turf", "Dirt", "All-Weather" or `NA`.

`HorseName` The string name of the horse.

`HorseId` A unique integer ID for each horse. As different horses can have the same name, this ID is constructed from the name of the Horse, the Sire and the Dam.

`Age` The age of the horse, in integer years, at the time of the event. Typically less than 10.

`Sex` A single character denoting the sex of the horse. I believe the codes are "M" for "Mare" (female four years or older), "G" for "Gelding", "F" for "Filly" (female under four years of age), "C" for "Colt" (male under four years of age), "H" for "Horse" (male four years of age and up), "R" for "Rig" (hard to explain), "A" for "???". There are some `NA` values as well.

`Weight_lbs` The weight in integer pounds of the jockey and any equipment. Typically around 120.

`PostPosition` The integer starting position of the horse. Typically there is a slight advantage to starting at the first or second post position.

`Medication` One of several codes indicating any medication the horse may be taking at the time of the race. I believe "L" stands for "Lasix", a common medication for lung conditions that is thought to give horses a slight boost in speed.

`MorningLine` A double indicating the "morning betting line" for win bets on the horse. It is not clear how to interpret this value, perhaps it is return on a dollar. Values range from 0.40 to 80.

`WN_pool` The total combined pool in win bets, in dollars, on this horse at post time.

`PL_pool` The total combined pool in place bets, in dollars, on this horse at post time.

`SH_pool` The total combined pool in show bets, in dollars, on this horse at post time.

`Finish` The integer finishing position of the horse. A 1 means first place. We only collect values of 1, 2, and 3, while the remaining finishing places are unknown and left as `NA`.

Note

The author makes no guarantees regarding correctness of this data.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

Source

Data were sourced from the web. Don't ask.

Examples

```
library(dplyr)
data(race_data)

# compute win bet efficiency
efficiency <- race_data %>%
  group_by(EventId) %>%
    mutate(ImpliedOdds=WN_pool / sum(WN_pool,na.rm=TRUE)) %>%
  ungroup() %>%
 mutate(OddsBucket=cut(ImpliedOdds,c(0,0.05,seq(0.1,1,by=0.10)),include.lowest=TRUE)) %>%
  group_by(OddsBucket) %>%
    summarize(PropWin=mean(as.numeric(coalesce(Finish==1,FALSE)),na.rm=TRUE),
              MedImpl=median(ImpliedOdds,na.rm=TRUE),
              nObs=n()) %>%
  ungroup()


if (require('ggplot2') && require('scales')) {
  efficiency %>%
    ggplot(aes(MedImpl,PropWin,size=nObs)) +
    geom_point() +
    scale_x_sqrt(labels=percent) +
    scale_y_sqrt(labels=percent) +
    geom_abline(slope=1,intercept=0,linetype=2,alpha=0.6) +
    labs(title='actual win probability versus implied win probability',
         size='# horses',
         x='implied win probability',
         y='observed win probability')
 }
```

rhenery *Random generation under the Henery (or Harville) softmax model.*

Description

Given base probabilities, and Henery gamma coefficients, performs random generation, using R's built in rand seed, of the final outcome of a race for each participant.

Usage

```
rhenery(mu, gamma = NULL)
```

Arguments

`mu`	a vector of the probabilities of taking first place.
`gamma`	a vector of the gamma coefficients. Should have length one less than `mu`, but if longer the unused elements are ignored. If shorter, we reserve the right to either throw an error or extend out the last gamma element. If not given, the coefficients are assumed to be all one, which is the Harville model.

Details

Given vectors μ and γ, first computes

$$\pi_{1,i} = \frac{\mu_i^{\gamma_1}}{\sum_j \mu_j^{\gamma_1}},$$

then assigns a 1 to participant i with probability $\pi_{1,i}$. The 'winning' participant is then removed from consideration, and the process is repeated using the remaining μ and γ vectors.

Typically one has that $\mu_i = \exp \eta_i$, for some 'odds', η_i.

When the γ are all one, you recover the Harville softmax model.

