Package 'recipes'

January 11, 2018

Title Preprocessing Tools to Create Design Matrices

Version 0.1.2

Description An extensible framework to create and preprocess design matrices. Recipes consist of one or more data manipulation and analysis ``steps". Statistical parameters for the steps can be estimated from an initial data set and then applied to other data sets. The resulting design matrices can then be used as inputs into statistical or machine learning models.

URL https://github.com/topepo/recipes

BugReports https://github.com/topepo/recipes/issues

Depends R (>= 3.1), dplyr, broom

Imports tibble, stats, ipred, dimRed (>= 0.1.0), lubridate, timeDate, ddalpha, purrr, rlang (>= 0.1.1), gower, RcppRoll, tidyselect (>= 0.1.1), magrittr, Matrix

Suggests testthat, rpart, kernlab, fastICA, RANN, igraph, knitr, caret, ggplot2, rmarkdown, covr

License GPL-2

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Max Kuhn [aut, cre], Hadley Wickham [aut], RStudio [cph]

Maintainer Max Kuhn <max@rstudio.com>

Repository CRAN

Date/Publication 2018-01-11 21:50:57 UTC

R topics documented:

`add_role` *Manually Add Roles*

Description

`add_role` can add a role definition to an existing variable in the recipe.

Usage

```
add_role(recipe, ..., new_role = "predictor")
```

Arguments

`recipe`	An existing `recipe()`.
`...`	One or more selector functions to choose which variables are being assigned a role. See `selections()` for more details.
`new_role`	A character string for a single role.

Details

If a variable is selected that currently has a role, the role is changed and a warning is issued.

Value

An updated recipe object.

Examples

```
data(biomass)

# Create the recipe manually
rec <- recipe(x = biomass)
rec
summary(rec)

rec <- rec %>%
  add_role(carbon, contains("gen"), sulfur, new_role = "predictor") %>%
  add_role(sample, new_role = "id variable") %>%
  add_role(dataset, new_role = "splitting variable") %>%
  add_role(HHV, new_role = "outcome")
rec
```

---

add_step *Add a New Operation to the Current Recipe*

Description

`add_step` adds a step to the last location in the recipe. `add_check` does the same for checks.

Usage

```
add_step(rec, object)

add_check(rec, object)
```

Arguments

`rec`	A `recipe()`.
`object`	A step or check object.

Value

A updated `recipe()` with the new operation in the last slot.

bake *Apply a Trained Data Recipe*

Description

For a recipe with at least one preprocessing operations that has been trained by `prep.recipe()`, apply the computations to new data.

Usage

```
bake(object, ...)

## S3 method for class 'recipe'
bake(object, newdata, ..., composition = "tibble")
```

Arguments

`object`	A trained object such as a `recipe()` with at least one preprocessing operation.
`...`	One or more selector functions to choose which variables will be returned by the function. See `selections()` for more details. If no selectors are given, the default is to use `everything()`.
`newdata`	A data frame or tibble for whom the preprocessing will be applied.
`composition`	Either "tibble", "matrix", or "dgCMatrix" for the format of the processed data set. Note that all computations during the baking process are done in a non-sparse format. Also, note that this argument should be called **after** any selectors and the selectors should only resolve to numeric columns (otherwise an error is thrown).

Details

`bake()` takes a trained recipe and applies the operations to a data set to create a design matrix.

If the original data used to train the data are to be processed, time can be saved by using the `retain = TRUE` option of `prep()` to avoid duplicating the same operations. With this option set, `juice()` can be used instead of `bake` with `newdata` equal to the training set.

Also, any steps with `skip = TRUE` will not be applied to the data when `bake` is invoked. `juice()` will always have all of the steps applied.

Value

A tibble, matrix, or sparse matrix that may have different columns than the original columns in `newdata`.

Author(s)

Max Kuhn

See Also

recipe(), juice(), prep()

---

biomass *Biomass Data*

Description

Ghugare et al (2014) contains a data set where different biomass fuels are characterized by the amount of certain molecules (carbon, hydrogen, oxygen, nitrogen, and sulfur) and the corresponding higher heating value (HHV). These data are from their Table S.2 of the Supplementary Materials

Value

biomass a data frame

Source

Ghugare, S. B., Tiwary, S., Elangovan, V., and Tambe, S. S. (2013). Prediction of Higher Heating Value of Solid Biomass Fuels Using Artificial Intelligence Formalisms. *BioEnergy Research*, 1-12.

Examples

```
data(biomass)
str(biomass)
```

---

check_cols *Check if all Columns are Present*

Description

check_cols creates a *specification* of a recipe step that will check if all the columns of the training frame are present in the new data.

Usage

```
check_cols(recipe, ..., role = NA, trained = FALSE, skip = FALSE)

## S3 method for class 'check_cols'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The check will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are checked in the check See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this check since no new variables are created.
trained	A logical for whether the selectors in ... have been resolved by prep().
skip	A logical. Should the check be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
x	A check_cols object.

Details

This check will break the bake function if any of the checked columns does contain NA values. If the check passes, nothing is changed to the data.

Examples

```
data(biomass)

biomass_rec <- recipe(HHV ~ ., data = biomass) %>%
   step_rm(sample, dataset) %>%
   check_cols(contains("gen")) %>%
   step_center(all_predictors())

## Not run:
bake(biomass_rec, biomass[, c("carbon", "HHV")])

## End(Not run)
```

---

check_missing *Check for Missing Values*

Description

check_missing creates a a *specification* of a recipe operation that will check if variables contain missing values.

Usage

```
check_missing(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'check_missing'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The check will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are checked in the check See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this check since no new variables are created.
trained	A logical for whether the selectors in ... have been resolved by prep().
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the check be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
x	A check_missing object.

Details

This check will break the bake function if any of the checked columns does contain NA values. If the check passes, nothing is changed to the data.

Value

An updated version of recipe with the new check added to the sequence of existing operations (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected).

Examples

```
data(credit_data)
is.na(credit_data) %>% colSums()

# If the test passes, newdata is returned unaltered
recipe(credit_data) %>%
  check_missing(Age, Expenses) %>%
  prep() %>%
  bake(credit_data)

# If your training set doesn't pass, prep() will stop with an error
```

```
## Not run:
recipe(credit_data)  %>%
  check_missing(Income) %>%
  prep()

## End(Not run)

# If newdata contain missing values, the check will stop bake()

train_data <- credit_data %>% dplyr::filter(Income > 150)
test_data  <- credit_data %>% dplyr::filter(Income <= 150 | is.na(Income))

rp <- recipe(train_data) %>%
  check_missing(Income) %>%
  prep()

bake(rp, train_data)
## Not run:
bake(rp, test_data)

## End(Not run)
```

`covers` *Raw Cover Type Data*

Description

These data are raw data describing different types of forest cover-types from the UCI Machine Learning Database (see link below). There is one column in the data that has a few difference pieces of textual information (of variable lengths).

Value

`covers` a data frame

Source

https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info

Examples

```
data(covers)
str(covers)
```

credit_data *Credit Data*

Description

These data are from the website of Dr. Lluís A. Belanche Muñoz by way of a github repository of Dr. Gaston Sanchez. One data point is a missing outcome was removed from the original data.

Value

`credit_data` a data frame

Source

https://github.com/gastonstat/CreditScoring, http://bit.ly/2kkBFrk

Examples

```
data(credit_data)
str(credit_data)
```

discretize *Discretize Numeric Variables*

Description

`discretize` converts a numeric vector into a factor with bins having approximately the same number of data points (based on a training set).

Usage

```
discretize(x, ...)

## Default S3 method:
discretize(x, ...)

## S3 method for class 'numeric'
discretize(x, cuts = 4, labels = NULL, prefix = "bin",
  keep_na = TRUE, infs = TRUE, min_unique = 10, ...)

## S3 method for class 'discretize'
predict(object, newdata, ...)
```

Arguments

x	A numeric vector
...	Options to pass to stats::quantile() that should not include x or probs.
cuts	An integer defining how many cuts to make of the data.
labels	A character vector defining the factor levels that will be in the new factor (from smallest to largest). This should have length cuts+1 and should not include a level for missing (see keep_na below).
prefix	A single parameter value to be used as a prefix for the factor levels (e.g. bin1, bin2, ...). If the string is not a valid R name, it is coerced to one.
keep_na	A logical for whether a factor level should be created to identify missing values in x.
infs	A logical indicating whether the smallest and largest cut point should be infinite.
min_unique	An integer defining a sample size line of dignity for the binning. If (the number of unique values)/(cuts+1) is less than min_unique, no discretization takes place.
object	An object of class discretize.
newdata	A new numeric object to be binned.

Details

discretize estimates the cut points from x using percentiles. For example, if cuts = 3, the function estimates the quartiles of x and uses these as the cut points. If cuts = 2, the bins are defined as being above or below the median of x.

The predict method can then be used to turn numeric vectors into factor vectors.

If keep_na = TRUE, a suffix of "_missing" is used as a factor level (see the examples below).

If infs = FALSE and a new value is greater than the largest value of x, a missing value will result.

Value

discretize returns an object of class discretize and predict.discretize returns a factor vector.

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

median(biomass_tr$carbon)
discretize(biomass_tr$carbon, cuts = 2)
discretize(biomass_tr$carbon, cuts = 2, infs = FALSE)
discretize(biomass_tr$carbon, cuts = 2, infs = FALSE, keep_na = FALSE)
discretize(biomass_tr$carbon, cuts = 2, prefix = "maybe a bad idea to bin")

carbon_binned <- discretize(biomass_tr$carbon)
```

```
table(predict(carbon_binned, biomass_tr$carbon))

carbon_no_infs <- discretize(biomass_tr$carbon, infs = FALSE)
predict(carbon_no_infs, c(50, 100))

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)
rec <- rec %>% step_discretize(carbon, hydrogen)
rec <- prep(rec, biomass_tr)
binned_te <- bake(rec, biomass_te)
table(binned_te$carbon)
```

`formula.recipe` *Create a Formula from a Prepared Recipe*

Description

In case a model formula is required, the formula method can be used on a recipe to show what predictors and outcome(s) could be used.

Usage

```
## S3 method for class 'recipe'
formula(x, ...)
```

Arguments

`x`	A recipe object where all steps have been prepared.
`...`	Note currently used.

Value

A formula.

Examples

```
formula(recipe(Species + Sepal.Length ~ ., data = iris))

iris_rec <- recipe(Species ~ ., data = iris) %>%
  step_center(all_numeric()) %>%
  prep(training = iris)
formula(iris_rec)
```

has_role *Role Selection*

Description

has_role, all_predictors, and all_outcomes can be used to select variables in a formula that have certain roles. Similarly, has_type, all_numeric, and all_nominal are used to select columns based on their data type. See selections() for more details. current_info is an internal function that is unlikely to help users while the others have limited utility outside of step function arguments.

Usage

```
has_role(match = "predictor", roles = current_info()$roles)

all_predictors(roles = current_info()$roles)

all_outcomes(roles = current_info()$roles)

has_type(match = "numeric", types = current_info()$types)

all_numeric(types = current_info()$types)

all_nominal(types = current_info()$types)

current_info()
```

Arguments

match	A single character string for the query. Exact matching is used (i.e. regular expressions won't work).
roles	A character string of roles for the current set of terms.
types	A character string of roles for the current set of data types.

Value

Selector functions return an integer vector while current_info returns an environment with vectors vars, roles, and types.

Examples

```
data(biomass)

rec <- recipe(biomass) %>%
  add_role(carbon, hydrogen, oxygen, nitrogen, sulfur,
           new_role = "predictor") %>%
  add_role(HHV, new_role = "outcome") %>%
  add_role(sample, new_role = "id variable") %>%
```

```
  add_role(dataset, new_role = "splitting indicator")
recipe_info <- summary(rec)
recipe_info

has_role("id variable", roles = recipe_info$role)
all_outcomes(roles = recipe_info$role)
```

`juice` *Extract Finalized Training Set*

Description

As steps are estimated by `prep`, these operations are applied to the training set. Rather than running `bake` to duplicate this processing, this function will return variables from the processed training set.

Usage

```
juice(object, ..., composition = "tibble")
```

Arguments

`object`	A `recipe` object that has been prepared with the option `retain = TRUE`.
`...`	One or more selector functions to choose which variables will be returned by the function. See `selections()` for more details. If no selectors are given, the default is to use `everything()`.
`composition`	Either "tibble", "matrix", or "dgCMatrix" for the format of the processed data set. Note that all computations during the baking process are done in a non-sparse format. Also, note that this argument should be called **after** any selectors and the selectors should only resolve to numeric columns (otherwise an error is thrown).

Details

When preparing a recipe, if the training data set is retained using `retain = TRUE`, there is no need to `bake` the recipe to get the preprocessed training set.

`juice` will return the results of a recipes where *all steps* have been applied to the data, irrespective of the value of the step's `skip` argument.

See Also

`recipe()` `prep.recipe()` `bake.recipe()`

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

sp_signed <- rec %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_spatialsign(all_predictors())

sp_signed_trained <- prep(sp_signed, training = biomass_tr, retain = TRUE)

tr_values <- bake(sp_signed_trained, newdata = biomass_tr, all_predictors())
og_values <- juice(sp_signed_trained, all_predictors())

all.equal(tr_values, og_values)
```

`names0` *Naming Tools*

Description

`names0` creates a series of `num` names with a common prefix. The names are numbered with leading zeros (e.g. `prefix01`-`prefix10` instead of `prefix1`-`prefix10`). `dummy_names` can be used for renaming unordered and ordered dummy variables (in `step_dummy()`).

