# Package 'nauf'

June 14, 2017

**Type** Package

**Title** Regression with NA Values in Unordered Factors

**Version** 1.1.0

**Description** Fits regressions where unordered factors can be set to NA in subsets of the data where they are not applicable or otherwise not contrastive by using sum contrasts and setting NA values to zero.

**Depends** R (>= 3.3.3), standardize (>= 0.2.1), lme4 (>= 1.1-12), rstanarm (>= 2.15.3)

**Imports** methods, utils, MASS (>= 7.3-45), stringr (>= 1.1.0), Matrix (>= 1.2-7.1), lsmeans (>= 2.25-5), pbkrtest (>= 0.4-7), car (>= 2.1-4), lmerTest (>= 2.0-33), Rcpp (>= 0.12.5), rstan (>= 2.15.1), bayesplot (>= 1.2.0), digest (>= 0.6.12), loo (>= 1.1.0), shinystan (>= 2.3.0)

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**URL** https://github.com/CDEager/nauf

**BugReports** https://github.com/CDEager/nauf/issues

**Suggests** testthat, knitr, rmarkdown, afex

**RoxygenNote** 5.0.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Christopher D. Eager [aut, cre]

**Maintainer** Christopher D. Eager <eagerstats@gmail.com>

**Repository** CRAN

**Date/Publication** 2017-06-14 21:45:28 UTC

# R topics documented:



---

`nauf-package` *Regression with NA values in unordered factors.*

---

## Description

It is often the case that a factor only makes sense in a subset of a dataset (i.e. for some observations a factor may simply not be meaningful), or that with observational datasets there are no observations in some levels of an interaction term. There are also cases where a random effects grouping factor is only applicable in a subset the data, and it is desireable to model the noise introduced by the repeated measures on the group members within the subset of the data where the repeated measures exist. The `nauf` package allows unordered factors and random effects grouping factors to be coded as `NA` in the subsets of the data where they are not applicable or otherwise not contrastive. It is highly recommended that variables be put on the same scale with `standardize` prior to using `nauf` functions (though this is not required).

## Contrasts

A detailed description of how `NA` values are treated is given in `nauf_contrasts`. These contrasts are implemented automatically through `nauf_model.frame`, which stores the information required to make fixed effects and random effects model matrices with `nauf` contrasts in its `terms` attribute. For details on the `terms` attribute, see `nauf.terms`. For fixed effects and random effects model matrices, see `nauf_model.matrix` and `nauf_glFormula`.

### Regressions

nauf contrasts have been implemented for fixed effects regressions that would normally be fit with lm, glm, and glm.nb; and for mixed effects regressions that would normally be fit with lmer, glmer, and glmer.nb. For fixed effects nauf regressions, see nauf_glm, and for mixed effects nauf regressions, see nauf_glmer. There is also support for Bayesian versions of these models as would normally be fit with stan_lm, stan_glm, stan_glm.nb, stan_lmer, stan_glmer, and stan_glmer.nb (see nauf_stan_glm and nauf_stan_glmer for details).

### ANOVAs

The anova function can be used with any frequentist (i.e. not Bayesian) nauf model. For fixed effects nauf models, the anova function uses the methods for corresponding non-nauf models (i.e. anova.lm, anova.glm, or anova.negbin depending on the regression family). For mixed effects nauf models, the anova function uses the anova.nauf.merMod method, which with default arguments is the same as anova.merMod. The anova.nauf.merMod method also allows Type III tests to be made via likelihood ratio tests, parametric bootstrapping, and, for linear models, the Satterthwaite and Kenward-Roger approximations of denominator degrees of freedom (similar to the mixed function in the afex package).

### Predicted Marginal Means

The nauf_ref.grid function can be used to construct reference grids for nauf models (constructing a ref.grid-class object). Predicted marginal means (often called least-squares means; but in the case of Bayesian regression they are posterior marginal means) can be calculated with this reference grid using the nauf_pmmeans function. The function also allows the user to flexibly specify a subset of the reference grid to use when calculating the marginal means, so that the effect of a factor can be tested with regards to the subset of the data where it is contrastive.

### Datasets and Vignette

For detailed examples of how to use the nauf package, see the "Using the nauf package" vignette, which makes use of the plosives and fricatives datasets included in the package.

---

## anova.nauf.merMod *Type III anovas for mixed effects* nauf *models.*

---

### Description

Obtain an anova table for a nauf.lmerMod or nauf.glmerMod model. Currently only Type III tests are supported.

### Usage

```
## S3 method for class 'nauf.lmerMod'
anova(object, ..., refit = TRUE, model.names = NULL,
  method = c("lme4", "S", "KR", "LRT", "PB", "nested-KR"),
  test_intercept = FALSE, args_test = NULL)
```

```
## S3 method for class 'nauf.glmerMod'
anova(object, ..., refit = TRUE, model.names = NULL,
  method = c("lme4", "LRT", "PB"), test_intercept = FALSE,
  args_test = NULL)
```

## Arguments

| | |
|---|---|
| object | A nauf.lmerMod or nauf.glmerMod. |
| ... | Additional nauf models for the lme4 method. See anova.merMod. |
| refit | For the lme4 method, a logical indicating whether nauf.lmerMod models fit with REML should be refit with ML prior to comparison with models in ...; default TRUE. See anova.merMod. |
| model.names | For the lme4 method, character vectors of model names to be used in the anova table. See anova.merMod. |
| method | The method for calculating p-values. See 'Details'. |
| test_intercept | For all methods besides lme4, whether a test should be performed for the intercept term (default FALSE). |
| args_test | For methods nested-KR and PB, an optional named list of arguments to be passed to KRmodcomp and PBmodcomp, respectively. |

## Details

There are six methods of p-value calculation which are supported:

**lme4** The default method. See anova.merMod.

**S** *nauf.lmerMod models only*. Computes F-tests using the Satterthwaite approximation of denominator degrees of freedom, implemented with calcSatterth.

**KR** *nauf.lmerMod models only*. If object was fit with maximum likelihood (ML), then the model is refit with restricted maximum likelihood (REML) first. Then computes F-tests using the Kenward-Roger approximation of denominator degrees of freedom, implemented with Anova.merMod.

**nested-KR** *nauf.lmerMod models only*. If object was fit with maximum likelihood (ML), then the model is refit with restricted maximum likelihood (REML) first. Then for each fixed effects term, a restricted nested model is fit lacking only that fixed effects term, and F-tests are computed using the Kenward-Roger approximation of denominator degrees of freedom, implemented with KRmodcomp. The full model and restricted models are returned along with the anova table, similar to mixed.

**LRT** If object is a nauf.lmerMod fit with REML, it is first refit with ML. Then restricted models are fit as in the nested-KR method, and Chi-squared (likelihood ratio) tests are computed. The full model and restricted models are returned along with the anova table, similar to mixed.

**PB** If object is a nauf.lmerMod fit with REML, it is first refit with ML. Then likelihood ratios are computed as for the LRT method, and p-values for the likelihood ratios are computed using parametric bootstrapping, implemented with PBmodcomp. The full model and restricted models are returned along with the anova table, similar to mixed.

## Value

The object returned depends on the method, and has class `nauf.mer.anova`. For the `lme4`, `S`, and `KR` methods, it is an `anova` table. For the `nested-KR`, `PB`, and `LRT` methods, a list with the anova table and restricted models is returned (similar to the output of `mixed`).

## See Also

`nauf.lmerMod` and `nauf.glmerMod` classes; `anova.merMod` for the `lme4` method; `Anova.merMod` for the `KR` method; `calcSatterth` for the `S` method; `mixed` for the `nested-KR`, `LRT`, and `PB` methods; `KRmodcomp` for the `nested-KR` method; `PBmodcomp` for the `PB` method.

## Examples

```
dat <- droplevels(subset(plosives, voicing == "Voiceless"))
dat$spont[dat$dialect == "Valladolid"] <- NA
sobj <- standardize(cdur ~ dialect * spont + (1 | speaker) + (1 | item), dat)

mod <- nauf_lmer(sobj$formula, sobj$data)

## Not run:
# lme4 method anova table
anova(mod)

# anova table using Satterthwaite approximation
anova(mod, method = "S")

# anova table using Kenward-Roger approximation
anova(mod, method = "KR")

# list with restricted models and Kenward-Roger table
anova(mod, method = "nested-KR")

# list with restricted models and parametric bootstrap table
# model is first refit with maximum likelihood
anova(mod, method = "PB")

# list with restricted models and likelihood ratio test table
# model is first refit with maximum likelihood
anova(mod, method = "LRT")

## End(Not run)
```

---

`as.shinystan,nauf.pmm.list-method` *Create a shinystan object for posterior marginal means from nauf models.*

---

### Description

Joins the samples elements from the nauf.pmm.stan objects in a nauf.pmm.list, adds the log-posterior to them, and passes the resulting array to the array method for as.shinystan.

### Usage

```
## S4 method for signature 'nauf.pmm.list'
as.shinystan(X, ...)
```

### Arguments

| | |
|---|---|
| X | A nauf.pmm.list. |
| ... | Not used. |

### Value

A shinystan object based on the posterior marginal means (and possibly pairwise comparisons) in X. This allows the results to be viewed in the shinystan GUI as if they had been included in the generated quantities block of the model.

