

# Package ‘amer’

February 14, 2012

**Type** Package

**Title** Additive mixed models with lme4

**Version** 0.6.10

**Date** 2011-04-06

**Author** Fabian Scheipl <fabian.scheipl@stat.uni-muenchen.de>

**Maintainer** Fabian Scheipl <fabian.scheipl@stat.uni-muenchen.de>

**Depends** methods, nlme, Matrix, splines, lme4

**Suggests** mlmRev, SASmixed, mgcv, ggplot2

**Description** Fitting generalized additive mixed models based on the mixed model algorithm of lme4

**License** GPL

**LazyLoad** yes

**Collate** ‘amer-class.R’ ‘amer.R’ ‘amerSetup.R’ ‘expand.call.R’ ‘expandBasis.R’ ‘expandMf.R’ ‘getF.R’ ‘indsF.R’ ‘plotF.R’ ‘set.mfrow.R’

**Repository** CRAN

**Date/Publication** 2011-04-06 18:13:22

## R topics documented:

amer . . . . .	2
amer-class . . . . .	3
bsp . . . . .	4
dog . . . . .	5
expand.call . . . . .	5
getF . . . . .	6
plotF . . . . .	7
predict . . . . .	8
rent . . . . .	9

reparameterizeDesign . . . . .	9
tp . . . . .	10
tpU . . . . .	11
tpU2d . . . . .	12
union . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

amer	<i>Fit a (generalized) additive mixed model using lmer..</i>
------	--

---

## Description

Fit a (generalized) additive mixed model using lmer

## Usage

```
amer(formula, data, family=NULL, REML=TRUE, control=list(), start=NULL,
      verbose=FALSE, weights, offset, contrasts=NULL, basisGenerators=c("tp", "tpU",
      "bsp"))
```

## Arguments

formula	a two-sided formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. The terms can include specifications for non-grouped random effects of types given in <code>basisGenerators</code> , see examples. The vertical bar character " " separates an expression for a model matrix and a grouping factor.
data	a data frame containing all the variables occurring in the formula.
family	a GLM family, see <code>glm</code> and <code>family</code> . If <code>family</code> is missing then a linear mixed model is fit; otherwise a generalized linear mixed model is fit.
REML	logical argument to <code>lmer</code> only. Should the estimates be chosen to optimize the REML criterion (as opposed to the log-likelihood)? Defaults to TRUE. Makes no sense for non-Gaussian response.
control	a list of control parameters for <code>lmer</code>
start	a named list of starting values for the parameters in the model. See <code>lmer</code> .
verbose	logical scalar. If TRUE verbose output is generated during the optimization of the parameter estimates.
weights	see <code>lmer</code>
offset	see <code>lmer</code>
contrasts	see <code>lmer</code>
basisGenerators	a character vector of names of functions that generate bases for function estimation in a way lamer can use. See <code>tp</code> for an example.

**Value**

An object of class [amer-class](#).

**Author(s)**

Fabian Scheipl

**See Also**

`tests/optionsTests.r` and the vignette for examples

---

amer-class

*Additive Mixed Model class*

---

**Description**

The "amer"-class represents additive or generalized additive mixed-effects models. It extends the "mer"-class.

**Objects from the Class**

Objects can be created by calls to the [amer](#) function.

**Slots**

The class "amer" extends [mer-class](#). It has only one additional slot:

**smooths:** a list with one entry for each smooth term in the model. Each entry contains the term in the model formula used to generate the spline basis, with attributes `indGrp`, `indPen`, `indUnpen`, `indConst` giving the respective indices of the variance components, random effects, non-constant fixed effects and constant (i.e. intercept, factor level effects) fixed effects associated with the smooth

**Methods**

see [mer-class](#) for a description of methods for mer-objects

**predict** signature(object = "mer") See [predict](#)

**See Also**

[getF](#) for extracting function values of the smooth terms.