Usage

```
names0(num, prefix = "x")

dummy_names(var, lvl, ordinal = FALSE)
```

Arguments

`num`	A single integer for how many elements are created.
`prefix`	A character string that will start each name.
`var`	A single string for the original factor name.
`lvl`	A character vectors of the factor levels (in order). When used with `step_dummy()`, `lvl` would be the suffixes that result *after* `model.matrix` is called (see the example below).
`ordinal`	A logical; was the original factor ordered?

Value

names0 returns a character string of length num and dummy_names generates a character vector the same length as lvl,

Examples

```
names0(9, "x")
names0(10, "x")

example <- data.frame(y = ordered(letters[1:5]),
                      z = factor(LETTERS[1:5]))

dummy_names("z", levels(example$z)[-1])

after_mm <- colnames(model.matrix(~y, data = example))[-1]
after_mm
levels(example$y)

dummy_names("y", substring(after_mm, 2), ordinal = TRUE)
```

---

okc *OkCupid Data*

Description

These are a sample of columns of users of OkCupid dating website. The data are from Kim and Escobedo-Land (2015).

Value

okc	a data frame

Source

Kim, A. Y., and A. Escobedo-Land. 2015. "OkCupid Data for Introductory Statistics and Data Science Courses." *Journal of Statistics Education: An International Journal on the Teaching and Learning of Statistics*.

Examples

```
data(okc)
str(okc)
```

prep *Train a Data Recipe*

Description

For a recipe with at least one preprocessing operation, estimate the required parameters from a training set that can be later applied to other data sets.

Usage

```
prep(x, ...)

## S3 method for class 'recipe'
prep(x, training = NULL, fresh = FALSE, verbose = FALSE,
  retain = FALSE, stringsAsFactors = TRUE, ...)
```

Arguments

x	an object
...	further arguments passed to or from other methods (not currently used).
training	A data frame or tibble that will be used to estimate parameters for preprocessing.
fresh	A logical indicating whether already trained operation should be re-trained. If TRUE, you should pass in a data set to the argument training.
verbose	A logical that controls wether progress is reported as operations are executed.
retain	A logical: should the *preprocessed* training set be saved into the template slot of the recipe after training? This is a good idea if you want to add more steps later but want to avoid re-training the existing steps. Also, it is advisable to use retain = TRUE if any steps use the option skip = FALSE.
stringsAsFactors	A logical: should character columns be converted to factors? This affects the preprocessed training set (when retain = TRUE) as well as the results of bake.recipe.

Details

Given a data set, this function estimates the required quantities and statistics required by any operations.

prep() returns an updated recipe with the estimates.

Note that missing data handling is handled in the steps; there is no global na.rm option at the recipe-level or in prep().

Also, if a recipe has been trained using prep() and then steps are added, prep() will only update the new operations. If fresh = TRUE, all of the operations will be (re)estimated.

As the steps are executed, the training set is updated. For example, if the first step is to center the data and the second is to scale the data, the step for scaling is given the centered data.

Value

A recipe whose step objects have been updated with the required quantities (e.g. parameter estimates, model objects, etc). Also, the `term_info` object is likely to be modified as the operations are executed.

Author(s)

Max Kuhn

`print.recipe` *Print a Recipe*

Description

Print a Recipe

Usage

```
## S3 method for class 'recipe'
print(x, form_width = 30, ...)
```

Arguments

`x`	A `recipe` object
`form_width`	The number of characters used to print the variables or terms in a formula
`...`	further arguments passed to or from other methods (not currently used).

Value

The original object (invisibly)

Author(s)

Max Kuhn

recipe *Create a Recipe for Preprocessing Data*

Description

A recipe is a description of what steps should be applied to a data set in order to get it ready for data analysis.

Usage

```
recipe(x, ...)

## Default S3 method:
recipe(x, ...)

## S3 method for class 'data.frame'
recipe(x, formula = NULL, ..., vars = NULL,
  roles = NULL)

## S3 method for class 'formula'
recipe(formula, data, ...)

## S3 method for class 'matrix'
recipe(x, ...)
```

Arguments

x, data	A data frame or tibble of the *template* data set (see below).
...	Further arguments passed to or from other methods (not currently used).
formula	A model formula. No in-line functions should be used here (e.g. log(x), x:y, etc.). These types of transformations should be enacted using step functions in this package. Dots are allowed as are simple multivariate outcome terms (i.e. no need for cbind; see Examples).
vars	A character string of column names corresponding to variables that will be used in any context (see below)
roles	A character string (the same length of vars) that describes a single role that the variable will take. This value could be anything but common roles are "outcome", "predictor", "case_weight", or "ID"

Details

Recipes are alternative methods for creating design matrices and for preprocessing data.

Variables in recipes can have any type of *role* in subsequent analyses such as: outcome, predictor, case weights, stratification variables, etc.

recipe objects can be created in several ways. If the analysis only contains outcomes and predictors, the simplest way to create one is to use a simple formula (e.g. `y ~ x1 + x2`) that does not contain inline functions such as `log(x3)`. An example is given below.

Alternatively, a recipe object can be created by first specifying which variables in a data set should be used and then sequentially defining their roles (see the last example).

There are two different types of operations that can be sequentially added to a recipe. **Steps** can include common operations like logging a variable, creating dummy variables or interactions and so on. More computationally complex actions such as dimension reduction or imputation can also be specified. **Checks** are operations that conduct specific tests of the data. When the test is satisfied, the data are returned without issue or modification. Otherwise, any error is thrown.

Once a recipe has been defined, the prep() function can be used to estimate quantities required for the operations using a data set (a.k.a. the training data). prep() returns another recipe.

To apply the recipe to a data set, the bake() function is used in the same manner as predict would be for models. This applies the steps to any data set.

Note that the data passed to recipe need not be the complete data that will be used to train the steps (by prep()). The recipe only needs to know the names and types of data that will be used. For large data sets, head could be used to pass the recipe a smaller data set to save time and memory.

Value

An object of class recipe with sub-objects:

var_info	A tibble containing information about the original data set columns
term_info	A tibble that contains the current set of terms in the data set. This initially defaults to the same data contained in var_info.
steps	A list of step or check objects that define the sequence of preprocessing operations that will be applied to data. The default value is NULL
template	A tibble of the data. This is initialized to be the same as the data given in the data argument but can be different after the recipe is trained.

Author(s)

Max Kuhn

Examples

```
###############################################
# simple example:
data(biomass)

# split data
biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

# When only predictors and outcomes, a simplified formula can be used.
rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)
```

```
# Now add preprocessing steps to the recipe.

sp_signed <- rec %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_spatialsign(all_predictors())
sp_signed

# now estimate required parameters
sp_signed_trained <- prep(sp_signed, training = biomass_tr)
sp_signed_trained

# apply the preprocessing to a data set
test_set_values <- bake(sp_signed_trained, newdata = biomass_te)

# or use pipes for the entire workflow:
rec <- biomass_tr %>%
  recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_spatialsign(all_predictors())

###############################################
# multivariate example

# no need for `cbind(carbon, hydrogen)` for left-hand side
multi_y <- recipe(carbon + hydrogen ~ oxygen + nitrogen + sulfur,
                  data = biomass_tr)
multi_y <- multi_y %>%
  step_center(all_outcomes()) %>%
  step_scale(all_predictors())

multi_y_trained <- prep(multi_y, training = biomass_tr)

results <- bake(multi_y_trained, biomass_te)

###############################################
# Creating a recipe manually with different roles

rec <- recipe(biomass_tr) %>%
  add_role(carbon, hydrogen, oxygen, nitrogen, sulfur,
           new_role = "predictor") %>%
  add_role(HHV, new_role = "outcome") %>%
  add_role(sample, new_role = "id variable") %>%
  add_role(dataset, new_role = "splitting indicator")
rec
```

---

recipes *recipes: A package for computing and preprocessing design matrices.*

Description

The recipes package can be used to create design matrices for modeling and to conduct preprocessing of variables. It is meant to be a more extensive framework that R's formula method. Some differences between simple formula methods and recipes are that

1. Variables can have arbitrary roles in the analysis beyond predictors and outcomes.
2. A recipe consists of one or more steps that define actions on the variables.
3. Recipes can be defined sequentially using pipes as well as being modifiable and extensible.

Basic Functions

The three main functions are recipe(), prep(), and bake().

recipe() defines the operations on the data and the associated roles. Once the preprocessing steps are defined, any parameters are estimated using prep(). Once the data are ready for transformation, the bake() function applies the operations.

Step Functions

These functions are used to add new actions to the recipe and have the naming convention "step_action". For example, step_center() centers the data to have a zero mean and step_dummy() is used to create dummy variables.

---

selections *Methods for Select Variables in Step Functions*

Description

When selecting variables or model terms in step functions, dplyr-like tools are used. The *selector* functions can choose variables based on their name, current role, data type, or any combination of these. The selectors are passed as any other argument to the step. If the variables are explicitly stated in the step function, this might be similar to:

```
recipe( ~ ., data = USArrests) %>%
  step_pca(Murder, Assault, UrbanPop, Rape, num = 3)
```

The first four arguments indicate which variables should be used in the PCA while the last argument is a specific argument to step_pca().

Note that:

1. The selector arguments should not contain functions beyond those supported (see below).
2. These arguments are not evaluated until the prep function for the step is executed.
3. The dplyr-like syntax allows for negative sings to exclude variables (e.g. -Murder) and the set of selectors will processed in order.
4. A leading exclusion in these arguments (e.g. -Murder) has the effect of adding all variables to the list except the excluded variable(s).

Also, select helpers from the tidyselect package can also be used: tidyselect::starts_with(), tidyselect::ends_with(), tidyselect::contains(), tidyselect::matches(), tidyselect::num_range(), tidyselect::everything(), and tidyselect::one_of(). For example:

```
recipe(Species ~ ., data = iris) %>%
  step_center(starts_with("Sepal"), -contains("Width"))
```

would only select Sepal.Length

Inline functions that specify computations, such as log(x), should not be used in selectors and will produce an error. A list of allowed selector functions is below.