### See Also

launch_shinystan

---

## as.shinystan,nauf.stanreg-method

*Create a shinystan object from a nauf.stanreg model.*

---

### Description

This is a simple wrapper function that calls the as.shinystan method for stanreg objects.

### Usage

```
## S4 method for signature 'nauf.stanreg'
as.shinystan(X, ...)
```

### Arguments

| | |
|---|---|
| X | A nauf.stanreg model. |
| ... | See as.shinystan. |

### Value

See as.shinystan.

## fricatives *Catalan and Spanish intervocalic alveolar fricatives.*

### Description

A dataset containing duration and voicing measures for /s/ from 16 speakers of Spanish and for /s/, /z/, and /S/ (the variant that is neutralized in underlying voicing) for 26 speakers of Catalan. The data are from a corpus of map task interviews in Spanish and Catalan and all fricatives are intervocalic. If you analyze the `fricatives` dataset in a publication, please cite Hualde and Prieto (2014) from the references section below.

### Usage

```
fricatives
```

### Format

A data frame with 1622 rows and 6 variables:

**dur** The duration of the fricative, as determiend by the onset and offset of aperiodic energy in the acoustic signal (milliseconds).

**pvoi** The proportion of the fricative which is voiced (measured including 30 milliseconds of each of the surrounding vowels).

**lang** The language of the map task interview (Catalan or Spanish).

**wordpos** Position of the fricative in the word (Final, Initial, or Medial).

**uvoi** The underlying voicing of the fricative (Neutralized, Voiced, or Voiceless).

**speaker** Speaker identifier (s01 through s42).

### References

Hualde, J. I., & Prieto, P. (2014). Lenition of intervocalic alveolar fricatives in Catalan and Spanish. Phonetica, 71(2), 109-127.

## nauf-pmmeans *Predicted marginal means for* `nauf` *models.*

### Description

Create a reference grid for a `nauf` model with `nauf_ref.grid`, and use the resultnig `nauf.ref.grid` as the `object` argument to `nauf_pmmeans` to obtain predicted marginal means and pairwise comparisons, optionally conditioning these predictions on certain subsets of the data via the `subset` argument.

## Usage

```
nauf_ref.grid(mod, KR = FALSE, ...)

nauf_pmmeans(object, specs, pairwise = FALSE, subset = NULL,
  na_as_level = NULL, by = NULL, ...)
```

## Arguments

| | |
|---|---|
| mod | A regression model fit with nauf contrasts. |
| KR | Only applies when mod is a nauf.lmerMod fit with REML = TRUE. If KR = TRUE, then the Kenward-Roger approximation is used to calculate degrees of freedom. If KR = FALSE (the default), then the Satterthwaite approximation is used. When mod is a nauf.lmerMod fit with REML = FALSE, then the Satterthwaite approximation is always used. The Kenward-Roger method is implemented with Lb_ddf and the Satterthwaite method is implemented with calcSatterth. |
| ... | Additional arguments are ignored with a warning. |
| object | A nauf.ref.grid object created with nauf_ref.grid. |
| specs | The fixed effects for which the full interaction term should be considered in the calculation of predicted marginal means. The preferred method is to specify the variables as a character vector. However, they can also be specified on the right hand side of a formula, optionally with the keyword pairwise on the left hand side to indicate that pairwise comparions should be performed. |
| pairwise | A logical (default FALSE) indicating whether pairwise comparisons of the predicted marginal means should be performed. If specs is a formula, then the pairwise argument is ignored and the left hand side of the formula is used to determined whether pairwise comparsions should be made. If by is not NULL then pairwise is forced to TRUE. |
| subset | A list indicating which subsets of the reference grid should be considered in the calculation of the predicted marginal means. See 'Details'. |
| na_as_level | A character vector of unordered factors in specs that have NA values that should be considered as levels. The default NULL indicates that NA should not be considered as a level for any unordered factors in specs. See 'Details'. |
| by | An optional character vector specifying unordered factors in specs. If specified, then pairwise comparisons are performed within each level of the full interaction term of the factors, rather than for all possible combinations. If an unordered factor listed in by is not included in specs, it is added to specs automatically. |

## Details

A reference grid creates a data frame which contains all possible combinations of the factors in a regression model, holding all covariates at their mean values. There are many options for ref.grid which are not currently supported for nauf models. The main functionality which is not currently supported is that the reference grid cannot be created specifying certain levels for variables (i.e. the at argument; this is handled through the subset argument to nauf_pmmeans). A direct call to ref.grid will result in warnings (or possibly errors), and inference made with the resulting object will be misleading and/or incorrect. Only nauf_ref.grid should be used. The nauf.ref.grid

returned by nauf_ref.grid can then be used as the object argument to nauf_pmmeans to obtain predicted marginal means and pairwise comparisons with p-values that adjust for familywise error rate.

The specs and pairwise arguments to nauf_pmmeans indicate what variables marginal means should be calculated for and wheter pairwise comparisons of these means should be made. If specs is a character vector, then pairwise is used; if specs is a formula, then the full iteraction of the terms on the right hand side of the formula is considered, and the left hand side is used to indicate pairwise comparisons. For example (where rg is a nauf.ref.grid):

```
# all of these calculate pmm's for each combination of the factors f1 and f2
# but not pairwise comparisons
nauf_pmmeans(rg, c("f1", "f2"))
nauf_pmmeans(rg, ~ f1 + f2)
nauf_pmmeans(rg, ~ f1 * f2)
nauf_pmmeans(rg, ~ f1:f2)

# all of these calculate the same pmm's, and additionally pairwise comparions
nauf_pmmeans(rg, c("f1", "f2"), pairwise = TRUE)
nauf_pmmeans(rg, pairwise ~ f1 + f2)
nauf_pmmeans(rg, pairwise ~ f1 * f2)
nauf_pmmeans(rg, pairwise ~ f1:f2)
```

If specs indicates a single covariate, the effect of an increase of 1 in the covariate is computed. If specs indicates multiple covariates, the effect of a simultaneous increase of 1 in all of the covariates is computed. If specs indicates a combination of factors and covariate(s), the the effect of an increase of 1 for the covariates is calcualted for each level of the full interaction of the factors.

If by is specified, then pairwise is forced to TRUE, and pairwise comparisons are performed within each level of the full interaction of the factors listed in by, rather than performing all possible pairwise comparisons. For example, if there are two factors f1 with levels A, B, and C, and a factor f2 with levels D and E:

```
# this will produce six pmmeans (A:D, A:E, B:D, B:E, C:D, C:E) and
# all 15 pairwise comparisons
nauf_pmmeans(rg, c("f1", "f2"), pairwise = TRUE)

# this would produce the same six pmmeans, but only three pairwise
# comparisons (A:D - A:E, B:D - B:E, C:D - C:E)
nauf_pmmeans(rg, c("f1", "f2"), by = "f1")

# this would produce the same six pmmeans, but only six pairwise comparisons
# (A:D - B:D, A:D - C:D, B:D - C:D, A:E - B:E, A:E - C:E, B:E - C:E)
nauf_pmmeans(rg, c("f1", "f2"), by = "f2")
```

The reference grid returned by nauf_ref.grid contains combinations of factors which are not actually possible in the data set. For example, if factor f1 has levels A and B, and factor f2 is NA when f1 = A, and takes values C and D when f1 = B, the reference grid will still contain the combinations f1 = A, f2 = C; f1 = A, f2 = D; and f1 = B, f2 = NA, even though these combinations are not possible. This is because it is impossible to know without the user's knowledge

which combinations make sense. In many cases, this is inconsequential for the computation of predicted marginal means, since the coding of unordered factors in nauf regressions will average over the effects. In cases where these rows in the reference grid will cause invalid estimates and pairwise comparisons, the subset argument can be used in the call to nauf_pmmeans to ensure only the correct subsets are considered. The default for the subset argument is NULL, indicating that the the entire reference grid should be considered. If not NULL, then subset must be a list which defines the valid subsets as lists of named character vectors, where the name of the character vector is an unordered factor in the model, and the vector itself contains the levels which define the subset (including NA in the case of factors which have NA values; when NA is specified as a level, there should be no quotes around it). Any row in the reference grid which matches the definition of at least one of the groups defined in subset is kept, and all others are dropped. So, continuing with the f1 and f2 example, if f2 = NA corresponds to f2 = D in meaning, and is coded as NA because all f1 = A observations are by necessity f2 = D, then to analyze the effect of f1, we want to compare the groups f1 = A, f2 = NA and f1 = B, f2 = D, which we could do with the following call:

```
nauf_pmmeans(rg, "f1", subset = list(
  list(f1 = "A", f2 = NA), list(f1 = "B", f2 = "D")))
```

This would produce an estimate for f1 = A and f1 = B, but conditioning on the subset where f1 is truly contrastive based on f2. If, on the other hand, f2 = NA does not correspond in interpretation to either f2 = C or f2 = D, but rather indicates that f2 is simply not meaningful when f1 = A, we would want to average over the effect of f2 within f1 = B, and compare this result to f1 = A, f2 = NA, which we could do with the following call:

```
nauf_pmmeans(rg, "f1", subset = list(
  list(f1 = "A", f2 = NA), list(f1 = "B", f2 = c("C", "D"))))
```