Value

A vector, of the same length as the probabilities, giving the entry of each horse. Note that the expected value of this returned thing makes sense, it is *not* the finished rank ordering of a race.

Author(s)

Steven E. Pav `<shabbychef@gmail.com>`

See Also

`rsm`

rsm — *Generate variates from a softmax distribution.*

Description

Generate variates from a softmax distribution under Harville or Henery models.

Usage

```
rsm(eta, g = NULL, mu = NULL, gamma = NULL)
```

Arguments

`eta`	a vector of the odds. Must be the same length as `g` if `g` is given.
`g`	a vector giving the group indices. If `NULL`, then we assume only one group is in consideration.
`mu`	a vector of the probablities. Must be the same length as `g` if `g` is given. If `mu` and `eta` are both given, we ignore `eta` and use `mu`.
`gamma`	a vector of the gamma parameters for the Henery model. Typically the first element should be 1. Omit or set all values to 1 to recover the Harville model. The last element will be extended if the gamma is not long enough for a given group. Note that gamma is expected to be very short, and is not 'aligned' with `eta` in any way.

Details

Given the η in odds space, and a grouping variable, returns values in one to the number of elements in a group. That is, we have a permutation of 1 through n_g on each group as output.

For a single group, the probability that the ith element of the output is a 1 is equal to

$$\pi_{1,i} = \frac{\mu_i^{\gamma_1}}{\sum_j \mu_j^{\gamma_1}}.$$

Once an element has been selected to have output 1, remove it from the set and iterate, but with the next γ elements.

Value

a vector of integers, each a permutation of one through the number of elements in each group.

Note

The output of this function satisfies a kind of order invariance that `rhenery` does not, at the cost of some computational inefficiency. Namely that for a fixed randseed, and for distinct `eta`, the output is equivariant with respect to permutation of the vector `eta` when there is only one group.

Regarding the 'direction', we associate higher odds with a smaller outcome. That is, the ith element of the output encodes the place that the ith participant took in the simulated 'race'; it should be small if the odds for that participant are very high.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

Examples

```
# simple use
set.seed(1234)
g <- ceiling(seq(1,10,by=0.1))
eta <- rnorm(length(g))
y <- rsm(eta,g=g)

# same same:
set.seed(235)
y1 <- rsm(eta,g=g)
set.seed(235)
y2 <- rsm(g=g,mu=smax(eta,g=g))
y1 - y2

# the default model is Harville
set.seed(1212)
y1 <- rsm(eta,g=g)
set.seed(1212)
y2 <- rsm(eta,g=g,gamma=c(1,1,1,1))
y1 - y2

# repeat several times with the cards stack against
# early runners
set.seed(1234)
colMeans(t(replicate(1000,rsm(sort(rnorm(10,sd=1.5)),g=rep(1,10)))))

# illustrate the invariance
mu <- (1:10) / 55
set.seed(1414)
y1 <- rsm(mu=mu,gamma=c(1,1,1))
set.seed(1414)
y2 <- rev(rsm(mu=rev(mu),gamma=c(1,1,1)))
y1 - y2


nfeat <- 5
set.seed(1234)
g <- ceiling(seq(0.1,1000,by=0.1))
X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
beta <- rnorm(nfeat)
eta <- X %*% beta
y <- rsm(eta,g=g)

idx <- order(g,y,decreasing=TRUE) - 1
fooey <- harsmlik(g,idx,eta,deleta=X)
set.seed(3493)
dib <- rnorm(length(beta))
xvl <- seq(-0.01,0.01,length.out=301)
```

```
rsu <- sapply(xvl,
              function(del) {
                beta1 <- beta + del * dib
                eta1 <- X %*% beta1
                fooey2 <- harsmlik(g,idx,eta1,deleta=X)
                as.numeric(fooey2) - as.numeric(fooey)
              })
drv <- sapply(xvl,
              function(del) {
                beta1 <- beta + del * dib
                eta1 <- X %*% beta1
                fooey2 <- harsmlik(g,idx,eta1,deleta=X)
                sum(attr(fooey2,'gradient') * dib)
              })

if (require('ggplot2') && require('dplyr')) {
  bestx <- xvl[which.max(rsu)]
  ph <- data.frame(x=xvl,lik=rsu,grd=drv) %>%
    ggplot(aes(x=x,y=lik)) +
    geom_point() +
    geom_line(aes(y=grd/200)) +
    geom_vline(xintercept=bestx,linetype=2,alpha=0.5) +
    geom_hline(yintercept=0,linetype=2,alpha=0.5)
  print(ph)
}


if (require('dplyr') && require('knitr')) {
  # expect this to be very small, almost always 1
  set.seed(1234)
  simdraw <- replicate(10000,{
    rsm(eta=c(100,rnorm(7)))[1]
  })

  as.data.frame(table(simdraw)) %>%
    mutate(prob=Freq / sum(Freq)) %>%
    knitr::kable()

  # expect this to be uniform on 2 through 8
  set.seed(1234)
  simdraw <- replicate(10000,{
    rsm(eta=c(100,rnorm(7)))[2]
  })

  as.data.frame(table(simdraw)) %>%
    mutate(prob=Freq / sum(Freq)) %>%
    knitr::kable()
}
```