Columns of the design matrix that may not exist when the step is coded can also be selected. For example, when using step_pca, the number of columns created by feature extraction may not be known when subsequent steps are defined. In this case, using matches("^PC") will select all of the columns whose names start with "PC" *once those columns are created*.

There are sets of functions that can be used to select variables based on their role or type: has_role() and has_type(). For convenience, there are also functions that are more specific: all_numeric(), all_nominal(), all_predictors(), and all_outcomes(). These can be used in conjunction with the previous functions described for selecting variables using their names:

```
data(biomass)
recipe(HHV ~ ., data = biomass) %>%
  step_center(all_numeric(), -all_outcomes())
```

This results in all the numeric predictors: carbon, hydrogen, oxygen, nitrogen, and sulfur.

If a role for a variable has not been defined, it will never be selected using role-specific selectors.

Selectors can be used in step_interact() in similar ways but must be embedded in a model formula (as opposed to a sequence of selectors). For example, the interaction specification could be ~ starts_with("Species"):Sepal.Width. This can be useful if Species was converted to dummy variables previously using step_dummy().

The complete list of allowable functions in steps:

- **By name**: tidyselect::starts_with(), tidyselect::ends_with(), tidyselect::contains(), tidyselect::matches(), tidyselect::num_range(), and tidyselect::everything()
- **By role**: has_role(), all_predictors(), and all_outcomes()
- **By type**: has_type(), all_numeric(), and all_nominal()

---

step_bagimpute *Imputation via Bagged Trees*

Description

step_bagimpute creates a *specification* of a recipe step that will create bagged tree models to impute missing data.

Usage

```
step_bagimpute(recipe, ..., role = NA, trained = FALSE, models = NULL,
  options = list(nbagg = 25, keepX = FALSE),
  impute_with = imp_vars(all_predictors()), seed_val = sample.int(10^4, 1),
  skip = FALSE)

imp_vars(...)

## S3 method for class 'step_bagimpute'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables. For step_bagimpute, this indicates the variables to be imputed. When used with imp_vars, the dots indicates which variables are used to predict the missing data in each variable. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
models	The ipred::ipredbagg() objects are stored here once this bagged trees have be trained by prep.recipe().
options	A list of options to ipred::ipredbagg(). Defaults are set for the arguments nbagg and keepX but others can be passed in. **Note** that the arguments X and y should not be passed here.
impute_with	A call to imp_vars to specify which variables are used to impute the variables that can include specific variable names seperated by commas or different selectors (see selections()). If a column is included in both lists to be imputed and to be an imputation predictor, it will be removed from the latter and not used to impute itself.
seed_val	A integer used to create reproducible models. The same seed is used across all imputation models.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_bagimpute object.

Details

For each variables requiring imputation, a bagged tree is created where the outcome is the variable of interest and the predictors are any other variables listed in the impute_with formula. One advantage to the bagged tree is that is can accept predictors that have missing values themselves.

This imputation method can be used when the variable of interest (and predictors) are numeric or categorical. Imputed categorical variables will remain categorical.

Note that if a variable that is to be imputed is also in `impute_with`, this variable will be ignored.

It is possible that missing values will still occur after imputation if a large majority (or all) of the imputing variables are also missing.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables selected) and `model` (the bagged tree object).

References

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer Verlag.

Examples

```
data("credit_data")

## missing data per column
vapply(credit_data, function(x) mean(is.na(x)), c(num = 0))

set.seed(342)
in_training <- sample(1:nrow(credit_data), 2000)

credit_tr <- credit_data[ in_training, ]
credit_te <- credit_data[-in_training, ]
missing_examples <- c(14, 394, 565)

rec <- recipe(Price ~ ., data = credit_tr)

impute_rec <- rec %>%
  step_bagimpute(Status, Home, Marital, Job, Income, Assets, Debt)

imp_models <- prep(impute_rec, training = credit_tr)

imputed_te <- bake(imp_models, newdata = credit_te, everything())

credit_te[missing_examples,]
imputed_te[missing_examples, names(credit_te)]

tidy(impute_rec, number = 1)
tidy(imp_models, number = 1)
```

step_bin2factor *Create a Factors from A Dummy Variable*

Description

step_bin2factor creates a *specification* of a recipe step that will create a two-level factor from a single dummy variable.

Usage

```
step_bin2factor(recipe, ..., role = NA, trained = FALSE, levels = c("yes",
  "no"), columns = NULL, skip = FALSE)

## S3 method for class 'step_bin2factor'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	Selector functions that choose which variables will be converted. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
levels	A length 2 character string that indicate the factor levels for the 1's (in the first position) and the zeros (second)
columns	A vector with the selected variable names. This is NULL until computed by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_bin2factor object.

Details

This operation may be useful for situations where a binary piece of information may need to be represented as categorical instead of numeric. For example, naive Bayes models would do better to have factor predictors so that the binomial distribution is modeled in stead of a Gaussian probability density of numeric binary data. Note that the numeric data is only verified to be numeric (and does not count levels).

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the columns that will be affected).

Examples

```
data(covers)

rec <- recipe(~ description, covers) %>%
 step_regex(description, pattern = "(rock|stony)", result = "rocks") %>%
 step_regex(description, pattern = "(rock|stony)", result = "more_rocks") %>%
 step_bin2factor(rocks)

tidy(rec, number = 3)

rec <- prep(rec, training = covers)
results <- bake(rec, newdata = covers)

table(results$rocks, results$more_rocks)

tidy(rec, number = 3)
```

---

step_BoxCox *Box-Cox Transformation for Non-Negative Data*

Description

`step_BoxCox` creates a *specification* of a recipe step that will transform data using a simple Box-Cox transformation.

Usage

```
step_BoxCox(recipe, ..., role = NA, trained = FALSE, lambdas = NULL,
  limits = c(-5, 5), nunique = 5, skip = FALSE)

## S3 method for class 'step_BoxCox'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.

lambdas	A numeric vector of transformation values. This is NULL until computed by prep.recipe().
limits	A length 2 numeric vector defining the range to compute the transformation parameter lambda.
nunique	An integer where data that have less possible values will not be evaluate for a transformation.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_BoxCox object.

Details

The Box-Cox transformation, which requires a strictly positive variable, can be used to rescale a variable to be more similar to a normal distribution. In this package, the partial log-likelihood function is directly optimized within a reasonable set of transformation values (which can be changed by the user).

This transformation is typically done on the outcome variable using the residuals for a statistical model (such as ordinary least squares). Here, a simple null model (intercept only) is used to apply the transformation to the *predictor* variables individually. This can have the effect of making the variable distributions more symmetric.

If the transformation parameters are estimated to be very closed to the bounds, or if the optimization fails, a value of NA is used and no transformation is applied.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value (the lambda estimate).

References

Sakia, R. M. (1992). The Box-Cox transformation technique: A review. *The Statistician*, 169-178..

See Also

step_YeoJohnson() recipe() prep.recipe() bake.recipe()

Examples

```
rec <- recipe(~ ., data = as.data.frame(state.x77))

bc_trans <- step_BoxCox(rec, all_numeric())

bc_estimates <- prep(bc_trans, training = as.data.frame(state.x77))
```

```
bc_data <- bake(bc_estimates, as.data.frame(state.x77))

plot(density(state.x77[, "Illiteracy"]), main = "before")
plot(density(bc_data$Illiteracy), main = "after")

tidy(bc_trans, number = 1)
tidy(bc_estimates, number = 1)
```

---

step_bs *B-Spline Basis Functions*

Description

step_ns creates a *specification* of a recipe step that will create new columns that are basis expansions of variables using B-splines.

Usage

```
step_bs(recipe, ..., role = "predictor", trained = FALSE, objects = NULL,
  options = list(df = NULL, degree = 3), skip = FALSE)

## S3 method for class 'step_bs'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new columns created from the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
objects	A list of splines::bs() objects created once the step has been trained.
options	A list of options for splines::bs() which should not include x.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_bs object.

Details

step_bs can new features from a single variable that enable fitting routines to model this variable in a nonlinear manner. The extent of the possible nonlinearity is determined by the df, degree, or knot arguments of splines::bs(). The original variables are removed from the data and new columns are added. The naming convention for the new variables is varname_bs_1 and so on.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected and holiday.

See Also

step_poly() recipe() step_ns() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

with_splines <- rec %>%
  step_bs(carbon, hydrogen)
with_splines <- prep(with_splines, training = biomass_tr)

expanded <- bake(with_splines, biomass_te)
expanded
```

---

step_center *Centering Numeric Data*

Description

step_center creates a *specification* of a recipe step that will normalize numeric data to have a mean of zero.

Usage

```
step_center(recipe, ..., role = NA, trained = FALSE, means = NULL,
  na.rm = TRUE, skip = FALSE)

## S3 method for class 'step_center'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
means	A named numeric vector of means. This is NULL until computed by prep.recipe().
na.rm	A logical value indicating whether NA values should be removed during computations.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_center object.

Details

Centering data means that the average of a variable is subtracted from the data. step_center estimates the variable means from the data used in the training argument of prep.recipe. bake.recipe then applies the centering to new data sets using these means.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value (the means).

See Also

recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

center_trans <- rec %>%
  step_center(carbon, contains("gen"), -hydrogen)

center_obj <- prep(center_trans, training = biomass_tr)
```

```
transformed_te <- bake(center_obj, biomass_te)

biomass_te[1:10, names(transformed_te)]
transformed_te

tidy(center_trans, number = 1)
tidy(center_obj, number = 1)
```

`step_classdist` *Distances to Class Centroids*

Description

`step_classdist` creates a a *specification* of a recipe step that will convert numeric data into Mahalanobis distance measurements to the data centroid. This is done for each value of a categorical class variable.

Usage

```
step_classdist(recipe, ..., class, role = "predictor", trained = FALSE,
  mean_func = mean, cov_func = cov, pool = FALSE, log = TRUE,
  objects = NULL, skip = FALSE)

## S3 method for class 'step_classdist'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`class`	A single character string that specifies a single categorical variable to be used as the class.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that resulting distances will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`mean_func`	A function to compute the center of the distribution.
`cov_func`	A function that computes the covariance matrix
`pool`	A logical: should the covariance matrix be computed by pooling the data for all of the classes?
`log`	A logical: should the distances be transformed by the natural log function?

objects	Statistics are stored here once this step has been trained by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_classdist object.

Details

step_classdist will create a

The function will create a new column for every unique value of the class variable. The resulting variables will not replace the original values and have the prefix classdist_.

Note that, by default, the default covariance function requires that each class should have at least as many rows as variables listed in the terms argument. If pool = TRUE, there must be at least as many data points are variables overall.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected), value (the centroid of the class), and class.

Examples

```
# in case of missing data...
mean2 <- function(x) mean(x, na.rm = TRUE)

rec <- recipe(Species ~ ., data = iris) %>%
  step_classdist(all_predictors(), class = "Species",
                 pool = FALSE, mean_func = mean2)

rec_dists <- prep(rec, training = iris)

dists_to_species <- bake(rec_dists, newdata = iris, everything())
## on log scale:
dist_cols <- grep("classdist", names(dists_to_species), value = TRUE)
dists_to_species[, c("Species", dist_cols)]

tidy(rec, number = 1)
tidy(rec_dists, number = 1)
```

step_corr *High Correlation Filter*

Description

step_corr creates a *specification* of a recipe step that will potentially remove variables that have large absolute correlations with other variables.