In this case, the second sub-list in the subset list indicates that if f1 = B and either f2 = C or f2 = D, then it belongs to the second subset. In this case, the subset argument is actually not necessary, since for f1 = A, we want to *not consider* the effect f2, and for f2 = B, we want to *average over all possible levels* of f2, and these are actually the same thing computationally for unordered factors in nauf models. That is, we would get the same result with:

```
nauf_pmmeans(rg, "f1")
```

Generally speaking, if all of the factors in specs do *not* contain NA values, then the subset argument is unnecessary. If any of the factors in specs *do* contain NA values, then you will almost always want to use the subset argument. Now consider that we are interested in f2. Because f2 is only contrastive when f1 = B, we probably want to call:

```
# note that because there are not multiple subsets being specified, you
# don't have to specify subset = list(list(f1 = "B")); nauf_pmmeans will
# assume list(f1 = "B") means list(list(f1 = "B"))
nauf_pmmeans(rg, "f2", subset = list(f1 = "B"))
```

This call will produce two estimates, one for f2 = C and one for f2 = D, conditioning on f1 = B. There will be no estimate for f2 = NA because, by default, no estiamtes are produced for combinations of factors where one factor is NA. If we wanted to compare the three possible groups (i.e.

f1 = A, f2 = NA; f1 = B, f2 = C; and f1 = B, f2 = D), then we could additionally use the na_as_level argument and change our subset:

```
nauf_pmmeans(rg, "f2", subset = list(
  list(f1 = "A", f2 = NA), list(f1 = "B", f2 = c("C", "D"))),
  na_as_level = "f2")

# this gives the same estimates, but the output will also show the
# corresponding level of f1, which is more transparent
nauf_pmmeans(rg, c("f1", "f2"), subset = list(
  list(f1 = "A", f2 = NA), list(f1 = "B", f2 = c("C", "D"))),
  na_as_level = "f2")
```

The easiest way to use the subset argument is to create a list that defines valid subsets for different regression terms of interest outside of nauf_pmmeans, and then using the relevant element of the list in the nauf_pmmeans call. For example:

```
pmmsubs <- list()
pmmsubs$f1 <- list(list(f1 = "A", f2 = NA), list(f1 = "B", f2 = c("C", "D")))
pmmsubs$f2 <- list(f1 = "B")

nauf_pmmeans(rg, "f1", subset = pmmsubs$f1)

nauf_pmmeans(rg, "f2", subset = pmmsubs$f2)

nauf_pmmeans(rg, c("f1", "f2"), subset = pmmsubs$f1, na_as_level = "f2")
```

This way you can just define the different subsets of the data once and not have to think about it at every nauf_pmmeans call.

## Value

nauf_ref.grid returns a nauf.ref.grid object, which is just a list with one element ref.grid of class ref.grid-class. This reference grid should not be used directly with lsmeans, but rather only with nauf_pmmeans. nauf_pmmeans returns a nauf.pmm.list object.

## See Also

nauf_contrasts, nauf_glm, nauf_glmer, nauf_stan_glm, and nauf_stan_glmer.

---

# nauf.merMod *Class for fitted mixed effects models with* nauf *contrasts.*

---

## Description

Models fit with nauf_lmer have class nauf.lmerMod (inheriting from lmerMod) and models fit with nauf_glmer and nauf_glmer.nb have class nauf.glmerMod (inheriting from glmerMod).

### Slots

rsp,Gp,call,frame,flist,cnms,lower,theta,beta,u,devcomp,pp,optinfo See merMod.

### Generic Methods

There are S3 methods specific to the nauf.lmerMod and nauf.glmerMod classes for the generic functions predict and anova, which behave differently from the methods for merMod objects. Other methods for generic functions from the stats, MASS, and lme4 packages should work as they would for merMod objects, with the exception of the method for the simulate function, for which the nauf method has more limited options than simulate.merMod, and is not meant for end user use at this time (the method is necessary for the PB method to work for anova). If you encounter a generic function in these packages which does not function properly, please report the issue at https://github.com/CDEager/nauf/issues.

### See Also

nauf_glmer, nauf_contrasts, and merMod.

---

## nauf.pmm.list *List of predicted marginal means objects for* nauf *models.*

---

### Description

The nauf_pmmeans function returns an object of class nauf.pmm.list.

### Details

The nauf.pmm.list object contains a first element pmmeans which contains the predicted marginal means, and possibly additional contrasts elements depending on whether the call to nauf_pmmeans indicated that pairwise comparisons should be made. If the by argument was specified, there will be a numbered contrasts element for each level of the full interaction of the factors listed in the by argument. The object also has a specs attribute which contains information about the variables and subsets from the call to the function.

The object has summary and print methods which first print information from the specs attribute, and then pass addtional function arguments along to each element in the list. For frequentist regressions, the elements in the nauf.pmm.list are lsmobj-class objects. In this case, the summary and print methods for the nauf.pmm.list take arguments listed in summary.ref.grid (e.g. infer, type, adjust, etc.). For Bayesian regressions, the elements in the nauf.pmm.list are nauf.pmm.stan objects. In this case, the summary and print methods for the nauf.pmm.list take arguments which are listed in nauf.pmm.stan.

For posterior marginal means from a Bayesian nauf model, there is a method for as.shinystan which allows launch_shinystan to be used.

nauf.pmm.stan *Posterior samples of marginal means from Bayesian* nauf *models.*

## Description

When nauf_pmmeans is used with Bayesian regressions, the elements of the resulting nauf.pmm.list have class nauf.pmm.stan.

## Usage

```
## S3 method for class 'nauf.pmm.stan'
summary(object, probs = c(0.025, 0.975),
  type = c("link", "response"), ...)

## S3 method for class 'summ.nauf.pmm.stan'
print(x, row.names = FALSE, mcse = FALSE,
  rhat = NULL, ...)

## S3 method for class 'nauf.pmm.stan'
as.array(x, ...)

## S3 method for class 'nauf.pmm.stan'
as.matrix(x, ...)

## S3 method for class 'nauf.pmm.stan'
print(x, ...)

## S3 method for class 'nauf.pmm.stan'
as.data.frame(x, row.names = NULL, optional = FALSE,
  ...)

## S3 method for class 'summ.nauf.pmm.stan'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | A nauf.pmm.stan object. |
| probs | A vector of quantiles to calculate for the estimates. The default is a 95 The median (0.5) is always added regardless of whether it is specified. |
| type | If "link" (the default), then the estimates are not transformed prior to summary; if "response", then the inverse link function in the object's family element is applied to the samples element prior to summary. |
| ... | See the Methods section. |
| x | Either a nauf.pmm.stan object, or the summ.nauf.pmm.stan object returned by calling summary on a nauf.pmm.stan object. |

`row.names, optional`
: Changes from the defaults are ignored.

`mcse`
: A logical indicating whether the Monte Carlo standard errors should be printed (default `FALSE` since the column is always recoverable by dividing the `SD` column by the square root of the `ESS` column).

`rhat`
: An optional logical indicating whether or not to print the Gelman-Rubin R-hat statistic. If `rhat = NULL` (the default), then the `Rhat` column of the summary is only printed if any of the statistics are greater that `1.1`. Regardless of the value of the `rhat` argument, a warning is issued if any of the statistics are greater than `1.1`.

## Details

The `nauf.pmm.stan` object is a list with the following elements.

**names** A data frame with the levels of each factor (or 'inc_1' for covariates), with one row for each element in the third dimension of `samples`.

**contrasts** The fixed effects model matrix showing the contrasts applied to the regression coefficients (rows correspond to the `names` element).

**samples** An array with three dimensions. The first corresponds to iterations, the second to chains, and the third to parameters (the same structure returned by `as.array.stanfit`).

**family** The regression family.

**inv.lbl** The label of the inverse link (e.g. probability, rate, etc.).

**misc** A list with additional information based on the type of model.

## Value

The returned object depends on the function. See the 'Methods' section.

## Methods

**summary** Returns a data frame with class `summ.nauf.pmm.stan`, with means, standard deviations, medians, mean absolute differences, the posterior probability that the estimate is greater than zero, quantiles specified in `probs`, the effective sample size, Monte Carlo standard error, and Gelman-Rubin R-hat statistic. Additional arguments in `...` are ignored. Quantiles, effective samples size, and Monte Carlo standard error, and R-hat statistics are computed using `monitor`.

**print** For `summ.nauf.pmm.stan` objects, the data frame is printed, omitting the Monte Carlo standard error and R-hat statistic based on `mcse` and `rhat`. If the inverse link function was used, then a message indicating the response type is also printed. Additional arguments in `...` are passed to `print.data.frame`. For `nauf.pmm.stan` objects, first `summary` is called, passing along `...`, and then print is called on the resulting `summ.nauf.pmm.stan`, also passing along `...` arguments.

**as.data.frame** For `summ.nauf.pmm.stan` objects, removes the `data.frame` contained in a `summ.nauf.pmm.stan`. Additional arguments in `...` are ignored. For `nauf.pmm.stan` objects, first `summary` is called, passing along `...`, and then `as.data.frame` is called on the resulting `summ.nauf.pmm.stan` object, ignoring `...` arugments.

**as.array** Returns the samples element of a nauf.pmm.stan object. Additional arguments in ... are ignored.

**as.matrix** Returns the samples element of a namf.pmm.stan object, flattening the array into a matrix such that the rows represent iterations ordered by chain and the columns represent parameters.

---

nauf.stanreg *Class for fitted Bayesian models with* nauf *contrasts.*

---

## Description

Models fit with nauf_stan_lm, nauf_stan_glm, nauf_stan_glm.nb, nauf_stan_lmer, nauf_stan_glmer, and nauf_stan_glmer.nb have class nauf.stanreg and inherit from class stanreg (see stanreg-objects for details on the elements contained in the fitted model object). The stanreg methods for the generic functions listed in the stanreg-methods page (including those linked to in the 'See Also' section) work for nauf.stanreg models, with the same restrictions on the re.form argument described in the predict.nauf.merMod page when using the posterior_predict and predict functions. The only exception is that the kfold function from the rstanarm package cannot be used on nauf.stanreg objects; instead, nauf_kfold should be used. The nauf_ref.grid and nauf_pmmeans functions also work with nauf.stanreg objects, as do as.shinystan and launch_shinystan.

## See Also

nauf_stan_glm, nauf_stan_glmer, nauf_contrasts, stanreg-objects, and stanreg-methods.

---

nauf.terms *Class for* terms *objects which contain information about* nauf *contrasts.*

---

## Description

When nauf_model.frame is called, a nauf.frame is returned, and this object's terms attribute has the (S3) class nauf.terms. The nauf.terms object has an attribute nauf.info which contains all of the information necessary to implement nauf_contrasts in regression.

## Details

The nauf.info attribute is a list with the following elements:

**formula** The formula argument to nauf_model.frame with double-bars expanded.

**resp** The name of the response variable.

**groups** A named list of random effects grouping factor levels.

**uf** A named list with an elment for each unordered factor. Each of these elements is a list of character vectors indicating the names of the levels that correspond to the elment's number's set of contrasts; i.e. the first element represents the main effect contrasts for the factor, the second element (if present) represents `.c2.` contrasts, and so on (see `nauf_contrasts`).

**of** A named list with an element for each ordered factor containing its levels and contrasts.

**num** A named list with an element for each numeric vector variable containing the variables' means.

**mat** A named list with an element for each matrix variable containing the variables' colmun means.

**extras** A character vector giving the names of offsets, weights, mustart, etc.

**cc** A list of contrast changes required as described in `nauf_contrasts`. The first element is a list of the changes required for the fixed effects. If the model has random effects, then there is an additional element for each element of the list returned by `findbars`. Each element of `cc` is a named list indicating which contrasts in the `uf` element of `nauf.info` should be used.

**hasna** A named logical vector with an entry for each variable indicating whether nor not the variable contains `NA` values.

**ncs_scale** The `ncs_scale` argument from the call to `nauf_model.frame`.

## See Also

`nauf_contrasts` and `nauf_model.frame`.

---

`nauf_contrasts` *Not applicable unordered factor contrasts.*

---

## Description

The `nauf_contrasts` function returns a list of contrasts applied to factors in an object created using a function in the `nauf` package. See 'Details'.

## Usage