smax *The softmax function.*

Description

The softmax function: exponentiate a vector and then normalize.

Usage

```
smax(eta, g = NULL)
```

Arguments

`eta`	numeric array of the odds. The odds are de-meaned within each group.
`g`	a vector giving the group indices. If `NULL`, then we assume only one group is in consideration.

Details

Given vector η for a single group, essentially computes vector μ defined by

$$\mu_i = \frac{\exp \eta_i}{\sum_j \exp \eta_j}.$$

Note that this computation should be invariant with respect to level shifts of the η, and thus we de-mean the odds first.

Value

the exponentiated data normalized. For the row-wise version, each row is soft maxed.

Note

This function can deal with overflow in a semi-coherent way.

Author(s)

Steven E. Pav <shabbychef@gmail.com>

See Also

normalize, inv_smax.

Examples

```
# we can deal with large values:
set.seed(2345)
eta <- rnorm(12,sd=1000)
smax(eta)
```

`smlik` *Softmax log likelihood under Harville and Henery Models.*

Description

Compute the softmax log likelihood and gradient of the same.

Usage

```
harsmlik(g, idx, eta, wt = NULL, deleta = NULL)

hensmlik(g, idx, eta, gamma, wt = NULL, deleta = NULL)
```

Arguments

`g` a vector giving the group indices.

`idx` a vector of integers, the same length as g, which describes the reverse sort order on the observations, first by group, then by place within the group. That is, the element `idx[0]` is the index of the last place finisher in the group `g[0]`; then `idx[1]` is the index of the next-to-last place finisher in `g[1]` (assuming it equals `g[0]`), and so on. The `idx` shall be zero based.

`eta` a vector of the odds. Must be the same length as g.

`wt` an optional vector of non-negative elements, the same length as g, giving the observation level weights. We then compute a weighted log likelihood, where the weights are in each summand. The weights should probably have mean 1, but that's just, like, my opinion, man. Negative weights throw an error. Note that the weights for the last place in each group have no effect on the computation.

`deleta` an optional matrix whose row count equals the number of elements of `g`, `idx`, and `eta`. The rows of `deleta` are the derivatives of eta with respect to some z. This is used to then maximize likelihood with respect to z.

`gamma` a vector of the gamma parameters. It is assumed that the first element is γ_2, while the last element is applied to all higher order tie breaks.

Details

Given vectors g, η and optionally the gradient of η with respect to some coefficients, computes the log likelihood under the softmax. The user must provide a reverse ordering as well, which is sorted first by the groups, g, and then, within a group, in increasing quality order. For a race, this means that the index is in order from the last place to the first place in that race, where the group numbers correspond to one race.