bsp

*Generate a reparameterized b-spline basis...***Description**

Generate a reparameterized b-spline basis

**Usage**

```
bsp(x, k=15, spline.degree=3, diff.ord=2, knots, by,
    allPen=FALSE, varying, diag=FALSE)
```

**Arguments**

x	covariate for the smooth function
k	integer: dimensionality of the basis
spline.degree	integer: degree of B-splines (defaults to cubic)
diff.ord	integer: order of the difference penalty on the un-reparameterized spline coefficients. Defaults to 2, that is, penalized deviations from linearity.
knots	double: vector of (inner & outer) knot locations
by	factor variable: estimate separate functions for each level - this assumes standard treatment contrasts for the supplied factor.
allPen	boolean: if TRUE, make design for group-specific curves with common smoothing parameter: all parameters (including the normally unpenalized basis functions in X) are penalized, every level of "by" has the same amount of smoothing if FALSE, make design for separate curves for each by-level: separate smoothing parameters for every level of "by", unpenalized estimates for the coefficients associated with X
varying	numeric: if not NULL, a varying coefficient model is fit: $f(x, \text{varying}) = f(x) * \text{varying}$
diag	logical: force a diagonal covariance-matrix for the random effects for X if allPen=TRUE?

**Details**

Generate a b-spline basis with equidistant knots in mixed model reparameterization

**Value**

list with entries: "X":  $n \times (\text{diff.ord} - 1)$  design matrix for unpenalized part (without intercept), "Z":  $n \times (k - \text{diff.ord} + 1)$  design matrix for penalized part,

**Author(s)**

Fabian Scheipl

**See Also**[tp](#)


---

dog	<i>Coronary sinus potassium concentration measurements for 36 dogs</i>
-----	--

---

**Description**

The dogs were divided into 4 treatment groups, and the measurements for each dog were taken every two minutes from 1 to 13 minutes after occlusion (i.e. an artificially induced heart attack) Taken from package `assist`, see there for a description.

**Source**

Grizzle, J. E. and Allen, D. M. (1969). Analysis of growth and dose response curves, *Biometrics* 25: 357-381

---

<code>expand.call</code>	<i>returns a call in which all of the arguments which were supplied or have presets are specified by their full names and supplied or default values.</i>
--------------------------	---

---

**Description**

returns a call in which all of the arguments which were supplied or have presets are specified by their full names and supplied or default values.

**Usage**

```
expand.call(definition, call=sys.call(sys.parent(1)),
            expand.dots=TRUE)
```

**Arguments**

<code>definition</code>	a function. See <a href="#">match.call</a> .
<code>call</code>	an unevaluated call to the function specified by <code>definition</code> . See <a href="#">match.call</a> .
<code>expand.dots</code>	logical. Should arguments matching <code>...</code> in the call be included or left as a ... argument? See <a href="#">match.call</a> .

**Value**

An object of class `call`.

**Author(s)**

Fabian Scheipl

**See Also**[match.call](#)


---

getF	<i>get the estimated function values from an amer-Fit...</i>
------	--

---

**Description**

get the estimated function values from an amer-Fit

**Usage**

```
getF(object, which, n=100, newdata, interval=c("NONE", "MCMC",
"RW"), addConst=TRUE, varying=1, level=0.9, sims=1000)
```

**Arguments**

object	a fitted additive (mixed) model of class <a href="#">amer-class</a>
which	(optional) an integer vector or a character vector of names giving the smooths for which fitted values are desired. Defaults to all.
n	if no newdata is given, fitted values for a regular grid with n values in the range of the respective covariates are returned
newdata	An optional data frame in which to look for variables with which to predict
interval	what method should be used to compute pointwise confidence/HPD intervals: RW= bias-adjusted empirical bayes, MCMC uses <a href="#">mcmcsamp</a>
addConst	boolean should the global intercept and intercepts for the levels of the by-variable be included in the fitted values (and their CIs) can also be a vector of the same length as which
varying	value of the varying-covariate (see <a href="#">tp</a> ) to be used if no newdata is supplied. Defaults to 1.
level	level for the confidence/HPD intervals
sims	how many iterates should be generated for the MCMC-based HPD-intervals

**Value**

a list with one `data.frame` for each function, giving newdata or the values of the generated grid plus the fitted values (and confidence/HPD intervals) if MCMC-intervals were requested, the list has an attribute "mcmc" containing the result of the call to [mcmcsamp](#), a [merMCMC-class](#) object.

**Note**

The formula used for the pointwise bias-adjusted CIs is taken from Ruppert and Wand's 'Semiparametric Regression' (2003), p. 140. These leave out the uncertainty associated with the variance component estimates. MCMC-intervals based on results from [mcmcsamp](#) don't seem to be very reliable yet and should be used with caution, especially for more complex models.