```
nauf_contrasts(object, inc_ordered = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | A `nauf.terms` object, a model frame made with `nauf_model.frame`, a `nauf.glm` model (see `nauf_glm`), or a `nauf.lmerMod` or `nauf.glmerMod` model. |
| `inc_ordered` | A logical indicating whether or not ordered factor contrasts should also be returned (default `FALSE`). |

## Details

In the nauf package, NA values are used to encode when an unordered factor is truly *not applicable*. This is different than "not available" or "missing at random". The concept applies only to unordered factors, and indicates that the factor is simply not meaningful for an observation, or that while the observation may technically be definable by one of the factor levels, the interpretation of its belonging to that level isn't the same as for other observations. For imbalanced observational data, coding unordered factors as NA may also be used to control for a factor that is only contrastive within a subset of the data due to the sampling scheme. To understand the output of the nauf_contrasts function, the treatment of unordered factor contrasts in the nauf package will first be discussed, using the plosives dataset included in the package as an example.

In the plosives dataset, the factor ling is coded as either Monolingual, indicating the observation is from a monolingual speaker of Spanish, or Bilingual, indicating the observation is from a Spanish-Quechua bilingual speaker. The dialect factor indicates the city the speaker is from (one of Cuzco, Lima, or Valladolid). The Cuzco dialect has both monolingual and bilingual speakers, but the Lima and Valladolid dialects have only monolingual speakers. In the case of Valladolid, the dialect is not in contact with Quechua, and so being monolingual in Valladolid does not mean the same thing as it does in Cuzco, where it indicates *monolingual as opposed to bilingual*. Lima has Spanish-Quechua bilingual speakers, but the research questions the dataset serves to answer are specific to monolingual speakers of Spanish in Lima. If we leave the ling factor coded as is in the dataset and use named_contr_sum to create the contrasts, we obtain the following:

| dialect | ling | dialectCuzco | dialectLima | lingBilingual |
|---|---|---|---|---|
| Cuzco | Bilingual | 1 | 0 | 1 |
| Cuzco | Monolingual | 1 | 0 | -1 |
| Lima | Monolingual | 0 | 1 | -1 |
| Valladolid | Monolingual | -1 | -1 | -1 |

With these contrasts, the regression coefficient dialectLima would not represent the difference between the intercept and the mean of the Lima dialect; the mean of the Lima dialect would be the (Intercept) + dialectLima - lingBilingual. The interpretation of the lingBilingual coefficient is similarly affected, and the intercept term averages over the predicted value for the non-existent groups of Lima bilingual speakers and Valladolid bilingual speakers, losing the interpretation as the corrected mean (insofar as there can be a corrected mean in this type of imbalanced data). With the nauf package, we can instead code non-Cuzco speakers' observations as NA for the ling factor (i.e. execute plosives$ling[plosives$dialect != "Cuzco"] <- NA). These NA values are allowed to pass into the regression's model matrix, and are then set to 0, effectively creating the following contrasts:

| dialect | ling | dialectCuzco | dialectLima | lingBilingual |
|---|---|---|---|---|
| Cuzco | Bilingual | 1 | 0 | 1 |
| Cuzco | Monolingual | 1 | 0 | -1 |
| Lima | NA | 0 | 1 | 0 |
| Valladolid | NA | -1 | -1 | 0 |

Because sum contrasts are used, a value of 0 for a dummy variable averages over the effect of the factor, and the coefficient lingBilingual only affects the predicted value for observations where dialect = Cuzco. In a regression fit with these contrasts, the coefficient dialectLima

represents what it should, namely the difference between the intercept and the mean of the Lima dialect, and the intercept is again the corrected mean. The `lingBilingual` coefficient is now the difference between Cuzco bilingual speakers and the corrected mean *of the Cuzco dialect*, which is `(Intercept) + dialectCuzco`. These `nauf` contrasts thus allow us to model all of the data in a single model without sacrificing the interpretability of the results. In sociolinguistics, this method is called *slashing* due to the use of a forward slash in GoldVarb to indicate that a factor is not applicable.

This same methodology can be applied to other parts of the `plosives` dataset where a factor's interpretation is the same for all observations, but is only contrastive within a subset of the data due to the sampling scheme. The `age` and `ed` factors (speaker age group and education level, respectively) are factors which can apply to speakers regardless of their dialect, but in the dataset they are only contrastive within the Cuzco dialect; all the Lima and Valladolid speakers are 40 years old or younger with a university education (in the case of Valladolid, the data come from an already-existing corpus; and in the case of Lima, the data were collected as part of the same dataset as the Cuzco data, but as a smaller control group). These factors can be treated just as the `ling` factor by setting them to `NA` for observations from Lima and Valladolid speakers. Similarly, there is no read speech data for the Valladolid speakers, and so `spont` could be coded as `NA` for observations from Valladolid speakers.

Using `NA` values can also allow the inclusion of a random effects structure which only applies to a subset of the data. The `plosives` dataset has data from both read (`spont = FALSE`; only Cuzco and Lima) and spontaneous (`spont = TRUE`; all three dialects) speech. For the read speech, there are exactly repeated measures on 54 items, as indicated by the `item` factor. For the spontaneous speech, there are not exactly repeated measures, and so in this subset, `item` is coded as `NA`. In a regression fit using `nauf_lmer`, `nauf_glmer`, or `nauf_glmer.nb` with `item` as a grouping factor, the random effects model matrix is created for the read speech just as it normally is, and for spontaneous speech observations all of the columns are set to `0` so that the `item` effects only affect the fitted values for read speech observations. In this way, the noise introduced by the read speech items can be accounted for while still including all of the data in one model, and the same random effects for `speaker` can apply to all observations (both read and spontaneous), which will lead to a more accurate estimation of the fixed, speaker, and item effects since more information is available than if the read and spontaneous speech were analyzed in separate models.