Usage

```
step_corr(recipe, ..., role = NA, trained = FALSE, threshold = 0.9,
  use = "pairwise.complete.obs", method = "pearson", removals = NULL,
  skip = FALSE)

## S3 method for class 'step_corr'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
threshold	A value for the threshold of absolute correlation values. The step will try to remove the minimum number of columns so that all the resulting absolute correlations are less than this value.
use	A character string for the use argument to the stats::cor() function.
method	A character string for the method argument to the stats::cor() function.
removals	A character string that contains the names of columns that should be removed. These values are not determined until prep.recipe() is called.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_corr object.

Details

This step attempts to remove variables to keep the largest absolute correlation between the variables less than threshold.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` which is the columns that will be removed.

Author(s)

Original R code for filtering algorithm by Dong Li, modified by Max Kuhn. Contributions by Reynald Lescarbeau (for original in `caret` package). Max Kuhn for the `step` function.

See Also

`step_nzv()` `recipe()` `prep.recipe()` `bake.recipe()`

Examples

```
data(biomass)

set.seed(3535)
biomass$duplicate <- biomass$carbon + rnorm(nrow(biomass))

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen +
                    sulfur + duplicate,
              data = biomass_tr)

corr_filter <- rec %>%
  step_corr(all_predictors(), threshold = .5)

filter_obj <- prep(corr_filter, training = biomass_tr)

filtered_te <- bake(filter_obj, biomass_te)
round(abs(cor(biomass_tr[, c(3:7, 9)])), 2)
round(abs(cor(filtered_te)), 2)

tidy(corr_filter, number = 1)
tidy(filter_obj, number = 1)
```

---

step_count *Create Counts of Patterns using Regular Expressions*

Description

`step_count` creates a *specification* of a recipe step that will create a variable that counts instances of a regular expression pattern in text.

Usage

```
step_count(recipe, ..., role = "predictor", trained = FALSE,
  pattern = ".", normalize = FALSE, options = list(),
  result = make.names(pattern), input = NULL, skip = FALSE)

## S3 method for class 'step_count'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	A single selector functions to choose which variable will be searched for the pattern. The selector should resolve into a single variable. See selections() for more details. For the tidy method, these are not currently used.
role	For a variable created by this step, what analysis role should they be assigned?. By default, the function assumes that the new dummy variable column created by the original variable will be used as a predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
pattern	A character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector. Coerced by as.character to a character string if possible.
normalize	A logical; should the integer counts be divided by the total number of characters in the string?.
options	A list of options to gregexpr() that should not include x or pattern.
result	A single character value for the name of the new variable. It should be a valid column name.
input	A single character value for the name of the variable being searched. This is NULL until computed by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_count object.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and result (the new column name).

Examples

```
data(covers)

rec <- recipe(~ description, covers) %>%
  step_count(description, pattern = "(rock|stony)", result = "rocks") %>%
  step_count(description, pattern = "famil", normalize = TRUE)

rec2 <- prep(rec, training = covers)
rec2

count_values <- bake(rec2, newdata = covers)
count_values

tidy(rec, number = 1)
tidy(rec2, number = 1)
```

---

step_date *Date Feature Generator*

Description

`step_date` creates a a *specification* of a recipe step that will convert date data into one or more factor or numeric variables.

Usage

```
step_date(recipe, ..., role = "predictor", trained = FALSE,
  features = c("dow", "month", "year"), abbr = TRUE, label = TRUE,
  ordinal = FALSE, columns = NULL, skip = FALSE)

## S3 method for class 'step_date'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables that will be used to create the new variables. The selected variables should have class `Date` or `POSIXct`. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new variable columns created by the original variables will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.

features	A character string that includes at least one of the following values: month, dow (day of week), doy (day of year), week, month, decimal (decimal date, e.g. 2002.197), quarter, semester, year.
abbr	A logical. Only available for features month or dow. FALSE will display the day of the week as an ordered factor of character strings, such as "Sunday". TRUE will display an abbreviated version of the label, such as "Sun". abbr is disregarded if label = FALSE.
label	A logical. Only available for features month or dow. TRUE will display the day of the week as an ordered factor of character strings, such as "Sunday." FALSE will display the day of the week as a number.
ordinal	A logical: should factors be ordered? Only available for features month or dow.
columns	A character string of variables that will be used as inputs. This field is a placeholder and will be populated once prep.recipe() is used.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_date object.

Details

Unlike other steps, step_date does *not* remove the original date variables. step_rm() can be used for this purpose.

Value

For step_date, an updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected), value (the feature names), and ordinal (a logical).

See Also

step_holiday() step_rm() recipe() prep.recipe() bake.recipe()

Examples

```
library(lubridate)

examples <- data.frame(Dan = ymd("2002-03-04") + days(1:10),
                       Stefan = ymd("2006-01-13") + days(1:10))
date_rec <- recipe(~ Dan + Stefan, examples) %>%
   step_date(all_predictors())

tidy(date_rec, number = 1)

date_rec <- prep(date_rec, training = examples)

date_values <- bake(date_rec, newdata = examples)
```

```
date_values

tidy(date_rec, number = 1)
```

---

step_depth *Data Depths*

Description

step_depth creates a a *specification* of a recipe step that will convert numeric data into measurement of *data depth*. This is done for each value of a categorical class variable.

Usage

```
step_depth(recipe, ..., class, role = "predictor", trained = FALSE,
  metric = "halfspace", options = list(), data = NULL, skip = FALSE)

## S3 method for class 'step_depth'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables that will be used to create the new features. See selections() for more details. For the tidy method, these are not currently used.
class	A single character string that specifies a single categorical variable to be used as the class.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that resulting depth estimates will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
metric	A character string specifying the depth metric. Possible values are "potential", "halfspace", "Mahalanobis", "simplicialVolume", "spatial", and "zonoid".
options	A list of options to pass to the underlying depth functions. See ddalpha::depth.halfspace(), ddalpha::depth.Mahalanobis(), ddalpha::depth.potential(), ddalpha::depth.projection(), ddalpha::depth.simplicial(), ddalpha::depth.simplicialVolume(), ddalpha::depth.spatial ddalpha::depth.zonoid().
data	The training data are stored here once after prep.recipe() is executed.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_depth object.

Details

Data depth metrics attempt to measure how close data a data point is to the center of its distribution. There are a number of methods for calculating death but a simple example is the inverse of the distance of a data point to the centroid of the distribution. Generally, small values indicate that a data point not close to the centroid. `step_depth` can compute a class-specific depth for a new data point based on the proximity of the new value to the training set distribution.

Note that the entire training set is saved to compute future depth values. The saved data have been trained (i.e. prepared) and baked (i.e. processed) up to the point before the location that `step_depth` occupies in the recipe. Also, the data requirements for the different step methods may vary. For example, using `metric = "Mahalanobis"` requires that each class should have at least as many rows as variables listed in the `terms` argument.

The function will create a new column for every unique value of the `class` variable. The resulting variables will not replace the original values and have the prefix `depth_`.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables selected) and `class`.

Examples

```
# halfspace depth is the default
rec <- recipe(Species ~ ., data = iris) %>%
  step_depth(all_predictors(), class = "Species")

rec_dists <- prep(rec, training = iris)

dists_to_species <- bake(rec_dists, newdata = iris)
dists_to_species

tidy(rec, number = 1)
tidy(rec_dists, number = 1)
```

---

step_discretize *Discretize Numeric Variables*

Description

`step_discretize` creates a a *specification* of a recipe step that will convert numeric data into a factor with bins having approximately the same number of data points (based on a training set).

Usage

```
step_discretize(recipe, ..., role = NA, trained = FALSE, objects = NULL,
  options = list(), skip = FALSE)

## S3 method for class 'step_discretize'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	For step_discretize, the dots specify one or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
objects	The discretize() objects are stored here once the recipe has be trained by prep.recipe().
options	A list of options to discretize(). A defaults is set for the argument x. Note that the using the options prefix and labels when more than one variable is being transformed might be problematic as all variables inherit those values.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_discretize object

Value

step_discretize returns an updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value (the breaks).

---

step_downsample *Down-Sample a Data Set Based on a Factor Variable*

Description

step_downsample creates a *specification* of a recipe step that will remove rows of a data set to make the occurrence of levels in a specific factor level equal.

Usage

```
step_downsample(recipe, ..., ratio = 1, role = NA, trained = FALSE,
  column = NULL, target = NA, skip = TRUE)

## S3 method for class 'step_downsample'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variable is used to sample the data. See selections() for more details. The selection should result in *single factor variable*. For the tidy method, these are not currently used.
ratio	A numeric value for the ratio of the minority-to-majority frequencies. The default value (1) means that all other levels are sampled down to have the same frequency as the least occurring level. A value of 2 would mean that the majority levels will have (at most) (approximately) twice as many rows than the minority level.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
column	A character string of the variable name that will be populated (eventually) by the ... selectors.
target	An integer that will be used to subsample. This should not be set by the user and will be populated by prep.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_downsample object.

Details

Down-sampling is intended to be performed on the *training* set alone. For this reason, the default is skip = TRUE. It is advisable to use prep(recipe, retain = TRUE) when preparing the recipe; in this way juice() can be used to obtain the down-sampled version of the data.

If there are missing values in the factor variable that is used to define the sampling, missing data are selected at random in the same way that the other factor levels are sampled. Missing values are not used to determine the amount of data in the minority level

For any data with factor levels occurring with the same frequency as the minority level, all data will be retained.

All columns in the data are sampled and returned by juice() and bake().

Keep in mind that the location of down-sampling in the step may have effects. For example, if centering and scaling, it is not clear whether those operations should be conducted *before* or *after* rows are removed.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the variable used to sample.

Examples

```
data(okc)

sort(table(okc$diet, useNA = "always"))

ds_rec <- recipe( ~ ., data = okc) %>%
  step_downsample(diet) %>%
  prep(training = okc, retain = TRUE)

table(juice(ds_rec)$diet, useNA = "always")

# since `skip` defaults to TRUE, baking the step has no effect
baked_okc <- bake(ds_rec, newdata = okc)
table(baked_okc$diet, useNA = "always")
```

`step_dummy` *Dummy Variables Creation*

Description

`step_dummy` creates a a *specification* of a recipe step that will convert nominal data (e.g. character or factors) into one or more numeric binary model terms for the levels of the original data.

Usage

```
step_dummy(recipe, ..., role = "predictor", trained = FALSE,
  contrast = options("contrasts"), naming = dummy_names, levels = NULL,
  skip = FALSE)

## S3 method for class 'step_dummy'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will be used to create the dummy variables. See `selections()` for more details. The selected variables must be factors. For the `tidy` method, these are not currently used.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the binary dummy variable columns created by the original variables will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`contrast`	A specification for which type of contrast should be used to make a set of full rank dummy variables. See `stats::contrasts()` for more details. **not currently working**

naming	A function that defines the naming convention for new dummy columns. See Details below.
levels	A list that contains the information needed to create dummy variables for each variable contained in terms. This is NULL until the step is trained by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_dummy object.

Details

step_dummy will create a set of binary dummy variables from a factor variable. For example, if an unordered factor column in the data set has levels of "red", "green", "blue", the dummy variable bake will create two additional columns of 0/1 data for two of those three values (and remove the original column). For ordered factors, polynomial contrasts are used to encode the numeric values.

By default, the missing dummy variable (i.e. the reference cell) will correspond to the first level of the unordered factor being converted.

The function allows for non-standard naming of the resulting variables. For an unordered factor named x, with levels "a" and "b", the default naming convention would be to create a new variable called x_b. Note that if the factor levels are not valid variable names (e.g. "some text with spaces"), it will be changed by base::make.names() to be valid (see the example below). The naming format can be changed using the naming argument and the function dummy_names() is the default. This function will also change the names of ordinal dummy variables. Instead of values such as ".L", ".Q", or "^4", ordinal dummy variables are given simple integer suffixes such as "_1", "_2", etc.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected).

See Also

step_factor2string(), step_string2factor(), dummy_names(), step_regex(), step_count(), step_ordinalscore(), step_unorder(), step_other() step_novel()

Examples

```
data(okc)
okc <- okc[complete.cases(okc),]

rec <- recipe(~ diet + age + height, data = okc)

dummies <- rec %>% step_dummy(diet)
dummies <- prep(dummies, training = okc)

dummy_data <- bake(dummies, newdata = okc)
```

```
unique(okc$diet)
grep("^diet", names(dummy_data), value = TRUE)

tidy(dummies, number = 1)
```

`step_factor2string` *Convert Factors to Strings*

Description

`step_factor2string` will convert one or more factor vectors to strings.

Usage

```
step_factor2string(recipe, ..., role = NA, trained = FALSE,
  columns = FALSE, skip = FALSE)

## S3 method for class 'step_factor2string'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will converted to strings See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`columns`	A character string of variables that will be converted. This is `NULL` until computed by `prep.recipe()`.
`skip`	A logical. Should the step be skipped when the recipe is baked by `bake.recipe()`? While all operations are baked when `prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`x`	A `step_factor2string` object.