Under the Harville model, the log likelihood on a given group, where we are indexing in *forward* order, is

$$\left(\eta_1 - \log \sum_{j \ge 1} \mu_j\right) + \left(\eta_2 - \log \sum_{j \ge 2} \mu_j\right) + \ldots$$

where $\mu_i = \exp \eta_i$. By “forward order”, we mean that η_1 corresponds to the participant taking first place within that group, η_2 took second place, and so on.

The Henery model log likelihood takes the form

$$\left(\eta_1 - \log \sum_{j \ge 1} \mu_j\right) + \left(\gamma_2 \eta_2 - \log \sum_{j \ge 2} \mu_j^{\gamma_2}\right) + ...$$

for gamma parameters, γ. The Henery model corresponds to the Harville model where all the gammas equal 1.

Weights in weighted estimation apply to each summand. The weight for the last place participant in a group is irrelevant. The weighted log likelihood under the Harville model is

$$w_1 \left(\eta_1 - \log \sum_{j \ge 1} \mu_j\right) + w_2 \left(\eta_2 - \log \sum_{j \ge 2} \mu_j\right) + ...$$

One should think of the weights as applying to the outcome, not the participant.

Value

The log likelihood. If `deleta` is given, we add an attribute to the scalar number, called `gradient` giving the derivative. For the Henery model we also include a term of `gradgamma` which is the gradient of the log likelihood with respect to the gamma vector.

Note

The likelihood function does not yet support ties.

To avoid incorrect inference when only the top performers are recorded, and all others are effectively tied, one should use weighting. Set the weights to zero for participants who are tied non-winners, and one for the rest So for example, if you observe the Gold, Silver, and Bronze medal winners of an Olympic event that had a starting field of 12 participants, set weights to 1 for the medal winners, and 0 for the others. Note that the weights do not attach to the participants, they attach to the place they took.

Author(s)

Steven E. Pav <`shabbychef@gmail.com`>

References

Harville, D. A. "Assigning probabilities to the outcomes of multi-entry competitions." Journal of the American Statistical Association 68, no. 342 (1973): 312-316. http://dx.doi.org/10.1080/01621459.1973.10482425

Henery, R. J. "Permutation probabilities as models for horse races." Journal of the Royal Statistical Society: Series B (Methodological) 43, no. 1 (1981): 86-91. http://dx.doi.org/10.1111/j.2517-6161.1981.tb01153.x

Examples

```
# a garbage example
set.seed(12345)
g <- as.integer(sort(ceiling(20 * runif(100))))
idx <- as.integer(rev(1:length(g)) - 1L)
eta <- rnorm(length(g))
foo <- harsmlik(g=g,idx=idx,eta=eta,deleta=NULL)

# an example with a real idx
nfeat <- 5
set.seed(1234)
g <- ceiling(seq(0.1,1000,by=0.1))
X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
beta <- rnorm(nfeat)
eta <- X %*% beta
y <- rsm(eta,g)

idx <- order(g,y,decreasing=TRUE) - 1
foores <- harsmlik(g,idx,eta,deleta=X)

# now reweight them
wt <- runif(length(g))
wt <- wt / mean(wt)   # mean 1 is recommended
foores <- harsmlik(g,idx,eta,wt=wt)

# try hensmlik
foores <- hensmlik(g,idx,eta,gamma=c(0.9,0.8,1),wt=wt)

# check the value of the gradient by numerical approximation

 nfeat <- 8
 set.seed(321)
 g <- ceiling(seq(0.1,1000,by=0.1))
 X <- matrix(rnorm(length(g) * nfeat),ncol=nfeat)
 beta <- rnorm(nfeat)
 eta <- X %*% beta
 y <- rsm(eta,g)

 idx <- order(g,y,decreasing=TRUE) - 1
 if (require(numDeriv)) {

  fastval <- attr(harsmlik(g,idx,eta,deleta=X),'gradient')
  numap <- grad(function(beta,g,idx,X) {
     eta <- X %*% beta
     as.numeric(harsmlik(g,idx,eta))
     },
     x=beta,g=g,idx=idx,X=X)
  rbind(fastval,numap)
 }
```

Index