There are two situations in which unordered factors will need more than one set of contrasts: (1) when an unordered factor with `NA` values interacts with another unordered factor, and some levels are collinear with `NA`; and (2) when an unordered factor is included as a slope for a random effects grouping factor that has `NA` values, but only a subset of the levels for the slope factor occur when the grouping factor is not `NA`. As an example of an interaction requiring new contrasts, consider the interaction `dialect * spont` (that is, suppose we are interested in whether the effect of `spont` is different for Cuzco and Lima). We code `spont` as `NA` when `dialect = Valladolid`, as mentioned above. This gives the following contrasts for the main effects:

| dialect | spont | dialectCuzco | dialectLima | spontTRUE |
|---|---|---|---|---|
| Cuzco | TRUE | 1 | 0 | 1 |
| Cuzco | FALSE | 1 | 0 | -1 |
| Lima | TRUE | 0 | 1 | 1 |
| Lima | FALSE | 0 | 1 | -1 |
| Valladolid | NA | -1 | -1 | 0 |

If we simply multiply these `dialect` and `spont` main effect contrasts together to obtain the contrasts for the interaction (which is what is done in the default `model.matrix` method), we get following contrasts:

| dialect | spont | dialectCuzco:spontTRUE | dialectLima:spontTRUE |
|---|---|---|---|
| Cuzco | TRUE | 1 | 0 |
| Cuzco | FALSE | -1 | 0 |
| Lima | TRUE | 0 | 1 |
| Lima | FALSE | 0 | -1 |
| Valladolid | NA | 0 | 0 |

However, these contrasts introduce an unnecessary parameter to the model which causes collinearity with the main effects since `spontTRUE = dialectCuzco:spontTRUE + dialectLima:spontTRUE` in all cases. The functions in the `nauf` package automatically recognize when this occurs, and create a second set of contrasts for `dialect` in which the `Valladolid` level is treated as if it were `NA` (through and additional call to `named_contr_sum`):

| dialect | dialect.c2.Cuzco |
|---|---|
| Cuzco | 1 |
| Lima | -1 |
| Valladolid | 0 |

This second set of `dialect` contrasts is only used when it needs to be. That is, in this case, these contrasts would be used in the creation of the model matrix columns for the interaction term `dialect:spont` term, but not in the creation of the model matrix columns for the main effect terms `dialect` and `spont`, and when the second set of contrasts is used, `.c2.` will appear between the name of the factor and the level so it can be easily identified:

| dialect | spont | dialectCuzco | dialectLima | spontTRUE | dialect.c2.Cuzco:spontTRUE |
|---|---|---|---|---|---|
| Cuzco | TRUE | 1 | 0 | 1 | 1 |
| Cuzco | FALSE | 1 | 0 | -1 | -1 |
| Lima | TRUE | 0 | 1 | 1 | -1 |
| Lima | FALSE | 0 | 1 | -1 | 1 |
| Valladolid | NA | -1 | -1 | 0 | 0 |

Turning now to an example of when a random slope requires new contrasts, consider a random `item` slope for `dialect`. Because `dialect = Valladolid` only when `item` is `NA`, using the main effect contrasts for `dialect` for the `item` slope would result in collinearity with the `item` intercept in the random effects model matrix:

| dialect | item | i01:(Intercept) | i01:dialectCuzco | i01:dialectLima |
|---|---|---|---|---|
| Cuzco | i01 | 1 | 1 | 0 |
| Cuzco | i02 | 0 | 0 | 0 |
| Cuzco | NA | 0 | 0 | 0 |
| Lima | i01 | 1 | 0 | 1 |
| Lima | i02 | 0 | 0 | 0 |
| Lima | NA | 0 | 0 | 0 |
| Valladolid | NA | 0 | 0 | 0 |

This table shows the random effects model matrix for `item i01` for all possible scenarios, with the rows corresponding to (in order): a Cuzco speaker producing the read speech plosive in `item i01`, a Cuzco speaker producing a read speech plosive in another `item`, a Cuzco speaker producing a spontaneous speech plosive, a Lima speaker producing the read speech plosive in `item i01`, a Lima speaker producing a read speech plosive in another `item`, a Lima speaker producing a spontaneous speech plosive, and a Valladolid speaker producing a spontaneous speech plosive. With the main effect contrasts for `dialect`, `i01:(Intercept) = i01:dialectCuzco + i01:dialectLima` in all cases, causing collinearity. Because this collinearity exists for all read speech item random effects model matrices, the model is unidentifiable. The functions in the `nauf` package automatically detect that this is the case, and remedy the situation by creating a new set of contrasts used for the `item` slope for `dialect`:

| dialect | item | i01:(Intercept) | i01:dialect.c2.Cuzco |
|---|---|---|---|
| Cuzco | i01 | 1 | 1 |
| Cuzco | i02 | 0 | 0 |
| Cuzco | NA | 0 | 0 |
| Lima | i01 | 1 | -1 |
| Lima | i02 | 0 | 0 |
| Lima | NA | 0 | 0 |
| Valladolid | NA | 0 | 0 |

If we were to, say, fit the model `intdiff ~ dialect * spont + (1 + dialect | item)`, then `nauf` would additionally recognize that the same set of altered contrasts for `dialect` are required in the fixed effects interaction term `dialect:spont` and the `item` slope for `dialect`, and both would be labeled with `.c2.`. In other (rare) cases, more than two sets of contrasts may be required for a factor, in which case they would have `.c3.`, `.c4.` and so on.

In this way, users only need to code unordered factors as `NA` in the subsets of the data where they are not contrastive, and `nauf` handles the rest. Having described in detail what `nauf` contrasts are, we now return to the `nauf_contrasts` function. The function can be used on objects of any `nauf` model, a `nauf.terms` object, or a model frame made by `nauf_model.frame`. It returns a named list with a matrix for each unordered factor in `object` which contains all contrasts associated the factor. For the model `intdiff ~ dialect * spont + (1 + dialect | item)`, the result would be a list with elements `dialect` and `spont` that contain the following matrices (see the 'Examples' section for code to generate this list):

| dialect | Cuzco | Lima | .c2.Cuzco |
|---|---|---|---|
| Cuzco | 1 | 0 | 1 |
| Lima | 0 | 1 | -1 |
| Valladolid | -1 | -1 | 0 |

| spont | TRUE |
|---|---|
| TRUE | 1 |
| FALSE | -1 |
| NA | 0 |

The default is for the list of contrasts to only contain information about unordered factors. If `inc_ordered = TRUE`, then the contrast matrices for any ordered factors in `object` are also in-

cluded.

## Value

A named list of contrasts for all unordered factors in `object`, and also optionally contrasts for ordered factors in `object`. See 'Details'.

## Note

The argument `ncs_scale` changes what value is used for the sum contrast deviations. The default value of `1` would give the contrast matrices in 'Details'. A value of `ncs_scale = 0.5`, for example, would result in replacing `1` with `0.5` and `-1` with `-0.5` in all of the contrast matrices.

## See Also

`nauf_model.frame`, `nauf_model.matrix`, `nauf_glFormula`, `nauf_glm`, and `nauf_glmer`.

## Examples

```
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA

mf <- nauf_model.frame(intdiff ~ dialect * spont + (1 + dialect | item), dat)
nauf_contrasts(mf)

mf <- nauf_model.frame(intdiff ~ dialect * spont + (1 + dialect | item),
  dat, ncs_scale = 0.5)
nauf_contrasts(mf)
```

---

# nauf_glFormula *Create a model frame and fixed and random effects model matrices using* nauf *contrasts.*

---

## Description

The same as the `lme4` `modular` functions `glFormula` and `lFormula`, but implementing `nauf_contrasts`. `nauf_lFormula` is used for linear mixed effects regressions (i.e. those that would be fit with `nauf_lmer`) and `nauf_glFormula` is used for genarlized linear mixed effects regressions (i.e. those that would be fit with `nauf_glmer` or `nauf_glmer.nb`). Both of the functions contain a call to `nauf_mkReTrms`, which serves the same purpose as the `lme4` function `mkReTrms`, but with `nauf_contrasts`, and, while `mkReTrms` is exported by `lme4`, `nauf_mkReTrms` is an internal function in the `nauf` package.

## Usage

```
nauf_glFormula(formula, data = NULL, family = gaussian, subset, weights,
  na.action = na.pass, offset, contrasts = NULL, mustart, etastart,
  control = lme4::glmerControl(), ncs_scale = attr(formula,
  "standardized.scale"), ...)

nauf_lFormula(formula, data = NULL, REML = TRUE, subset, weights,
  na.action = na.pass, offset, contrasts = NULL,
  control = lme4::lmerControl(), ncs_scale = attr(formula,
  "standardized.scale"), ...)
```

## Arguments

`formula, data, family, REML, subset, weights, offset, control, mustart, etastart, ...`
: See `glFormula`.

`na.action, contrasts`
: Changes from default values are ignored. See `nauf_model.frame`.

`ncs_scale`
: A positive number to be passed as the `scale` argument to `named_contr_sum` for all unordered factors. See `nauf_model.frame`.

## Value

A list with the following elements:

**fr** The model frame (with class `nauf.frame`). See `nauf_model.frame`.

**X** The fixed effects model matrix with `nauf_contrasts` applied. See `nauf_model.matrix`.

**reTrms** A list containing the random effects model matrix and other information about the random effects structure. The elements of the list have the same structure as that returned by `mkReTrms`, but incorporating `nauf_contrasts`.

**REML** (`nauf_lFormula` only): A logical indicating if restricted maximum likelihood was used (copy of argument).

**family** (`nauf_glFormula` only): The regression family (copy of argument).

**formula** The `formula` argument, but with any double vertical bars expanded (e.g. `(1 + x || subj)` becomes `(1 | subj) + (0 + x | subj)`).