Details

`prep` has an option `stringsAsFactors` that defaults to `TRUE`. If this step is used with the default option, the string(s() produced by this step will be converted to factors after all of the steps have been prepped.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the columns that will be affected).

See Also

`step_string2factor()` `step_dummy()`

Examples

```
data(okc)

rec <- recipe(~ diet + location, data = okc)

rec <- rec %>%
  step_string2factor(diet)

factor_test <- rec %>%
  prep(training = okc,
       stringsAsFactors = FALSE,
       retain = TRUE) %>%
  juice
# diet is a
class(factor_test$diet)

rec <- rec %>%
  step_factor2string(diet)

string_test <- rec %>%
  prep(training = okc,
       stringsAsFactors = FALSE,
       retain = TRUE) %>%
  juice
# diet is a
class(string_test$diet)

tidy(rec, number = 1)
```

---

step_holiday *Holiday Feature Generator*

Description

`step_holiday` creates a a *specification* of a recipe step that will convert date data into one or more binary indicator variables for common holidays.

Usage

```
step_holiday(recipe, ..., role = "predictor", trained = FALSE,
  holidays = c("LaborDay", "NewYearsDay", "ChristmasDay"), columns = NULL,
  skip = FALSE)

## S3 method for class 'step_holiday'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be used to create the new variables. The selected variables should have class Date or POSIXct. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new variable columns created by the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
holidays	A character string that includes at least one holiday supported by the timeDate package. See timeDate::listHolidays() for a complete list.
columns	A character string of variables that will be used as inputs. This field is a placeholder and will be populated once prep.recipe() is used.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_holiday object.

Details

Unlike other steps, step_holiday does *not* remove the original date variables. step_rm() can be used for this purpose.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected and holiday.

See Also

step_date() step_rm() recipe() prep.recipe() bake.recipe() timeDate::listHolidays()

Examples

```
library(lubridate)

examples <- data.frame(someday = ymd("2000-12-20") + days(0:40))
holiday_rec <- recipe(~ someday, examples) %>%
   step_holiday(all_predictors())

holiday_rec <- prep(holiday_rec, training = examples)
holiday_values <- bake(holiday_rec, newdata = examples)
holiday_values
```

`step_hyperbolic` *Hyperbolic Transformations*

Description

`step_hyperbolic` creates a *specification* of a recipe step that will transform data using a hyperbolic function.

Usage

```
step_hyperbolic(recipe, ..., role = NA, trained = FALSE, func = "sin",
  inverse = TRUE, columns = NULL, skip = FALSE)

## S3 method for class 'step_hyperbolic'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`func`	A character value for the function. Valid values are "sin", "cos", or "tan".
`inverse`	A logical: should the inverse function be used?
`columns`	A character string of variable names that will be populated (eventually) by the `terms` argument.
`skip`	A logical. Should the step be skipped when the recipe is baked by `bake.recipe()`? While all operations are baked when `prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`x`	A `step_hyperbolic` object.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the columns that will be affected), `inverse`, and `func`.

See Also

`step_logit()` `step_invlogit()` `step_log()` `step_sqrt()` `recipe()` `prep.recipe()` `bake.recipe()`

Examples

```
set.seed(313)
examples <- matrix(rnorm(40), ncol = 2)
examples <- as.data.frame(examples)

rec <- recipe(~ V1 + V2, data = examples)

cos_trans <- rec  %>%
  step_hyperbolic(all_predictors(),
                  func = "cos", inverse = FALSE)

cos_obj <- prep(cos_trans, training = examples)

transformed_te <- bake(cos_obj, examples)
plot(examples$V1, transformed_te$V1)

tidy(cos_trans, number = 1)
tidy(cos_obj, number = 1)
```

---

step_ica *ICA Signal Extraction*

Description

`step_ica` creates a *specification* of a recipe step that will convert numeric data into one or more independent components.

Usage

```
step_ica(recipe, ..., role = "predictor", trained = FALSE, num = 5,
  options = list(), res = NULL, prefix = "IC", skip = FALSE)

## S3 method for class 'step_ica'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be used to compute the components. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new independent component columns created by the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num	The number of ICA components to retain as new predictors. If num is greater than the number of columns or the number of possible components, a smaller value will be used.
options	A list of options to fastICA::fastICA(). No defaults are set here. **Note** that the arguments X and n.comp should not be passed here.
res	The fastICA::fastICA() object is stored here once this preprocessing step has be trained by prep.recipe().
prefix	A character string that will be the prefix to the resulting new variables. See notes below.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_ica object.

Details

Independent component analysis (ICA) is a transformation of a group of variables that produces a new set of artificial features or components. ICA assumes that the variables are mixtures of a set of distinct, non-Gaussian signals and attempts to transform the data to isolate these signals. Like PCA, the components are statistically independent from one another. This means that they can be used to combat large inter-variables correlations in a data set. Also like PCA, it is advisable to center and scale the variables prior to running ICA.

This package produces components using the "FastICA" methodology (see reference below).

The argument num controls the number of components that will be retained (the original variables that are used to derive the components are removed from the data). The new components will have names that begin with prefix and a sequence of numbers. The variable names are padded with zeros. For example, if num < 10, their names will be IC1 - IC9. If num = 101, the names would be IC001 - IC101.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected), value (the loading), and component.

References

Hyvarinen, A., and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5), 411-430.

See Also

step_pca() step_kpca() step_isomap() recipe() prep.recipe() bake.recipe()

Examples

```
# from fastICA::fastICA
set.seed(131)
S <- matrix(runif(400), 200, 2)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
X <- as.data.frame(S %*% A)

tr <- X[1:100, ]
te <- X[101:200, ]

rec <- recipe( ~ ., data = tr)

ica_trans <- step_center(rec,  V1, V2)
ica_trans <- step_scale(ica_trans, V1, V2)
ica_trans <- step_ica(ica_trans, V1, V2, num = 2)
ica_estimates <- prep(ica_trans, training = tr)
ica_data <- bake(ica_estimates, te)

plot(te$V1, te$V2)
plot(ica_data$IC1, ica_data$IC2)

tidy(ica_trans, number = 3)
tidy(ica_estimates, number = 3)
```

---

step_interact *Create Interaction Variables*

Description

step_interact creates a *specification* of a recipe step that will create new columns that are interaction terms between two or more variables.

Usage

```
step_interact(recipe, terms, role = "predictor", trained = FALSE,
  objects = NULL, sep = "_x_", skip = FALSE)

## S3 method for class 'step_interact'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
terms	A traditional R formula that contains interaction terms. This can include . and selectors.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new columns created from the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
objects	A list of terms objects for each individual interaction.
sep	A character value used to delineate variables in an interaction (e.g. var1_x_var2 instead of the more traditional var1:var2).
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_interact object
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.

Details

step_interact can create interactions between variables. It is primarily intended for **numeric data**; categorical variables should probably be converted to dummy variables using step_dummy() prior to being used for interactions.

Unlike other step functions, the terms argument should be a traditional R model formula but should contain no inline functions (e.g. log). For example, for predictors A, B, and C, a formula such as ~A:B:C can be used to make a three way interaction between the variables. If the formula contains terms other than interactions (e.g. (A+B+C)^3) only the interaction terms are retained for the design matrix.

The separator between the variables defaults to "_x_" so that the three way interaction shown previously would generate a column named A_x_B_x_C. This can be changed using the sep argument.

When dummy variables are created and are used in interactions, selectors can help specify the interactions succinctly. For example, suppose a factor column X gets converted to dummy variables x_2, x_3, ..., x_6 using step_dummy(). If you wanted an interaction with numeric column z, you could create a set of specific interaction effects (e.g. x_2:z + x_3:z and so on) or you could use starts_with("z_"):z. When prep() evaluates this step, starts_with("z_") resolves to (x_2 + x_3 + x_4 + x_5 + x6) so that the formula is now (x_2 + x_3 + x_4 + x_5 + x6):z and all two-way interactions are created.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the interaction effects.

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

int_mod_1 <- rec %>%
  step_interact(terms = ~ carbon:hydrogen)

int_mod_2 <- rec %>%
  step_interact(terms = ~ (matches("gen$") + sulfur)^2)

int_mod_1 <- prep(int_mod_1, training = biomass_tr)
int_mod_2 <- prep(int_mod_2, training = biomass_tr)

dat_1 <- bake(int_mod_1, biomass_te)
dat_2 <- bake(int_mod_2, biomass_te)

names(dat_1)
names(dat_2)

tidy(int_mod_1, number = 1)
tidy(int_mod_2, number = 1)
```

---

step_intercept *Add intercept (or constant) column*

Description

`step_intercept` creates a *specification* of a recipe step that will add an intercept or constant term in the first column of a data matrix. `step_intercept` has defaults to *predictor* role so that it is by default called in the bake step. Be careful to avoid unintentional transformations when calling steps with `all_predictors`.

Usage

```
step_intercept(recipe, ..., role = "predictor", trained = FALSE,
  name = "intercept", value = 1, skip = FALSE)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	Argument ignored; included for consistency with other step specification functions.

role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new columns created from the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated. Again included for consistency.
name	Character name for new added column
value	A numeric constant to fill the intercept column. Defaults to 1.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any).

See Also

recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)
rec_trans <- recipe(HHV ~ ., data = biomass_tr[, -(1:2)]) %>%
  step_intercept(value = 2)

rec_obj <- prep(rec_trans, training = biomass_tr)

with_intercept <- bake(rec_obj, biomass_te)
with_intercept
```

---

step_invlogit *Inverse Logit Transformation*

Description

step_invlogit creates a *specification* of a recipe step that will transform the data from real values to be between zero and one.

Usage

```
step_invlogit(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'step_invlogit'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_invlogit object.

Details

The inverse logit transformation takes values on the real line and translates them to be between zero and one using the function f(x) = 1/(1+exp(-x)).

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected.

See Also

step_logit() step_log() step_sqrt() step_hyperbolic() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)
```

```
ilogit_trans <- rec  %>%
  step_center(carbon, hydrogen) %>%
  step_scale(carbon, hydrogen) %>%
  step_invlogit(carbon, hydrogen)

ilogit_obj <- prep(ilogit_trans, training = biomass_tr)

transformed_te <- bake(ilogit_obj, biomass_te)
plot(biomass_te$carbon, transformed_te$carbon)
```

`step_isomap` *Isomap Embedding*

Description

`step_isomap` creates a *specification* of a recipe step that will convert numeric data into one or more new dimensions.

Usage

```
step_isomap(recipe, ..., role = "predictor", trained = FALSE, num = 5,
  options = list(knn = 50, .mute = c("message", "output")), res = NULL,
  prefix = "Isomap", skip = FALSE)

## S3 method for class 'step_isomap'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will be used to compute the dimensions. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new dimension columns created by the original variables will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`num`	The number of isomap dimensions to retain as new predictors. If `num` is greater than the number of columns or the number of possible dimensions, a smaller value will be used.
`options`	A list of options to `dimRed::Isomap()`.
`res`	The `dimRed::Isomap()` object is stored here once this preprocessing step has be trained by `prep.recipe()`.

prefix	A character string that will be the prefix to the resulting new variables. See notes below.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_isomap object

Details

Isomap is a form of multidimensional scaling (MDS). MDS methods try to find a reduced set of dimensions such that the geometric distances between the original data points are preserved. This version of MDS uses nearest neighbors in the data as a method for increasing the fidelity of the new dimensions to the original data values.