**Author(s)**

Fabian Scheipl

**See Also**[plotF](#), `tests/optionsTests.r` and the vignette for examples

---

plotF	<i>plot estimated function values + pointwise confidence intervals for an amer-Fit...</i>
-------	---

---

**Description**

plot estimated function values + pointwise confidence intervals for an amer-Fit

**Usage**

```
plotF(object, which, n=100, interval="RW", addConst=TRUE,
      trans=I, level=0.9, sims=1000, auto.layout=TRUE, rug=TRUE,
      legendPos="topright", ...)
```

**Arguments**

object	a fitted additive (mixed) model of class <a href="#">amer-class</a>
which	(optional) an integer vector or a character vector of names giving the smooths for which plots are desired. Defaults to all.
n	fitted values for a regular grid with n values in the range of the respective covariates are calculated
interval	what method should be used to compute pointwise confidence/HPD intervals. See <a href="#">getF</a> .
addConst	boolean should the global intercept and intercepts for the levels of the by-variable be included in the fitted values (and their CIs)
trans	a function that should be applied to the fitted values and ci's before plotting (e.g. the inverse link function to get plots on the scale of the response)
level	level for the confidence/HPD intervals
sims	how many iterates should be generated for the MCMC-based HPD-intervals
auto.layout	automatically set plot layout via <code>par()\$mfrow</code>
rug	add <a href="#">rug</a> -plots of the observed covariate locations
legendPos	a (vector of) keyword(s) where to put labels of by-variables (see <a href="#">legend</a> ). "none" if you don't want a legend.
...	arguments passed on to the low-level plot functions ( <code>plot</code> , <code>matlines</code> ), <code>legend</code> , and <code>title</code>

**Value**

invisibly returns the result of the `getF`-call used to do the plots

**Author(s)**

Fabian Scheipl

**See Also**

`getF`; `plotF`, `tests/optionsTests.r` and the vignette for examples

---

predict

*Get predicted/fitted values for new data*

---

**Description**

Get predicted/fitted values for the new data supplied.

**Usage**

```
## S4 method for signature 'amer'
predict(object, newdata, type=c("response", "linpred", "terms"), ...)
```

**Arguments**

object	a fitted model object inheriting from class <code>amer</code> .
newdata	a <code>data.frame</code> containing observations with all variables used in the original fit for which predicted/fitted values should be computed.
type	see Value
...	Additional, optional arguments for some methods. At present none are used.

**Value**

the `data.frame` supplied in `newdata` with additional columns containing:  
 the components of the linear predictor  $X\beta$ , named as in `fixef(object)` (prefixed by `lp.`),  
 the components of the random effects  $Z_i b_i$ , named after the grouping factors (prefixed by `blup.`),  
 the smooth terms, named as in `names(objects@smooths)`.

If `type="response"` the "fit" column contains the fitted values on the scale of the response, if `type="linpred"` the "fit" column contains the linear predictor (i.e. fixed + random + smooth effects).

**Note**

If newdata contains additional levels of grouping factors not present in the original data, these are assigned random effects of zero.

The function doesn't check whether the new data supplied for the smooth terms is within the range of the original data. Extrapolating spline fits can be dangerous.

**Author(s)**

Fabian Scheipl

---

rent	<i>Monthly rents for 1967 Munich flats</i>
------	--

---

**Description**

(Taken from package gamlss.data, see there)

**Source**

gamlss.data

---

reparameterizeDesign	<i>Reparameterize a spline basis...</i>
----------------------	---

---

**Description**

Reparameterize a spline basis

**Usage**

```
reparameterizeDesign(B, K, tol=1e-10)
```

**Arguments**

B	matrix containing the basis functions
K	penalty matrix (i.e. the penalty term for the fitted spline is the quadratic form of K and the spline coefficients)
tol	eigenvalues smaller than tol are considered zero

**Details**

Reparameterize a spline basis via the spectral decomposition of its penalty matrix into a design for the unpenalized part of the function and a design for the penalized part with i.i.d. spline coefficients. A centering constraint is applied to the design of the unpenalized part, which only makes sense if the intercept column is actually in the column space of X, i.e. constant functions are in the nullspace of the penalty.