**wmsgs** Warning messages (if any).

## See Also

`nauf_contrasts` for a description of the contrasts applied to unordered factors; `nauf_model.frame` and `nauf_model.matrix` for the creation of the `fr` and `X` elements of the returned list, respectively; and `nauf_lmer`, `nauf_glmer.nb`, and `nauf_glmer` for fitting mixed effects regressions with gaussian, negative binomial, and all other families, respectively.

## Examples

```
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
dat_form <- intdiff ~ voicing * dialect * spont +
  (1 + voicing * spont | speaker) + (1 + dialect | item)
sobj <- standardize(dat_form, dat)
lmod <- nauf_lFormula(sobj$formula, sobj$data)

vless <- droplevels(subset(dat, voicing == "Voiceless"))
vless$fully_voiced <- vless$vdur == 0
vless_form <- fully_voiced ~ dialect * spont +
  (1 + spont | speaker) + (1 + dialect | item)
svless <- standardize(vless_form, vless, family = binomial)
glmod <- nauf_glFormula(svless$formula, svless$data, family = binomial)
```

---

# nauf_glm *Fit a fixed effects regression using* nauf *contrasts.*

---

## Description

The fixed effects regression functions nauf_lm, nauf_glm.nb, and nauf_glm fit linear, negative binomial, and other generalized linear models, respectively, impelementing nauf_contrasts.

## Usage

```
nauf_glm(formula, family = gaussian, data = NULL, weights, subset,
  na.action = na.pass, start = NULL, etastart, mustart, offset,
  control = list(...), model = TRUE, method = "glm.fit", x = TRUE,
  y = TRUE, contrasts = NULL, ncs_scale = attr(formula,
  "standardized.scale"), ...)

nauf_lm(formula, data = NULL, subset, weights, na.action = na.pass,
  method = "qr", model = TRUE, x = TRUE, y = TRUE, qr = TRUE,
  singular.ok = TRUE, contrasts = NULL, offset, ncs_scale = attr(formula,
  "standardized.scale"), ...)

nauf_glm.nb(formula, data = NULL, weights, subset, na.action = na.pass,
  start = NULL, etastart, mustart, control = stats::glm.control(...),
  method = "glm.fit", model = TRUE, x = TRUE, y = TRUE,
  contrasts = NULL, ncs_scale = attr(formula, "standardized.scale"), ...,
  init.theta, link = log)
```

## Arguments

formula, data, subset, offset, weights, method, control, start, etastart, mustart, qr, singular.ok, See lm, glm, and glm.nb.

`na.action, contrasts`
: Changes to the default values for these arguments are ignored with a warning. See `nauf_model.frame`.

`model, x, y`
: Chages to the default values for these arguments are ignored with a warning. See 'Details'.

`ncs_scale`
: A positive number to be passed as the `scale` argument to `named_contr_sum` for all unordered factors. See `nauf_model.frame`.

## Details

`nauf_lm` is based on `lm` in the `stats` package, `nauf_glm` on `glm` in the `stats` package, and `nauf_glm.nb` on `glm.nb` in the `MASS` package; but implementing `nauf_contrasts`. The `nauf` functions have all the same arguments as the functions they are based on, but additionally `ncs_scale`, which is passed to `nauf_model.frame`. Other than `ncs_scale`, the arguments have the same functions as they do in the functions they are based on. The default values for `na.action`, `contrasts`, `model`, `x`, and `y` cannot be changed. For `na.action` and `contrasts`, see `nauf_model.frame`. Forcing `model`, `x`, and `y` to be `TRUE` ensures that the fitted model retains the model frame, model matrix, and response, respectively. This is necessary for some generic functions applied to the fitted model to work properly.

## Value

A fitted model with class `nauf.glm` (inheriting from `lm`, and also possibly `glm` and/or `negbin` depending on the regression). For the elements contained in the object, and additional class attributes the model may contain, see `lm`, `glm`, and `glm.nb`.

## Note on Generics

Methods for S3 generic functions from the `stats` and `MASS` packages should work for `nauf.glm` models as they normally would for the related regression models not fit with `nauf_contrasts`. If you encounter a generic function in these packages which does not function properly, please report the issue at `https://github.com/CDEager/nauf/issues`.

## See Also

`nauf_contrasts` for a description of the contrasts applied to unordered factors; and `lm`, `glm`, and `glm.nb` for argument definitions.

## Examples

```
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
sobj <- standardize(intdiff ~ voicing * dialect * spont, dat)
mod <- nauf_lm(sobj$formula, sobj$data)
```

nauf_glmer *Fit a mixed effects regression using* nauf *contrasts.*

## Description

The mixed effects regression functions nauf_lmer, nauf_glmer.nb, and nauf_glmer fit linear, negative binomial, and other generalized linear mixed effects models, respectively, impelementing nauf_contrasts.

## Usage

```
nauf_glmer(formula, data = NULL, family = gaussian,
  control = lme4::glmerControl(), start = NULL, verbose = 0L, nAGQ = 1L,
  subset, weights, na.action = na.pass, offset, contrasts = NULL, mustart,
  etastart, devFunOnly = FALSE, ncs_scale = attr(formula,
  "standardized.scale"), ...)

nauf_lmer(formula, data = NULL, REML = TRUE,
  control = lme4::lmerControl(), start = NULL, verbose = 0L, subset,
  weights, na.action = na.pass, offset, contrasts = NULL,
  devFunOnly = FALSE, ncs_scale = attr(formula, "standardized.scale"), ...)

nauf_glmer.nb(..., interval = log(th) + c(-3, 3), tol = 5e-05,
  verbose = FALSE, nb.control = NULL, initCtrl = list(limit = 20, eps = 2
  * tol, trace = verbose))
```

## Arguments

formula, data, subset, weights, offset, control, start, devFunOnly, verbose, REML, family, etastart
: See lmer, glmer, and glmer.nb. Note that many arguments that are passed to nauf_glmer are passed to nauf_glmer.nb through ..., including ncs_scale.

na.action, contrasts
: Changes to the default values for these arguments are ignored with a warning. See nauf_model.frame.

ncs_scale
: A positive number to be passed as the scale argument to named_contr_sum for all unordered factors. See nauf_model.frame. For nauf_glmer.nb, ncs_scale is passed through ....

## Details

nauf_lmer, nauf_glmer, and nauf_glmer.nb are based on the lme4 functions lmer, glmer, and glmer.nb, respectively, but implement nauf_contrasts. The nauf functions have all the same arguments as the functions they are based on, but additionally ncs_scale, which is passed to nauf_model.frame. Other than ncs_scale, the arguments have the same functions as they do in the functions they are based on. The default values for na.action and contrasts cannot be changed (see nauf_model.frame).

## Value

A fitted model of class nauf.lmerMod (nauf_lmer) or nauf.glmerMod (nauf_glmer and nauf_glmer.nb).

## See Also

nauf_contrasts for a description of the contrasts applied to unordered factors; and lmer, glmer, and glmer.nb for argument definitions.

## Examples

```
## Not run:
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
sobj <- standardize(intdiff ~ voicing * dialect * spont +
  (1 + voicing * spont | speaker) + (1 + dialect | item), dat)
mod <- nauf_lmer(sobj$formula, sobj$data)

## End(Not run)
```

---

# nauf_kfold *Cross-validation for* nauf.stanreg *models.*

---

## Description

The same as kfold, but ensuring the use of nauf_contrasts.

## Usage

```
nauf_kfold(x, K = 10, save_fits = FALSE)
```

## Arguments

x, K, save_fits
: See kfold.

## Value

An object with classes kfold and loo.

nauf_model.frame *Create a model frame using* nauf *contrasts.*

## Description

nauf_model.frame creates a model frame which employs nauf_contrasts for unordered factors.

## Usage

```
nauf_model.frame(formula, data = NULL, subset = NULL, na.action = na.pass,
  drop.unused.levels = TRUE, xlev = NULL, contrasts = NULL,
  ncs_scale = attr(formula, "standardized.scale"), ...)
```

## Arguments

formula, data, subset, ...
: See model.frame.

na.action, drop.unused.levels, xlev, contrasts
: Changes from default values for these arguments are ignored with a warning.

ncs_scale
: A positive number passed as the scale argument to named_contr_sum for all unordered factor contrasts. The default is to first check whether formula comes from a standardized object returned by standardize. If it is, then the scale argument from the standardize call is used. If it is not, then ncs_scale is set to 1. The value for ncs_scale can also be set explicitly. If it is set explicitly and formula is from a standardized object with a different scale than the explicitly set value, then the explicitly set value is used and a warning is issued.

## Details

First, the default method for model.frame is called. Then any variable in the resulting model frame that is not an unordered factor but has only two unique non-NA values is coerced to an unordered factor. Unordered factors are then assigned contrasts with named_contr_sum, passing ncs_scale as the function's scale argument. Then, necessary contrast changes in interaction terms and random effects slopes are determined as described in nauf_contrasts.

The recommended usage is to first standardize the regression variables, and then use the formula and data elements in the resulting standardized object as arguments to nauf_model.frame. When this is done, ncs_scale is obtained from the standardized.scale attribute of the formula, unless ncs_scale is specified as a value which does not match the standardized scale, in which case the explicitly specified ncs_scale argument is used with a warning. If standardize is not used prior to calling nauf_model.frame, then ncs_scale defaults to 1 unless explicitly specified in the function call, in which case the specified value is used.

Changes from the following default values are ignored with a warning:

**na.action = na.pass** This default value is required in order for NA values to be treated as defined in nauf_contrasts.