It is advisable to center and scale the variables prior to running Isomap (step_center and step_scale can be used for this purpose).

The argument num controls the number of components that will be retained (the original variables that are used to derive the components are removed from the data). The new components will have names that begin with prefix and a sequence of numbers. The variable names are padded with zeros. For example, if num < 10, their names will be Isomap1 - Isomap9. If num = 101, the names would be Isomap001 - Isomap101.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected).

References

De Silva, V., and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems*. 721-728.

dimRed, a framework for dimensionality reduction, https://github.com/gdkrmr

See Also

step_pca() step_kpca() step_ica() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

im_trans <- rec %>%
```

```
  step_YeoJohnson(all_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_isomap(all_predictors(),
              options = list(knn = 100),
              num = 2)

im_estimates <- prep(im_trans, training = biomass_tr)

im_te <- bake(im_estimates, biomass_te)

rng <- extendrange(c(im_te$Isomap1, im_te$Isomap2))
plot(im_te$Isomap1, im_te$Isomap2,
     xlim = rng, ylim = rng)

tidy(im_trans, number = 4)
tidy(im_estimates, number = 4)
```

`step_knnimpute` *Imputation via K-Nearest Neighbors*

Description

`step_knnimpute` creates a *specification* of a recipe step that will impute missing data using nearest neighbors.

Usage

```
step_knnimpute(recipe, ..., role = NA, trained = FALSE, K = 5,
  impute_with = imp_vars(all_predictors()), ref_data = NULL,
  columns = NULL, skip = FALSE)

## S3 method for class 'step_knnimpute'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables. For `step_knnimpute`, this indicates the variables to be imputed. When used with `imp_vars`, the dots indicates which variables are used to predict the missing data in each variable. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`K`	The number of neighbors.

impute_with	A call to imp_vars to specify which variables are used to impute the variables that can include specific variable names separated by commas or different selectors (see selections()). If a column is included in both lists to be imputed and to be an imputation predictor, it will be removed from the latter and not used to impute itself.
ref_data	A tibble of data that will reflect the data preprocessing done up to the point of this imputation step. This is NULL until the step is trained by prep.recipe().
columns	The column names that will be imputed and used for imputation. This is NULL until the step is trained by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_knnimpute object.

Details

The step uses the training set to impute any other data sets. The only distance function available is Gower's distance which can be used for mixtures of nominal and numeric data.

Once the nearest neighbors are determined, the mode is used to predictor nominal variables and the mean is used for numeric data.

Note that if a variable that is to be imputed is also in impute_with, this variable will be ignored.

It is possible that missing values will still occur after imputation if a large majority (or all) of the imputing variables are also missing.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables for imputation), predictors (those variables used to impute), and K.

References

Gower, C. (1971) "A general coefficient of similarity and some of its properties," Biometrics, 857-871.

Examples

```
library(recipes)
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training", ]
biomass_te <- biomass[biomass$dataset == "Testing", ]
biomass_te_whole <- biomass_te

# induce some missing data at random
set.seed(9039)
```

```
carb_missing <- sample(1:nrow(biomass_te), 3)
nitro_missing <- sample(1:nrow(biomass_te), 3)

biomass_te$carbon[carb_missing] <- NA
biomass_te$nitrogen[nitro_missing] <- NA

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

ratio_recipe <- rec %>%
  step_knnimpute(all_predictors(), K = 3)
ratio_recipe2 <- prep(ratio_recipe, training = biomass_tr)
imputed <- bake(ratio_recipe2, biomass_te)

# how well did it work?
summary(biomass_te_whole$carbon)
cbind(before = biomass_te_whole$carbon[carb_missing],
      after = imputed$carbon[carb_missing])

summary(biomass_te_whole$nitrogen)
cbind(before = biomass_te_whole$nitrogen[nitro_missing],
      after = imputed$nitrogen[nitro_missing])

tidy(ratio_recipe, number = 1)
tidy(ratio_recipe2, number = 1)
```

`step_kpca` *Kernel PCA Signal Extraction*

Description

`step_kpca` a *specification* of a recipe step that will convert numeric data into one or more principal components using a kernel basis expansion.

Usage

```
step_kpca(recipe, ..., role = "predictor", trained = FALSE, num = 5,
  res = NULL, options = list(kernel = "rbfdot", kpar = list(sigma = 0.2)),
  prefix = "kPC", skip = FALSE)

## S3 method for class 'step_kpca'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will be used to compute the components. See `selections()` for more details. For the `tidy` method, these are not currently used.

role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new principal component columns created by the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num	The number of PCA components to retain as new predictors. If num is greater than the number of columns or the number of possible components, a smaller value will be used.
res	An S4 kernlab::kpca() object is stored here once this preprocessing step has be trained by prep.recipe().
options	A list of options to kernlab::kpca(). Defaults are set for the arguments kernel and kpar but others can be passed in. **Note** that the arguments x and features should not be passed here (or at all).
prefix	A character string that will be the prefix to the resulting new variables. See notes below.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_kpca object

Details

Kernel principal component analysis (kPCA) is an extension a PCA analysis that conducts the calculations in a broader dimensionality defined by a kernel function. For example, if a quadratic kernel function were used, each variable would be represented by its original values as well as its square. This nonlinear mapping is used during the PCA analysis and can potentially help find better representations of the original data.

As with ordinary PCA, it is important to standardized the variables prior to running PCA (step_center and step_scale can be used for this purpose).

When performing kPCA, the kernel function (and any important kernel parameters) must be chosen. The **kernlab** package is used and the reference below discusses the types of kernels available and their parameter(s). These specifications can be made in the kernel and kpar slots of the options argument to step_kpca.

The argument num controls the number of components that will be retained (the original variables that are used to derive the components are removed from the data). The new components will have names that begin with prefix and a sequence of numbers. The variable names are padded with zeros. For example, if num < 10, their names will be kPC1 - kPC9. If num = 101, the names would be kPC001 - kPC101.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected).

References

Scholkopf, B., Smola, A., and Muller, K. (1997). Kernel principal component analysis. *Lecture Notes in Computer Science*, 1327, 583-588.

Karatzoglou, K., Smola, A., Hornik, K., and Zeileis, A. (2004). kernlab - An S4 package for kernel methods in R. *Journal of Statistical Software*, 11(1), 1-20.

See Also

`step_pca()` `step_ica()` `step_isomap()` `recipe()` `prep.recipe()` `bake.recipe()`

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

kpca_trans <- rec %>%
  step_YeoJohnson(all_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_kpca(all_predictors())

kpca_estimates <- prep(kpca_trans, training = biomass_tr)

kpca_te <- bake(kpca_estimates, biomass_te)

rng <- extendrange(c(kpca_te$kPC1, kpca_te$kPC2))
plot(kpca_te$kPC1, kpca_te$kPC2,
     xlim = rng, ylim = rng)

tidy(kpca_trans, number = 4)
tidy(kpca_estimates, number = 4)
```

---

step_lincomb *Linear Combination Filter*

Description

`step_lincomb` creates a *specification* of a recipe step that will potentially remove numeric variables that have linear combinations between them.

Usage

```
step_lincomb(recipe, ..., role = NA, trained = FALSE, max_steps = 5,
  removals = NULL, skip = FALSE)

## S3 method for class 'step_lincomb'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
max_steps	A value.
removals	A character string that contains the names of columns that should be removed. These values are not determined until prep.recipe() is called.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_lincomb object.

Details

This step finds exact linear combinations between two or more variables and recommends which column(s) should be removed to resolve the issue. This algorithm may need to be applied multiple times (as defined by max_steps).

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be removed.

Author(s)

Max Kuhn, Kirk Mettler, and Jed Wing

See Also

step_nzv() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass$new_1 <- with(biomass,
                      .1*carbon - .2*hydrogen + .6*sulfur)
biomass$new_2 <- with(biomass,
                      .5*carbon - .2*oxygen + .6*nitrogen)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen +
                sulfur + new_1 + new_2,
              data = biomass_tr)

lincomb_filter <- rec %>%
  step_lincomb(all_predictors())

lincomb_filter_trained <- prep(lincomb_filter, training = biomass_tr)
lincomb_filter_trained

tidy(lincomb_filter, number = 1)
tidy(lincomb_filter_trained, number = 1)
```

---

step_log *Logarithmic Transformation*

Description

step_log creates a *specification* of a recipe step that will log transform data.

Usage

```
step_log(recipe, ..., role = NA, trained = FALSE, base = exp(1),
  columns = NULL, skip = FALSE)

## S3 method for class 'step_log'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.

base	A numeric value for the base.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_log object.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the columns that will be affected) and base.

See Also

step_logit() step_invlogit() step_hyperbolic() step_sqrt() recipe() prep.recipe() bake.recipe()

Examples

```
set.seed(313)
examples <- matrix(exp(rnorm(40)), ncol = 2)
examples <- as.data.frame(examples)

rec <- recipe(~ V1 + V2, data = examples)

log_trans <- rec  %>%
  step_log(all_predictors())

log_obj <- prep(log_trans, training = examples)

transformed_te <- bake(log_obj, examples)
plot(examples$V1, transformed_te$V1)

tidy(log_trans, number = 1)
tidy(log_obj, number = 1)
```

---

step_logit *Logit Transformation*

Description

step_logit creates a *specification* of a recipe step that will logit transform the data.

Usage

```
step_logit(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'step_logit'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_logit object.

Details

The logit transformation takes values between zero and one and translates them to be on the real line using the function f(p) = log(p/(1-p)).

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected.

See Also

step_invlogit() step_log() step_sqrt() step_hyperbolic() recipe() prep.recipe() bake.recipe()

Examples

```
set.seed(313)
examples <- matrix(runif(40), ncol = 2)
examples <- data.frame(examples)

rec <- recipe(~ X1 + X2, data = examples)

logit_trans <- rec  %>%
```

```
  step_logit(all_predictors())

logit_obj <- prep(logit_trans, training = examples)

transformed_te <- bake(logit_obj, examples)
plot(examples$X1, transformed_te$X1)

tidy(logit_trans, number = 1)
tidy(logit_obj, number = 1)
```

`step_lowerimpute` *Impute Numeric Data Below the Threshold of Measurement*

Description

`step_lowerimpute` creates a *specification* of a recipe step designed for cases where the non-negative numeric data cannot be measured below a known value. In these cases, one method for imputing the data is to substitute the truncated value by a random uniform number between zero and the truncation point.

Usage

```
step_lowerimpute(recipe, ..., role = NA, trained = FALSE,
  threshold = NULL, skip = FALSE)

## S3 method for class 'step_lowerimpute'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`threshold`	A named numeric vector of lower bounds This is `NULL` until computed by `prep.recipe()`.
`skip`	A logical. Should the step be skipped when the recipe is baked by `bake.recipe()`? While all operations are baked when `prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`x`	A `step_lowerimpute` object.

Details

step_lowerimpute estimates the variable minimums from the data used in the training argument of prep.recipe. bake.recipe then simulates a value for any data at the minimum with a random uniform value between zero and the minimum.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value for the estimated threshold.

Examples

```
library(recipes)
data(biomass)

## Truncate some values to emulate what a lower limit of
## the measurement system might look like

biomass$carbon <- ifelse(biomass$carbon > 40, biomass$carbon, 40)
biomass$hydrogen <- ifelse(biomass$hydrogen > 5, biomass$carbon, 5)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

impute_rec <- rec %>%
  step_lowerimpute(carbon, hydrogen)

tidy(impute_rec, number = 1)

impute_rec <- prep(impute_rec, training = biomass_tr)

tidy(impute_rec, number = 1)

transformed_te <- bake(impute_rec, biomass_te)

plot(transformed_te$carbon, biomass_te$carbon,
     xlab = "pre-imputation", ylab = "imputed")
```

---

step_meanimpute *Impute Numeric Data Using the Mean*

Description

step_meanimpute creates a *specification* of a recipe step that will substitute missing values of numeric variables by the training set mean of those variables.

Usage

```
step_meanimpute(recipe, ..., role = NA, trained = FALSE, means = NULL,
  trim = 0, skip = FALSE)

## S3 method for class 'step_meanimpute'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
means	A named numeric vector of means. This is NULL until computed by prep.recipe().
trim	The fraction (0 to 0.5) of observations to be trimmed from each end of the variables before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_meanimpute object.

