<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0.0.3</td>
</tr>
<tr>
<td>Title</td>
<td>Bindings for Generalized Additive Models (GAM)</td>
</tr>
<tr>
<td>License</td>
<td>MIT + file LICENSE</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://hsbadr.github.io/additive/">https://hsbadr.github.io/additive/</a>, <a href="https://github.com/hsbadr/additive">https://github.com/hsbadr/additive</a></td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="https://github.com/hsbadr/additive/issues">https://github.com/hsbadr/additive/issues</a></td>
</tr>
<tr>
<td>Depends</td>
<td>mgcv (&gt;= 1.8-36), parsnip (&gt;= 0.1.7), R (&gt;= 3.6.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>dplyr, purrr, rlang, stats, tibble, utils</td>
</tr>
<tr>
<td>Suggests</td>
<td>covr, devtools, knitr, recipes, rmarkdown, roxygen2, spelling, testthat, workflows</td>
</tr>
<tr>
<td>VignetteBuilder</td>
<td>knitr</td>
</tr>
<tr>
<td>Config/testthat/edition</td>
<td>3</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.1.2</td>
</tr>
<tr>
<td>Collate</td>
<td>'additive_init.R' 'additive_load.R' 'additive_make.R' 'additive.R'</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
<tr>
<td>Language</td>
<td>en-US</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>no</td>
</tr>
<tr>
<td>Author</td>
<td>Hamada S. Badr [aut, cre] (<a href="https://orcid.org/0000-0002-9808-2344">https://orcid.org/0000-0002-9808-2344</a>)</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Hamada S. Badr <a href="mailto:badr@jhu.edu">badr@jhu.edu</a></td>
</tr>
<tr>
<td>Repository</td>
<td>CRAN</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2021-09-30 10:10:06 UTC</td>
</tr>
</tbody>
</table>
**R topics documented:**

- additive

**Description**

`additive()` is a way to generate a *specification* of a model before fitting and allows the model to be created using `mgcv` package in R.