**drop.unused.levels = TRUE** This default value is set because `nauf_model.frame` assumes that `data` is not new data. To create a `nauf.frame` with new data, the `terms` attribute of an already existing `nauf.frame` (which has class `nauf.terms`) can be used as the `formula` argument to `model.frame`.

**xlev = NULL** This default is necessary for the same reasons as the default value for `drop.unused.levels`.

**contrasts = NULL** For unordered factors, contrasts are automatically created with `named_contr_sum`, as sum contrasts are necessary to implement `nauf_contrasts`. To specify custom contrasts for ordered factors, the custom contrasts should be explicitly assigned to the ordered factor in `data` (this is automatically done if `standardize` is used first as recommended).

## Value

A model frame with second class attribute `nauf.frame`. Its `formula` attribute has class `nauf.formula` and its `terms` attribute has class `nauf.terms`.

## See Also

`nauf_contrasts` for a description of the contrasts applied to unordered factors, `nauf_model.matrix` for obtaining a fixed effects model matrix, and `nauf_glFormula` for obtaining both fixed effects and random effects model matrices.

## Examples

```
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
form <- intdiff ~ voicing * dialect * spont +
  (1 + voicing * spont | speaker) + (1 + dialect | item)

## default behavior when standardize is not used
# defaults to ncs_scale = 1
mf <- nauf_model.frame(form, dat)

# uses specified ncs_scale = 0.5
mf_0.5 <- nauf_model.frame(form, dat, ncs_scale = 0.5)

## standardize first (recommended use)
sobj <- standardize(form, dat)
sobj_0.5 <- standardize(form, dat, scale = 0.5)

# uses ncs_scale = 1 from attr(sobj$formula, "standardized.scale")
mf_sobj <- nauf_model.frame(sobj$formula, sobj$data)

# uses ncs_scale = 0.5 from attr(sobj_0.5$formula, "standardized.scale")
mf_sobj_0.5 <- nauf_model.frame(sobj_0.5$formula, sobj_0.5$data)

## Not run:
## not recommended
# uses specified ncs_scale = 0.5 and issues a warning since
# attr(sobj$formula, "standardized.scale") = 1
mf_warning <- nauf_model.frame(sobj$formula, sobj$data, ncs_scale = 0.5)
```

```
## End(Not run)
```

---

`nauf_model.matrix` *Create a fixed effects model matrix using* `nauf` *contrasts.*

---

## Description

`nauf_model.matrix` creates a model matrix which employs `nauf_contrasts` for unordered factors.

## Usage

```
nauf_model.matrix(object = NULL, data = NULL, ...)
```

## Arguments

| | |
|---|---|
| `object` | A `nauf.frame` or a regression formula. See 'Details'. |
| `data` | A `nauf.frame` or a `data.frame` containing the variables in `object` if `object` is a regression formula. See 'Details'. |
| `...` | Further arguments to be passed to `nauf_model.frame` when `object` is a regression formula and `data` is a `data.frame`. See 'Details'. |

## Details

Exactly what happens depends on the values of `object` and `data`. The following possibilities are evaluated in the order listed:

**object is a nauf.frame** All arguments besides `object` are ignored, and the information in `object` is used to create the model matrix.

**data is a nauf.frame** All arguments besides `data` are ignored, and the information in `data` is used to create the model matrix.

**object is a formula and data is a data.frame** `nauf_model.frame` is called with `formula = object` and `data = data`, passing along any additional arguments in `...` (including `ncs_scale`). Then the model matrix is created using the information in the resulting `nauf.frame`.

**any other argument values** An error is returned.

## Value

A fixed effects model matrix that implements `nauf_contrasts`. Unlike the default `model.matrix` method, the model matrix does not have a `contrasts` attribute, since multiple sets of contrasts may be required for some unordered factors.

## See Also

`nauf_contrasts` for a description of the contrasts applied to unordered factors, `nauf_model.frame` for creating a model frame with `nauf` contrasts, and `nauf_glFormula` for obtaining both fixed effects and random effects model matrices.

## Examples

```
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
form <- intdiff ~ voicing * dialect * spont +
  (1 + voicing * spont | speaker) + (1 + dialect | item)
sobj <- standardize(form, dat)
mf <- nauf_model.frame(sobj$formula, sobj$data)

## the following all result in the same model matrix
mm1 <- nauf_model.matrix(mf)
mm2 <- nauf_model.matrix(form, mf)  # 'form' ignored
mm3 <- nauf_model.matrix(sobj$formula, sobj$data)
```

---

nauf_stan_glm *Fit a Bayesian fixed effects regression with* nauf *contrasts.*

---

## Description

The Bayesian fixed effects regression functions nauf_stan_lm, nauf_stan_glm.nb, and nauf_stan_glm fit linear, negative binomial, and other generalized linear models, respectively, impelementing nauf_contrasts.

## Usage

```
nauf_stan_glm(formula, family = gaussian(), data = NULL, weights, subset,
  na.action = na.pass, offset = NULL, model = TRUE, x = TRUE,
  y = TRUE, contrasts = NULL, ncs_scale = attr(formula,
  "standardized.scale"), ..., prior = rstanarm::normal(),
  prior_intercept = rstanarm::normal(), prior_aux = rstanarm::cauchy(0, 5),
  prior_PD = FALSE, algorithm = "sampling", adapt_delta = NULL,
  QR = FALSE, sparse = FALSE)

nauf_stan_lm(formula, data = NULL, subset, weights, na.action = na.pass,
  model = TRUE, x = TRUE, y = TRUE, singular.ok = TRUE,
  contrasts = NULL, offset, ncs_scale = attr(formula, "standardized.scale"),
  ..., prior = rstanarm::R2(stop("'location' must be specified")),
  prior_intercept = NULL, prior_PD = FALSE, algorithm = "sampling",
  adapt_delta = NULL)

nauf_stan_glm.nb(formula, data = NULL, weights, subset, na.action = na.pass,
  offset = NULL, model = TRUE, x = TRUE, y = TRUE, contrasts = NULL,
  link = "log", ncs_scale = attr(formula, "standardized.scale"), ...,
  prior = rstanarm::normal(), prior_intercept = rstanarm::normal(),
  prior_aux = rstanarm::cauchy(0, 5), prior_PD = FALSE,
  algorithm = "sampling", adapt_delta = NULL, QR = FALSE)
```

## Arguments

| | |
|---|---|
| formula, data, family, subset, weights, na.action, offset, contrasts, model, x, y, ncs_scale | See nauf_model.frame. |
| ... | Further arguments to be passed to sampling. See stan_glmer for details. |
| algorithm | Changes from the default "sampling" result in an error. Only MCMC is currently supported. |
| singular.ok | See stan_lm. |
| link, prior, prior_intercept, prior_aux, prior_PD, adapt_delta, QR, sparse | See stan_glm and stan_lm. |

## Details

nauf_stan_lm, nauf_stan_glm, and nauf_stan_glm.nb are based on the rstanarm functions stan_lm, stan_glm, and stan_glm.nb, respectively, but implement nauf_contrasts. The nauf functions have all the same arguments as the functions they are based on, but additionally ncs_scale, which is passed to nauf_model.frame. Other than ncs_scale, the arguments have the same functions as they do in the functions they are based on. The default values for na.action, contrasts, model, x, and y cannot be changed. For na.action and contrasts, see nauf_model.frame. Forcing model, x, and y to be TRUE ensures that the fitted model retains the model frame, model matrix, and response, respectively. This is necessary for some generic functions applied to the fitted model to work properly. The default priors for the nauf Bayesian fixed effects regression functions are the defaults from rstanarm version 2.15.3; if you have a later version of the rstanarm package, then the default priors for the nauf regression fitting functions may be different from the rstanarm defaults.

## Value

A nauf.stanreg object.

## See Also

nauf_contrasts for a description of the treatment of NA values, stan_glm for a description of the priors, and the documentation for Stan and the rstan and rstanarm packages for algorithmic details.

## Examples

```
## Not run:
dat <- fricatives
dat$uvoi[!(dat$lang == "Catalan" & dat$wordpos == "Medial")] <- NA
sobj <- standardize(dur ~ lang * wordpos + uvoi, dat)
mod <- nauf_stan_lm(sobj$formula, sobj$data, prior = R2(location = 0.5))

## End(Not run)
```

nauf_stan_glmer *Fit a Bayesian mixed effects regression with* nauf *contrasts.*

## Description

The Bayesian mixed effects regression functions nauf_stan_lmer, nauf_stan_glmer.nb, and nauf_stan_glmer fit linear, negative binomial, and other generalized linear mixed effects models, respectively, impelementing nauf_contrasts.