Details

step_meanimpute estimates the variable means from the data used in the training argument of prep.recipe. bake.recipe then applies the new values to new data sets using these averages.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and model (the mean value).

Examples

```
data("credit_data")

## missing data per column
vapply(credit_data, function(x) mean(is.na(x)), c(num = 0))

set.seed(342)
in_training <- sample(1:nrow(credit_data), 2000)
```

```
credit_tr <- credit_data[ in_training, ]
credit_te <- credit_data[-in_training, ]
missing_examples <- c(14, 394, 565)

rec <- recipe(Price ~ ., data = credit_tr)

impute_rec <- rec %>%
  step_meanimpute(Income, Assets, Debt)

imp_models <- prep(impute_rec, training = credit_tr)

imputed_te <- bake(imp_models, newdata = credit_te, everything())

credit_te[missing_examples,]
imputed_te[missing_examples, names(credit_te)]

tidy(impute_rec, number = 1)
tidy(imp_models, number = 1)
```

`step_modeimpute` *Impute Nominal Data Using the Most Common Value*

Description

`step_modeimpute` creates a *specification* of a recipe step that will substitute missing values of nominal variables by the training set mode of those variables.

Usage

```
step_modeimpute(recipe, ..., role = NA, trained = FALSE, modes = NULL,
  skip = FALSE)

## S3 method for class 'step_modeimpute'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`modes`	A named character vector of modes. This is `NULL` until computed by `prep.recipe()`.

skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_modeimpute object.

Details

step_modeimpute estimates the variable modes from the data used in the training argument of prep.recipe. bake.recipe then applies the new values to new data sets using these values. If the training set data has more than one mode, one is selected at random.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and model (the mode value).

Examples

```
data("credit_data")

## missing data per column
vapply(credit_data, function(x) mean(is.na(x)), c(num = 0))

set.seed(342)
in_training <- sample(1:nrow(credit_data), 2000)

credit_tr <- credit_data[ in_training, ]
credit_te <- credit_data[-in_training, ]
missing_examples <- c(14, 394, 565)

rec <- recipe(Price ~ ., data = credit_tr)

impute_rec <- rec %>%
  step_modeimpute(Status, Home, Marital)

imp_models <- prep(impute_rec, training = credit_tr)

imputed_te <- bake(imp_models, newdata = credit_te, everything())

table(credit_te$Home, imputed_te$Home, useNA = "always")

tidy(impute_rec, number = 1)
tidy(imp_models, number = 1)
```

step_novel *Simple Value Assignments for Novel Factor Levels*

Description

step_novel creates a *specification* of a recipe step that will assign a previously unseen factor level to a new value.

Usage

```
step_novel(recipe, ..., role = NA, trained = FALSE, new_level = "new",
  objects = NULL, skip = FALSE)

## S3 method for class 'step_novel'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables that will be affected by the step. These variables should be character or factor types. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
new_level	A single character value that will be assigned to new factor levels.
objects	A list of objects that contain the information on factor levels that will be determined by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_novel object.

Details

The selected variables are adjusted to have a new level (given by new_level) that is placed in the last position. During preparation there will be no data points associated with this new level since all of the data have been seen.

Note that if the original columns are character, they will be converted to factors by this step.

Missing values will remain missing.

If new_level is already in the data given to prep, an error is thrown.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the columns that will be affected) and `value` (the factor levels that is used for the new value)

See Also

`step_factor2string()`, `step_string2factor()`, `dummy_names()`, `step_regex()`, `step_count()`, `step_ordinalscore()`, `step_unorder()`, `step_other()`

Examples

```
data(okc)

okc_tr <- okc[1:30000,]
okc_te <- okc[30001:30006,]
okc_te$diet[3] <- "cannibalism"
okc_te$diet[4] <- "vampirism"

rec <- recipe(~ diet + location, data = okc_tr)

rec <- rec %>%
  step_novel(diet, location)
rec <- prep(rec, training = okc_tr)

processed <- bake(rec, okc_te)
tibble(old = okc_te$diet, new = processed$diet)

tidy(rec, number = 1)
```

---

step_ns *Nature Spline Basis Functions*

Description

`step_ns` creates a *specification* of a recipe step that will create new columns that are basis expansions of variables using natural splines.

Usage

```
step_ns(recipe, ..., role = "predictor", trained = FALSE, objects = NULL,
  options = list(df = 2), skip = FALSE)

## S3 method for class 'step_ns'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new columns created from the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
objects	A list of splines::ns() objects created once the step has been trained.
options	A list of options for splines::ns() which should not include x.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_ns object.

Details

step_ns can new features from a single variable that enable fitting routines to model this variable in a nonlinear manner. The extent of the possible nonlinearity is determined by the df or knot arguments of splines::ns(). The original variables are removed from the data and new columns are added. The naming convention for the new variables is varname_ns_1 and so on.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected and holiday.

See Also

step_poly() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

with_splines <- rec %>%
```

```
  step_ns(carbon, hydrogen)
with_splines <- prep(with_splines, training = biomass_tr)

expanded <- bake(with_splines, biomass_te)
expanded
```

---

step_num2factor *Convert Numbers to Factors*

Description

step_num2factor will convert one or more numeric vectors to factors (ordered or unordered). This can be useful when categories are encoded as integers.

Usage

```
step_num2factor(recipe, ..., role = NA, transform = function(x) x,
  trained = FALSE, levels = NULL, ordered = FALSE, skip = FALSE)

## S3 method for class 'step_num2factor'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will converted to factors. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
transform	A function taking a single argument x that can be used to modify the numeric values prior to determining the levels (perhaps using base::paste() or base::format()).
trained	A logical to indicate if the quantities for preprocessing have been estimated.
levels	A list of values that will be used as the levels. These are the numeric data converted to character and ordered. This is NULL until computed by prep.recipe().
ordered	A single logical value; should the factor(s) be ordered?
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_num2factor object.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables selected) and `ordered`.

Note

If `bake` is used on a data set where a new value is in the column being converted, `bake` will silently give values of `NA` to these rows (see the example below).

See Also

`step_factor2string()`, `step_string2factor()`, `step_dummy()`

Examples

```
iris2 <- iris
iris2$Species <- as.numeric(iris2$Species)

rec <- recipe(~ ., data = iris2)

make_factor <- rec %>% step_num2factor(Species)
make_factor <- prep(make_factor,
                    training = iris2,
                    retain = TRUE)

# note that `diet` is a factor
juice(make_factor) %>% head
okc %>% head
tidy(make_factor, number = 1)

# When novel values are exposed
with_transform <- rec %>%
  step_num2factor(Species, transform = function(x) paste0("val_", x))

with_transform <- prep(with_transform,
                       training = iris2[1:75,])
new_values <- bake(with_transform, newdata = iris2, Species)
table(new_values[["Species"]], iris2$Species, useNA = "ifany")
```

---

step_nzv *Near-Zero Variance Filter*

Description

`step_nzv` creates a *specification* of a recipe step that will potentially remove variables that are highly sparse and unbalanced.

Usage

```
step_nzv(recipe, ..., role = NA, trained = FALSE, options = list(freq_cut
  = 95/5, unique_cut = 10), removals = NULL, skip = FALSE)

## S3 method for class 'step_nzv'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables that will evaluated by the filtering. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
options	A list of options for the filter (see Details below).
removals	A character string that contains the names of columns that should be removed. These values are not determined until prep.recipe() is called.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_nzv object.

Details

This step diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics:

1. they have very few unique values relative to the number of samples and
2. the ratio of the frequency of the most common value to the frequency of the second most common value is large.

For example, an example of near zero variance predictor is one that, for 1000 samples, has two distinct values and 999 of them are a single value.

To be flagged, first the frequency of the most prevalent value over the second most frequent value (called the "frequency ratio") must be above freq_cut. Secondly, the "percent of unique values," the number of unique values divided by the total number of samples (times 100), must also be below unique_cut.

In the above example, the frequency ratio is 999 and the unique value percentage is 0.0001.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be removed.

See Also

step_corr() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass$sparse <- c(1, rep(0, nrow(biomass) - 1))

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen +
                    nitrogen + sulfur + sparse,
              data = biomass_tr)

nzv_filter <- rec %>%
  step_nzv(all_predictors())

filter_obj <- prep(nzv_filter, training = biomass_tr)

filtered_te <- bake(filter_obj, biomass_te)
any(names(filtered_te) == "sparse")

tidy(nzv_filter, number = 1)
tidy(filter_obj, number = 1)
```

---

step_ordinalscore *Convert Ordinal Factors to Numeric Scores*

Description

step_ordinalscore creates a *specification* of a recipe step that will convert ordinal factor variables into numeric scores.

Usage

```
step_ordinalscore(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  convert = as.numeric, skip = FALSE)

## S3 method for class 'step_ordinalscore'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.

...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variables that will be converted. This is NULL until computed by prep.recipe().
convert	A function that takes an ordinal factor vector as an input and outputs a single numeric variable.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_ordinalscore object.

Details

Dummy variables from ordered factors with C levels will create polynomial basis functions with C-1 terms. As an alternative, this step can be used to translate the ordered levels into a single numeric vector of values that represent (subjective) scores. By default, the translation uses a linear scale (1, 2, 3, ... C) but custom score functions can also be used (see the example below).

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the columns that will be affected).

Examples

```
fail_lvls <- c("meh", "annoying", "really_bad")

ord_data <-
  data.frame(item = c("paperclip", "twitter", "airbag"),
             fail_severity = factor(fail_lvls,
                                    levels = fail_lvls,
                                    ordered = TRUE))

model.matrix(~fail_severity, data = ord_data)

linear_values <- recipe(~ item + fail_severity, data = ord_data) %>%
  step_dummy(item) %>%
  step_ordinalscore(fail_severity)

linear_values <- prep(linear_values, training = ord_data, retain = TRUE)

juice(linear_values, everything())

custom <- function(x) {
```

```
  new_values <- c(1, 3, 7)
  new_values[as.numeric(x)]
}

nonlin_scores <- recipe(~ item + fail_severity, data = ord_data) %>%
  step_dummy(item) %>%
  step_ordinalscore(fail_severity, convert = custom)

tidy(nonlin_scores, number = 2)

nonlin_scores <- prep(nonlin_scores, training = ord_data, retain = TRUE)

juice(nonlin_scores, everything())

tidy(nonlin_scores, number = 2)
```

`step_other` *Collapse Some Categorical Levels*

Description

`step_other` creates a *specification* of a recipe step that will potentially pool infrequently occurring values into an "other" category.