**Value**

list with entries: "X":  $n \times (p-1)$  design matrix for unpenalized part (without intercept) "Z":  $n \times (k-p-1)$  design matrix for penalized part  $p$  is the dimension+1 of the penalty nullspace (e.g. 2 for a linear TP-basis or a B-spline with 2nd order difference penalty)

**Author(s)**

Fabian Scheipl

**See Also**

source code of tpU2D for a usage example

---

tp	<i>Generate a truncated power basis for penalized spline smoothing.</i>
----	---

---

**Description**

Generate a truncated power basis for penalized spline smoothing.

**Usage**

```
tp(x, degree=1, k=15, by=NULL, allPen=FALSE, varying=NULL, diag=FALSE,
   knots=quantile(x, probs = (1:(k - degree))/(k - degree + 1)),
   centerscale=NULL, scaledknots=FALSE)
```

**Arguments**

x	covariate for the smooth function
degree	integer: degree of truncated polynomials (0: piecewise constant, 1: piecewise linear etc..)
k	integer: dimensionality of the basis (i.e.: number of knots + degree)
by	factor variable: estimate separate functions for each level - this assumes standard treatment contrasts for the supplied factor.
allPen	boolean: if TRUE, make design for group-specific curves with common smoothing parameter: all parameters (including the normally unpenalized basis functions in X) are penalized, every level of "by" has the same amount of smoothing if FALSE, make design for separate curves for each by-level: separate smoothing parameters for every level of "by", unpenalized estimates for the coefficients associated with X
varying	numeric: if not NULL, a varying coefficient model is fit: $f(x, \text{varying}) = f(x) * \text{varying}$
diag	logical: force a diagonal covariance-matrix for the random effects for X if allPen=TRUE?
knots	vector of knot locations (optional). Defaults to quantile-based knots at the $i/(k+1-\text{degree})$ -quantiles for $i = 1, \dots, k-\text{degree}$ .
centerscale	numeric(2): center&scale x by these values if not NULL
scaledknots	boolean: are knot locations given for the rescaled x-values?

**Details**

Truncated power bases have degree unpenalized basis functions, namely  $x^1, \dots, x^{\text{degree}}$  and  $k$ -degree penalized basis functions that contain the positive part  $(x - \kappa_j)^{\text{degree}}$  for knots  $\kappa_j, j = 1, \dots, k - \text{degree}$ . This function can be used as a reference when implementing other basisGenerators that can be used for `amer`-fits. All such functions need to return a list of at least X (unpenalized basis functions, a matrix with zero columns if there are none of those), and Z (penalized basis functions) that has a `call`-attribute with the expanded call returned by `expand.call()`. All such functions need to have at least arguments `x`, `by`, `allPen`, `diag` and `varying`. See also section 4.4 in the vignette for an example on how to write your own basis-generating functions.

**Value**

list with entries: "X":  $n \times \text{degree}$  design matrix for unpenalized part (without intercept) (or a list of those for every level of `by` if `allPen=F`), "Z":  $n \times (k - \text{degree})$  design matrix for penalized part (or a list of those for every level of `by` if `allPen=F`),

**Author(s)**

Fabian Scheipl

**See Also**

[tp](#)

---

tpU	<i>Generate a modified truncated power basis for penalized spline smoothing.</i>
-----	--

---

**Description**

Generate a modified truncated power basis for penalized spline smoothing.

**Usage**

```
tpU(x, degree=2, k=15, unpen=1, by, allPen=FALSE, varying,
    diag=FALSE, knots=quantile(x, probs = (2:(k - degree +
    1))/(k - degree + 2)), centerscale, scaledknots=FALSE)
```

**Arguments**

<code>x</code>	covariate for the smooth function
<code>degree</code>	integer: degree of truncated polynomials (0: piecewise constant, 1: piecewise linear etc..)
<code>k</code>	integer: dimensionality of the basis (i.e.: number of knots + degree)
<code>unpen</code>	integer: degree of the unpenalized nullspace, must be lower than <code>degree</code> : 1 for pen. deviations from linearity, 2 for pen. deviations from quadratic from etc.

by	factor variable: estimate separate functions for each level - this assumes standard treatment contrasts for the supplied factor.
allPen	boolean: if TRUE, make design for group-specific curves with common smoothing parameter: all parameters (including the normally unpenalized basis functions in X) are penalized, every level of "by" has the same amount of smoothing. If FALSE, make design for separate curves for each by-level: separate smoothing parameters for every level of "by", unpenalized estimates for the coefficients associated with X
varying	numeric: if not NULL, a varying coefficient model is fit: $f(x, \text{varying}) = f(x) * \text{varying}$
diag	logical: force a diagonal covariance-matrix for the random effects for X if allPen=TRUE?
knots	vector of knot locations (optional). Defaults to quantile-based knot placement at the $(i + 1)/(k + 2)$ -quantiles for $i = 1, \dots, k$ .
centerscale	numeric(2): center&scale x by these values if not NULL
scaledknots	boolean: are knots given for the rescaled x-values?