## Usage

```
nauf_stan_glmer(formula, data = NULL, family = gaussian, subset, weights,
  na.action = na.pass, offset, contrasts = NULL, ncs_scale = attr(formula,
  "standardized.scale"), ..., prior = rstanarm::normal(),
  prior_intercept = rstanarm::normal(), prior_aux = rstanarm::cauchy(0, 5),
  prior_covariance = rstanarm::decov(), prior_PD = FALSE,
  algorithm = "sampling", adapt_delta = NULL, QR = FALSE,
  sparse = FALSE)

nauf_stan_lmer(formula, data = NULL, subset, weights, na.action = na.pass,
  offset, contrasts = NULL, ncs_scale = attr(formula, "standardized.scale"),
  ..., prior = rstanarm::normal(), prior_intercept = rstanarm::normal(),
  prior_aux = rstanarm::cauchy(0, 5), prior_covariance = rstanarm::decov(),
  prior_PD = FALSE, algorithm = "sampling", adapt_delta = NULL,
  QR = FALSE)

nauf_stan_glmer.nb(formula, data = NULL, subset, weights,
  na.action = na.pass, offset, contrasts = NULL, link = "log",
  ncs_scale = attr(formula, "standardized.scale"), ...,
  prior = rstanarm::normal(), prior_intercept = rstanarm::normal(),
  prior_aux = rstanarm::cauchy(0, 5), prior_covariance = rstanarm::decov(),
  prior_PD = FALSE, algorithm = "sampling", adapt_delta = NULL,
  QR = FALSE)
```

## Arguments

formula, data, family, subset, weights, na.action, offset, contrasts, ncs_scale
: See nauf_model.frame and nauf_glFormula.

...
: Further arguments to be passed to sampling. See stan_glmer for details.

algorithm
: Changes from the default "sampling" result in an error. Only MCMC is currently supported.

link, prior, prior_intercept, prior_aux, prior_covariance, prior_PD, adapt_delta, QR, sparse
: See stan_glmer.

## Details

nauf_stan_lmer, nauf_stan_glmer, and nauf_stan_glmer.nb are based on the rstanarm functions stan_lmer, stan_glmer, and stan_glmer.nb, respectively, but implement nauf_contrasts. The nauf functions have all the same arguments as the functions they are based on, but additionally ncs_scale, which is passed to nauf_model.frame. Other than ncs_scale, the arguments have the same functions as they do in the functions they are based on. The default values for na.action and contrasts cannot be changed (see nauf_model.frame). The default priors for the nauf Bayesian mixed effects regression functions are the defaults from rstanarm version 2.15.3; if you have a later version of the rstanarm package, then the default priors for the nauf regression fitting functions may be different from the rstanarm defaults.

## Value

A nauf.stanreg object.

## See Also

nauf_contrasts for a description of the treatment of NA values, stan_glmer for a description of the priors, and the documentation for Stan and the rstan and rstanarm packages for algorithmic details.

## Examples

```
## Not run:
sobj <- standardize(vdur ~ place + stress + spont +
  (1 + place + stress + spont | speaker) + (1 | item),
  subset(plosives, dialect == "Cuzco" & voicing == "Voiceless"))

mod <- nauf_stan_lmer(sobj$formula, sobj$data,
  prior = normal(0, 1, autoscale = FALSE),
  prior_intercept = normal(0, 1, autoscale = FALSE),
  prior_aux = normal(0, 1, autoscale = FALSE),
  prior_covariance = decov(2, 1.5, 2, 0.25)
)

## End(Not run)
```

---

# plosives *Spanish intervocalic plosives.*

---

## Description

A dataset containing measures of plosive strength for instances of intervocalic Spanish /p/, /t/, /k/, /b/, /d/ and /g/. The data are taken from read speech and informal interviews of 30 speakers in Cuzco, Peru and 8 speakers in Lima, Peru; and from 18 speakers from Valladolid, Spain in the task dialogues in the Spanish portion of the Glissando Corpus (Garrido et al. 2013). If you analyze the plosives dataset in a publication, please cite Eager (2017) from the references section below.

## Usage

```
plosives
```

## Format

A data frame with 5281 rows and 21 variables:

**cdur** Total plosive duration, measured from preceding vowel intensity maximum to following vowel intensity maximum, in milliseconds. Set to `0` for elided plosives.

**vdur** Duration of the period of voicelessness in the vowel-consonant-vowel sequence in milliseconds. Set to `0` for fully voiced plosives and elided plosives.

**vpct** Percentage of the consonant duration which was voiceless. For non-elided plosives, `vpct = vdur / cdur`, and for elided plosives, `vpct = 0`.

**intdiff** The maximum intensity in the following vowel minus the minimum intensity in the plosive, in decibels. Set to `0` for elided plosives.

**intvel** The maximum rising velocity of the intensity contour between the consonant minimum intensity and following vowel maximum intensity, in decibels per millisecond. Set to `0` for elided plosives.

**voicing** The underlying voicing of the plosive (Voiced or Voiceless).

**place** Place of articulation (Bilabial, Dental, or Velar).

**stress** Syllabic stress context (Tonic, Post-Tonic, or Unstressed).

**prevowel** Preceding vowel phoneme identity (a, e, i, o, or u).

**posvowel** Following vowel phoneme identity (a, e, i, o, or u).

**wordpos** Position of the plosive in the word (Initial or Medial).

**wordfreq** Number of times the word containing the plosive occurs in the CREA corpus (Real Academia Espanola).

**speechrate** Local speech rate around the consonant in nuclei per second (nuclei located using De Jong and Wempe's (2009) script).

**spont** Whether the speech was spontaneous (TRUE) or read (FALSE).

**dialect** The city where the speaker is from (Cuzco, Lima, or Valladolid).

**sex** The speaker's sex (Female or Male).

**age** The speaker's age group (Older or Younger) based on whether they were over 40 years old, or 40 years old or younger at the time of recording.

**ed** The speaker's highest level of education (Secondary or University).

**ling** The speaker's language background (Bilingual or Monolingual) based on whether they spoke only Spanish, or both Spanish and Quechua.

**speaker** Speaker identifier (s01 through s56).

**item** Read speech item identifier (i01 through i54). Set to `NA` for spontaneous speech.

## Note

The ptk dataset in the standardize package is a subset of the plosives dataset, but with the speakers renumbered:

```
d <- droplevels(subset(plosives,
  dialect == "Valladolid" & voicing == "Voiceless"))

levels(d$speaker)  # s39 to s56
levels(ptk$speaker)  # s01 to s18

levels(d$speaker) <- levels(ptk$speaker)
d <- d[, colnames(ptk)]
rownames(d) <- NULL

all.equal(d, ptk)  # TRUE
```

## References

Eager, Christopher D. (2017). Contrast preservation and constraints on individual phonetic variation. Doctoral thesis. University of Illinois at Urbana-Champaign.

Garrido, J. M., Escudero, D., Aguilar, L., Cardenoso, V., Rodero, E., de-la-Mota, C., ... Bonafonte, A. (2013). Glissando: a corpus for multidisciplinary prosodic studies in Spanish and Catalan. Language Resources and Evaluation, 47(4), 945-971.

Real Academia Espanola. Corpus de referencia del espanol actual (CREA). Banco de Datos. http://www.rae.es

De Jong, N. H., & Wempe, T. (2009). Praat script to detect syllable nuclei and measure speech rate automatically. Behavior Research Methods, 41(2), 385-390.

---

# predict.nauf.merMod *Predictions from a mixed effects* nauf *model at new data values.*

---

## Description

The predict method for nauf.glmerMod and nauf.lmerMod objects (the results of nauf_glmer, nauf_glmer.nb, and nauf_glmer). It is based on predict.merMod, but currently some options are not supported for nauf models.

## Usage

```
## S3 method for class 'nauf.glmerMod'
predict(object, newdata = NULL, newparams = NULL,
  re.form = NULL, ReForm, REForm, REform, terms = NULL, type = c("link",
  "response"), allow.new.levels = FALSE, na.action = na.pass, ...)

## S3 method for class 'nauf.lmerMod'
```

```
predict(object, newdata = NULL, newparams = NULL,
  re.form = NULL, ReForm, REForm, REform, terms = NULL, type = c("link",
  "response"), allow.new.levels = FALSE, na.action = na.pass, ...)
```

## Arguments

| | |
|---|---|
| object | A nauf.lmerMod or nauf.glmerMod. |
| newdata | A data frame to make predictions on. |
| newparams, terms, allow.new.levels | Changes to default values are not currently supported and result in an error. |
| re.form | Formula for random effects to condition on. Currently, only NULL (the default, indicating conditioning on *all* random effects in the model) and NA or ~ 0 (indicating to use only the fixed effects in the predictions) are supported (i.e. you cannot currently condition on a subset of the random effects). |
| ReForm, REForm, REform | Older versions of re.form in lme4 which are now deprecated. |
| type | Whether the predictions should be transformed with the inverse link function. |
| na.action | Changes from default of na.pass are ignored with a warning. |
| ... | Additional parameters (currently unused and ignored with a warning). |

## Value

A numeric vector of predicted values.

## See Also

predict.merMod, nauf_lmer, nauf_glmer, nauf_glmer.nb, nauf.lmerMod, and nauf.glmerMod.

## Examples

```
## Not run:
dat <- plosives
dat$spont[dat$dialect == "Valladolid"] <- NA
sobj <- standardize(intdiff ~ voicing * dialect * spont +
  (1 + voicing * spont | speaker) + (1 + dialect | item), dat)

mod <- nauf_lmer(sobj$formula, sobj$data)
fit <- predict(mod)  # fitted values
preds <- predict(mod, sobj$data)  # predict same data using all ranef
preds_fe <- predict(mod, sobj$data, re.form = NA)  # only use fixef

isTRUE(all.equal(fit, preds))  # TRUE
isTRUE(all.equal(preds, preds_fe))  # FALSE

## End(Not run)
```

# Index