**Usage**

```r
additive(
  mode = "regression",
  engine = "mgcv",
  fitfunc = NULL,
  formula.override = NULL,
  family = NULL,
  method = NULL,
  optimizer = NULL,
  control = NULL,
  scale = NULL,
  gamma = NULL,
  knots = NULL,
  sp = NULL,
  min.sp = NULL,
  paraPen = NULL,
  chunk.size = NULL,
  rho = NULL,
  AR.start = NULL,
  H = NULL,
  G = NULL,
  offset = NULL,
  weights = NULL,
  subset = NULL,
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  drop.intercept = NULL,
  drop.unused.levels = NULL,
  cluster = NULL,
  nthreads = NULL,
  gc.level = NULL,
  use.chol = NULL,
)
```
```r
samfrac = NULL,
coef = NULL,
discrete = NULL,
select = NULL,
fit = NULL
)

## S3 method for class 'additive'
update(
  object,
  parameters = NULL,
  fitfunc = NULL,
  formula.override = NULL,
  family = NULL,
  method = NULL,
  optimizer = NULL,
  control = NULL,
  scale = NULL,
  gamma = NULL,
  knots = NULL,
  sp = NULL,
  min.sp = NULL,
  paraPen = NULL,
  chunk.size = NULL,
  rho = NULL,
  AR.start = NULL,
  H = NULL,
  G = NULL,
  offset = NULL,
  weights = NULL,
  subset = NULL,
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  drop.intercept = NULL,
  drop.unused.levels = NULL,
  cluster = NULL,
  nthreads = NULL,
  gc.level = NULL,
  use.chol = NULL,
  samfrac = NULL,
  coef = NULL,
  discrete = NULL,
  select = NULL,
  fit = NULL,
  fresh = FALSE,
  ...
)`
additive_fit(formula, data, ...)

Arguments

mode 
A single character string for the prediction outcome mode. Possible values for
this model are "unknown", "regression", or "classification".

engine 
A single character string specifying what computational engine to use for fitting.
Possible engines are listed below. The default for this model is "mgcv".

fitfunc 
A named character vector that describes how to call a function for fitting a gen-
eralized additive model. This defaults to c(pkg = "mgcv", fun = "gam") (gam).
fitfunc should have elements pkg and fun. The former is optional but is
recommended and the latter is required. For example, c(pkg = "mgcv", fun = 
"bam") would be used to invoke bam for big data. A user-specified function is
also accepted provided that it is fully compatible with gam.

formula.override 
Overrides the formula; for details see formula.gam.

family 
This is a family object specifying the distribution and link to use in fitting etc
(see glm and family). See family.mgcv for a full list of what is available, which
goes well beyond exponential family. Note that quasi families actually result
in the use of extended quasi-likelihood if method is set to a RE/ML method
(McCullagh and Nelder, 1989, 9.6).

method 
The smoothing parameter estimation method. "GCV.Cp" to use GCV for un-
known scale parameter and Mallows’ Cp/UBRE/AIC for known scale. "GACV.Cp"
is equivalent, but using GACV in place of GCV. "REML" for REML estimation,
including of unknown scale. "P-REML" for REML estimation, but using a Pearson
estimate of the scale. "ML" and "P-ML" are similar, but using maximum
likelihood in place of REML. Beyond the exponential family "REML" is the de-
fault, and the only other option is "ML".

optimizer 
An array specifying the numerical optimization method to use to optimize the
smoothing parameter estimation criterion (given by method). "perf" (depre-
cated) for performance iteration. "outer" for the more stable direct approach.
"outer" can use several alternative optimizers, specified in the second element
of optimizer: "newton" (default), "bfgs", "optim", "nlm" and "nlm.fd" (the
latter is based entirely on finite differenced derivatives and is very slow). "efs"

control 
A list of fit control parameters to replace defaults returned by gam.control.
Values not set assume default values.

scale 
If this is positive then it is taken as the known scale parameter. Negative signals
that the scale parameter is unknown. 0 signals that the scale parameter is 1 for
Poisson and binomial and unknown otherwise. Note that (RE)ML methods can
only work with scale parameter 1 for the Poisson and binomial cases.

gamma 
Increase this beyond 1 to produce smoother models. gamma multiplies the ef-
ective degrees of freedom in the GCV or UBRE/AIC. coden/gamma can be viewed
as an effective sample size in the GCV score, and this also enables it to be used
with REML/ML. Ignored with P-RE/ML or the efs optimizer.
additive

knots  this is an optional list containing user specified knot values to be used for basis construction. For most bases the user simply supplies the knots to be used, which must match up with the k value supplied (note that the number of knots is not always just k). See tprs for what happens in the “tp”/“ts” case. Different terms can use different numbers of knots, unless they share a covariate.

sp  A vector of smoothing parameters can be provided here. Smoothness parameters must be supplied in the order that the smooth terms appear in the model formula. Negative elements indicate that the parameter should be estimated, and hence a mixture of fixed and estimated parameters is possible. If smooths share smoothing parameters then length(sp) must correspond to the number of underlying smoothing parameters.

min.sp  Lower bounds can be supplied for the smoothing parameters. Note that if this option is used then the smoothing parameters full.sp, in the returned object, will need to be added to what is supplied here to get the smoothing parameters actually multiplying the penalties. length(min.sp) should always be the same as the total number of penalties (so it may be longer than sp, if smooths share smoothing parameters).

paraPen  optional list specifying any penalties to be applied to parametric model terms. gam.models explains more.

chunk.size  The model matrix is created in chunks of this size, rather than ever being formed whole. Reset to 4*p if chunk.size < 4*p where p is the number of coefficients.

rho  An AR1 error model can be used for the residuals (based on dataframe order), of Gaussian-identity link models. This is the AR1 correlation parameter. Standardized residuals (approximately uncorrelated under correct model) returned in std.rsd if non zero. Also usable with other models when discrete=TRUE, in which case the AR model is applied to the working residuals and corresponds to a GEE approximation.

AR.start  logical variable of same length as data, TRUE at first observation of an independent section of AR1 correlation. Very first observation in data frame does not need this. If NULL then there are no breaks in AR1 correlation.

H  A user supplied fixed quadratic penalty on the parameters of the GAM can be supplied, with this as its coefficient matrix. A common use of this term is to add a ridge penalty to the parameters of the GAM in circumstances in which the model is close to un-identifiable on the scale of the linear predictor, but perfectly well defined on the response scale.

G  Usually NULL, but may contain the object returned by a previous call to gam with fit=FALSE, in which case all other arguments are ignored except for sp, gamma, in.out, scale, control, method optimizer and fit.

offset  Can be used to supply a model offset for use in fitting. Note that this offset will always be completely ignored when predicting, unlike an offset included in formula (this used to conform to the behaviour of lm and glm).

weights  prior weights on the contribution of the data to the log likelihood. Note that a weight of 2, for example, is equivalent to having made exactly the same observation twice. If you want to re-weight the contributions of each datum without changing the overall magnitude of the log likelihood, then you should normalize the weights (e.g. weights <-weights/mean(weights)).
subset  an optional vector specifying a subset of observations to be used in the fitting
process.
start  Initial values for the model coefficients.
etastart  Initial values for the linear predictor.
mustart  Initial values for the expected response.
drop.intercept  Set to TRUE to force the model to really not have the a constant in the parametric
model part, even with factor variables present. Can be vector when formula is
a list.
drop.unused.levels  by default unused levels are dropped from factors before fitting. For some
smooths involving factor variables you might want to turn this off. Only do
so if you know what you are doing.
cluster  bam can compute the computationally dominant QR decomposition in parallel
using parLapply from the parallel package, if it is supplied with a cluster on
which to do this (a cluster here can be some cores of a single machine). See
details and example code.
nthreads  Number of threads to use for non-cluster computation (e.g. combining results
from cluster nodes). If NA set to max(1,length(cluster)). See details.
gc.level  to keep the memory footprint down, it can help to call the garbage collector
often, but this takes a substantial amount of time. Setting this to zero means that
garbage collection only happens when R decides it should. Setting to 2 gives
frequent garbage collection. 1 is in between. Not as much of a problem as it
used to be.
use.chol  By default bam uses a very stable QR update approach to obtaining the QR de-
composition of the model matrix. For well conditioned models an alternative
accumulates the crossproduct of the model matrix and then finds its Choleski
decomposition, at the end. This is somewhat more efficient, computationally.
samfrac  For very large sample size Generalized additive models the number of iterations
needed for the model fit can be reduced by first fitting a model to a random
sample of the data, and using the results to supply starting values. This initial fit
is run with sloppy convergence tolerances, so is typically very low cost. samfrac
is the sampling fraction to use. 0.1 is often reasonable.
coef  initial values for model coefficients
discrete  experimental option for setting up models for use with discrete methods em-
ployed in bam. Do not modify.
select  If this is TRUE then gam can add an extra penalty to each term so that it can be
penalized to zero. This means that the smoothing parameter estimation that is
part of fitting can completely remove terms from the model. If the corresponding
smoothing parameter is estimated as zero then the extra penalty has no effect.
Use gamma to increase level of penalization.
fit  If this argument is TRUE then gam sets up the model and fits it, but if it is FALSE
then the model is set up and an object G containing what would be required to fit
is returned is returned. See argument G.
object  A Generalized Additive Model (GAM) specification.
parameters  A 1-row tibble or named list with main parameters to update. If the individual arguments are used, these will supersede the values in parameters. Also, using engine arguments in this object will result in an error.

fresh  A logical for whether the arguments should be modified in-place of or replaced wholesale.

...  Other arguments passed to internal functions.

formula  A GAM formula, or a list of formulae (see `formula.gam` and also `gam.models`). These are exactly like the formula for a GLM except that smooth terms, s, te, ti and t2, can be added to the right hand side to specify that the linear predictor depends on smooth functions of predictors (or linear functionals of these).

data  A data frame or list containing the model response variable and covariates required by the formula. By default the variables are taken from `environment(formula)`: typically the environment from which `gam` is called.

Details

The arguments are converted to their specific names at the time that the model is fit. Other options and argument can be set using `set_engine()`. If left to their defaults here (NULL), the values are taken from the underlying model functions. If parameters need to be modified, `update()` can be used in lieu of recreating the object from scratch.

The data given to the function are not saved and are only used to determine the mode of the model. For `additive()`, the possible modes are "regression" and "classification".

The model can be created by the `fit()` function using the following engines:

- mgcv: "mgcv"

Value

An updated model specification.

Engine Details

Engines may have pre-set default arguments when executing the model fit call. For this type of model, the template of the fit calls are:

```r
additive() %>%
  set_engine("mgcv") %>%
  translate()
```

```r
## Generalized Additive Model (GAM) Specification (regression)
##
## Computational engine: mgcv
##
## Model fit template:
## additive::additive_fit(formula = missing_arg(), data = missing_arg())
```
See Also

mgcv-package, gam, bam, gamObject, gam.models, smooth.terms, predict.gam, plot.gam, summary.gam, gam.side, gam.selection, gam.control, gam.check, vis.gam, family.mgcv, formula.gam, family.formula, update.formula.

Examples

additive()

show_model_info("additive")

additive(mode = "classification")
additive(mode = "regression")

set.seed(2020)
dat <- gamSim(1, n = 400, dist = "normal", scale = 2)

additive_mod <-
  additive() %>%
  set_engine("mgcv") %>%
  fit(
    y ~ s(x0) + s(x1) + s(x2) + s(x3),
    data = dat
  )

summary(additive_mod$fit)

model <- additive(select = FALSE)
model
update(model, select = TRUE)
update(model, select = TRUE, fresh = TRUE)
Index

additive, 2
additive_fit(additive), 2

bam, 4, 6, 8

family, 4, 8
family.mgcv, 4, 8
formula, 8
formula.gam, 4, 7, 8

gam, 4, 8
gam.check, 8
gam.control, 4, 8
gam.models, 5, 7, 8
gam.selection, 8
gam.side, 8
gamObject, 8
glm, 4

parLapply, 6
plot.gam, 8
predict.gam, 8

s, 7
smooth.terms, 8
summary.gam, 8

t2, 7
te, 7
ti, 7
tprs, 5

update.additive(additive), 2
update.formula, 8

vis.gam, 8