**Value**

list with entries: "X":  $n \times \text{unpen}$  design matrix for unpenalized part (without intercept) "Z":  $n \times (k - \text{unpen})$  design matrix for penalized part

**Note**

This is a more detailed implementation of the example on how to define additional basis generating functions in the vignette.

**Author(s)**

Fabian Scheipl

**See Also**

[tp](#)

---

tpU2d

*Generate a 2-D truncated power basis...*

---

**Description**

Generate a 2-D truncated power basis

**Usage**

```
tpU2d(x, z, kx=7, kz=7, degree=2, dimU=1, by, allPen=FALSE,
      varying, diag=FALSE, knotsx=quantile(x, probs = (2:(kx -
      dimU + 1))/(kx - dimU + 2)), knotsz=quantile(z, probs =
      (2:(kz - dimU + 1))/(kz - dimU + 2)), centerscale,
      scaledknots=FALSE)
```

**Arguments**

x	covariate for the smooth function
z	covariate for the smooth function
kx,kz	integer: dimensionality of the marginal bases
dimU	integer: dimension of marginal nullspaces: 1 means unpenalized terms up to linear-linear-interaction, 2 means unpenalized terms up to quadratic-quadratic-interactions etc.
degree	integer: degree of (truncated) polynomial
by	factor variable: estimate separate functions for each level - this assumes standard treatment contrasts for the supplied factor.
allPen	boolean: if TRUE, make design for group-specific curves with common smoothing parameter: all parameters (including the normally unpenalized basis functions in X) are penalized, every level of "by" has the same amount of smoothing if FALSE, make design for separate curves for each by-level: separate smoothing parameters for every level of "by", unpenalized estimates for the coefficients associated with X
varying	numeric: if not NULL, a varying coefficient model is fit: $f(x, \text{varying}) = f(x) * \text{varying}$
diag	logical: force a diagonal covariance-matrix for the random effects for X if allPen=TRUE?
knotsx,knotsz	double: vector of knot locations (defaults to marginal-quantile-based knot locations)
centerscale	numeric(2): center&scale x by these values if not NULL
scaledknots	boolean: are knot locations given for the rescaled x-values?

**Details**

Generates a two-dimensional tensor product spline basis of marginal truncated power bases with a single smoothing parameter. The penalty matrix is the sum of the two kronecker products of the penalties for the marginal bases with the identity matrix.

**Value**

list with entries: "X":  $n \times (2 * \text{degree} + \text{degree}^2)$  design matrix for unpenalized part (without intercept): marginal polynomials plus their interactions, "Z":  $n \times ((kx+1)*(kz+1) - (1 + 2 * \text{degree} + \text{degree}^2))$  design matrix for penalized part,

**Author(s)**

Fabian Scheipl

**See Also**

[tp](#), [tpU](#)

---

union

*Union membership and hourly wages for 534 US workers*

---

**Description**

Union membership and hourly wages for 534 US workers. Taken from package SemiPar.

**Source**

Berndt, E.R. (1991) *The Practice of Econometrics*. New York: Addison-Wesley.

# Index

## \*Topic **classes**

amer-class, 3

## \*Topic **datasets**

dog, 5

rent, 9

union, 14

amer, 2, 3, 8, 11

amer-class, 3, 6, 7

amer-class, 3

bsp, 4

dog, 5

expand.call, 5, 11

family, 2

getF, 3, 6, 7, 8

glm, 2

legend, 7

lmer, 2

match.call, 5, 6

mcmcsamp, 6

mer-class, 3

merMCMC-class, 6

plotF, 7, 7, 8

predict, 3, 8

predict, amer-method (predict), 8

rent, 9

reparameterizeDesign, 9

rug, 7

tp, 2, 5, 6, 10, 11–13

tpU, 11, 13

tpU2d, 12

union, 14