Usage

```
step_other(recipe, ..., role = NA, trained = FALSE, threshold = 0.05,
  other = "other", objects = NULL, skip = FALSE)

## S3 method for class 'step_other'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables that will potentially be reduced. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`threshold`	A single numeric value in (0, 1) for pooling.
`other`	A single character value for the "other" category.
`objects`	A list of objects that contain the information to pool infrequent levels that is determined by `prep.recipe()`.

skip — A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

x — A step_other object.

Details

The overall proportion of the categories are computed. The "other" category is used in place of any categorical levels whose individual proportion in the training set is less than threshold.

If no pooling is done the data are unmodified (although character data may be changed to factors based on the value of stringsAsFactors in prep.recipe()). Otherwise, a factor is always returned with different factor levels.

If threshold is less than the largest category proportion, all levels except for the most frequent are collapsed to the other level.

If the retained categories include the value of other, an error is thrown. If other is in the list of discarded levels, no error occurs.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the columns that will be affected) and retained (the factor levels that were not pulled into "other")

See Also

step_factor2string(), step_string2factor(), dummy_names(), step_regex(), step_count(), step_ordinalscore(), step_unorder(), step_novel()

Examples

```
data(okc)

set.seed(19)
in_train <- sample(1:nrow(okc), size = 30000)

okc_tr <- okc[ in_train,]
okc_te <- okc[-in_train,]

rec <- recipe(~ diet + location, data = okc_tr)


rec <- rec %>%
  step_other(diet, location, threshold = .1, other = "other values")
rec <- prep(rec, training = okc_tr)

collapsed <- bake(rec, okc_te)
table(okc_te$diet, collapsed$diet, useNA = "always")
```

```
tidy(rec, number = 1)
```

`step_pca` *PCA Signal Extraction*

Description

`step_pca` creates a *specification* of a recipe step that will convert numeric data into one or more principal components.

Usage

```
step_pca(recipe, ..., role = "predictor", trained = FALSE, num = 5,
  threshold = NA, options = list(), res = NULL, prefix = "PC",
  skip = FALSE)

## S3 method for class 'step_pca'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will be used to compute the components. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new principal component columns created by the original variables will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`num`	The number of PCA components to retain as new predictors. If `num` is greater than the number of columns or the number of possible components, a smaller value will be used.
`threshold`	A fraction of the total variance that should be covered by the components. For example, `threshold = .75` means that `step_pca` should generate enough components to capture 75% of the variability in the variables. Note: using this argument will override and resent any value given to `num`.
`options`	A list of options to the default method for `stats::prcomp()`. Argument defaults are set to `retx = FALSE`, `center = FALSE`, `scale. = FALSE`, and `tol = NULL`. **Note** that the argument `x` should not be passed here (or at all).
`res`	The `stats::prcomp.default()` object is stored here once this preprocessing step has be trained by `prep.recipe()`.
`prefix`	A character string that will be the prefix to the resulting new variables. See notes below

skip — A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

x — A step_pca object.

Details

Principal component analysis (PCA) is a transformation of a group of variables that produces a new set of artificial features or components. These components are designed to capture the maximum amount of information (i.e. variance) in the original variables. Also, the components are statistically independent from one another. This means that they can be used to combat large inter-variables correlations in a data set.

It is advisable to standardized the variables prior to running PCA. Here, each variable will be centered and scaled prior to the PCA calculation. This can be changed using the options argument or by using step_center() and step_scale().

The argument num controls the number of components that will be retained (the original variables that are used to derive the components are removed from the data). The new components will have names that begin with prefix and a sequence of numbers. The variable names are padded with zeros. For example, if num < 10, their names will be PC1 - PC9. If num = 101, the names would be PC001 - PC101.

Alternatively, threshold can be used to determine the number of components that are required to capture a specified fraction of the total variance in the variables.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected), value (the loading), and component.

References

Jolliffe, I. T. (2010). *Principal Component Analysis*. Springer.

See Also

step_ica() step_kpca() step_isomap() recipe() prep.recipe() bake.recipe()

Examples

```
rec <- recipe( ~ ., data = USArrests)
pca_trans <- rec %>%
  step_center(all_numeric()) %>%
  step_scale(all_numeric()) %>%
  step_pca(all_numeric(), num = 3)
pca_estimates <- prep(pca_trans, training = USArrests)
pca_data <- bake(pca_estimates, USArrests)
```

```
rng <- extendrange(c(pca_data$PC1, pca_data$PC2))
plot(pca_data$PC1, pca_data$PC2,
     xlim = rng, ylim = rng)

with_thresh <- rec %>%
  step_center(all_numeric()) %>%
  step_scale(all_numeric()) %>%
  step_pca(all_numeric(), threshold = .99)
with_thresh <- prep(with_thresh, training = USArrests)
bake(with_thresh, USArrests)

tidy(pca_trans, number = 3)
tidy(pca_estimates, number = 3)
```

---

step_poly *Orthogonal Polynomial Basis Functions*

Description

step_poly creates a *specification* of a recipe step that will create new columns that are basis expansions of variables using orthogonal polynomials.

Usage

```
step_poly(recipe, ..., role = "predictor", trained = FALSE,
  objects = NULL, options = list(degree = 2), skip = FALSE)

## S3 method for class 'step_poly'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the new columns created from the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
objects	A list of stats::poly() objects created once the step has been trained.
options	A list of options for stats::poly() which should not include x or simple. Note that the option raw = TRUE will produce the regular polynomial values (not orthogonalized).

skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_poly object.

Details

step_poly can new features from a single variable that enable fitting routines to model this variable in a nonlinear manner. The extent of the possible nonlinearity is determined by the degree argument of stats::poly(). The original variables are removed from the data and new columns are added. The naming convention for the new variables is varname_poly_1 and so on.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the columns that will be affected) and degree.

See Also

step_ns() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

quadratic <- rec %>%
  step_poly(carbon, hydrogen)
quadratic <- prep(quadratic, training = biomass_tr)

expanded <- bake(quadratic, biomass_te)
expanded

tidy(quadratic, number = 1)
```

---

step_profile *Create a Profiling Version of a Data Set*

Description

step_profile creates a *specification* of a recipe step that will fix the levels of all variables but one and will create a sequence of values for the remaining variable. This step can be helpful when creating partial regression plots for additive models.

Usage

```
step_profile(recipe, ..., profile = NULL, pct = 0.5, index = 1,
  grid = list(pctl = TRUE, len = 100), columns = NULL, role = NA,
  trained = FALSE, skip = FALSE)

## S3 method for class 'step_profile'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will fixed to a single value. See selections() for more details. For the tidy method, these are not currently used.
profile	A call to dplyr::vars()) to specify which variable will be profiled (see selections()). If a column is included in both lists to be fixed and to be profiled, an error is thrown.
pct	A value between 0 and 1 that is the percentile to fix continuous variables. This is applied to all continuous variables captured by the selectors. For date variables, either the minimum, median, or maximum used based on their distance to pct.
index	The level that qualitative variables will be fixed. If the variables are character (not factors), this will be the index of the sorted unique values. This is applied to all qualitative variables captured by the selectors.
grid	A named list with elements pctl (a logical) and len (an integer). If pctl = TRUE, then len denotes how many percentiles to use to create the profiling grid. This creates a grid between 0 and 1 and the profile is determined by the percentiles of the data. For example, if pctl = TRUE and len = 3, the profile would contain the minimum, median, and maximum values. If pctl = FALSE, it defines how many grid points between the minimum and maximum values should be created. This parameter is ignored for qualitative variables (since all of their possible levels are profiled). In the case of date variables, pctl = FALSE will always be used since there is no quantile method for dates.
columns	A character string that contains the names of columns that should be fixed and their values. These values are not determined until prep.recipe() is called.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_profile object.

Details

This step is atypical in that, when baked, the `newdata` argument is ignored; the resulting data set is based on the fixed and profiled variable's information.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (which is the columns that will be affected), and `type` (fixed or profiled).

Examples

```
data(okc)

# Setup a grid across date but keep the other values fixed
recipe(~ diet + height + date, data = okc) %>%
  step_profile(-date, profile = vars(date)) %>%
  prep(training = okc, retain = TRUE) %>%
  juice


##########

# An *additive* model; not for use when there are interactions or
# other functional relationships between predictors

lin_mod <- lm(mpg ~ poly(disp, 2) + cyl + hp, data = mtcars)

# Show the difference in the two grid creation methods

disp_pctl <- recipe(~ disp + cyl + hp, data = mtcars) %>%
  step_profile(-disp, profile = vars(disp)) %>%
  prep(training = mtcars, retain = TRUE)

disp_grid <- recipe(~ disp + cyl + hp, data = mtcars) %>%
  step_profile(
    -disp,
    profile = vars(disp),
    grid = list(pctl = FALSE, len = 100)
  ) %>%
  prep(training = mtcars, retain = TRUE)

grid_data <- juice(disp_grid)
grid_data <- grid_data %>%
  mutate(pred = predict(lin_mod, grid_data),
         method = "grid")

pctl_data <- juice(disp_pctl)
pctl_data <- pctl_data %>%
  mutate(pred = predict(lin_mod, pctl_data),
         method = "percentile")
```

```
plot_data <- bind_rows(grid_data, pctl_data)

library(ggplot2)

ggplot(plot_data, aes(x = disp, y = pred)) +
  geom_point(alpha = .5, cex = 1) +
  facet_wrap(~ method)
```

---

step_range *Scaling Numeric Data to a Specific Range*

Description

step_range creates a *specification* of a recipe step that will normalize numeric data to be within a pre-defined range of values.

Usage

```
step_range(recipe, ..., role = NA, trained = FALSE, min = 0, max = 1,
  ranges = NULL, skip = FALSE)

## S3 method for class 'step_range'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be scaled. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
min	A single numeric value for the smallest value in the range
max	A single numeric value for the largest value in the range
ranges	A character vector of variables that will be normalized. Note that this is ignored until the values are determined by prep.recipe(). Setting this value will be ineffective.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_range object.

Details

When a new data point is outside of the ranges seen in the training set, the new values are truncated at `min` or `max`.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables selected), `min`, and `max`.

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

ranged_trans <- rec %>%
  step_range(carbon, hydrogen)

ranged_obj <- prep(ranged_trans, training = biomass_tr)

transformed_te <- bake(ranged_obj, biomass_te)

biomass_te[1:10, names(transformed_te)]
transformed_te

tidy(ranged_trans, number = 1)
tidy(ranged_obj, number = 1)
```

---

step_ratio *Ratio Variable Creation*

Description

`step_ratio` creates a a *specification* of a recipe step that will create one or more ratios out of numeric variables.

Usage

```
step_ratio(recipe, ..., role = "predictor", trained = FALSE,
  denom = denom_vars(), naming = function(numer, denom)
  make.names(paste(numer, denom, sep = "_o_")), columns = NULL,
  skip = FALSE)
```

```
denom_vars(...)

## S3 method for class 'step_ratio'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be used in the *numerator* of the ratio. When used with denom_vars, the dots indicates which variables are used in the *denominator*. See selections() for more details. For the tidy method, these are not currently used.
role	For terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the newly created ratios created by the original variables will be used as predictors in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
denom	A call to denom_vars to specify which variables are used in the denominator that can include specific variable names separated by commas or different selectors (see selections()). If a column is included in both lists to be numerator and denominator, it will be removed from the listing.
naming	A function that defines the naming convention for new ratio columns.
columns	The column names used in the ratios. This argument is not populated until prep.recipe() is executed.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_ratio object

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and denom.

Examples

```
library(recipes)
data(biomass)

biomass$total <- apply(biomass[, 3:7], 1, sum)
biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen +
                    sulfur + total,
              data = biomass_tr)
```

```
ratio_recipe <- rec %>%
  # all predictors over total
  step_ratio(all_predictors(), denom = denom_vars(total)) %>%
  # get rid of the original predictors
  step_rm(all_predictors(), -ends_with("total"))

ratio_recipe <- prep(ratio_recipe, training = biomass_tr)

ratio_data <- bake(ratio_recipe, biomass_te)
ratio_data
```

`step_regex` *Create Dummy Variables using Regular Expressions*

Description

`step_regex` creates a *specification* of a recipe step that will create a new dummy variable based on a regular expression.

Usage

```
step_regex(recipe, ..., role = "predictor", trained = FALSE,
  pattern = ".", options = list(), result = make.names(pattern),
  input = NULL, skip = FALSE)

## S3 method for class 'step_regex'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	A single selector functions to choose which variable will be searched for the pattern. The selector should resolve into a single variable. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	For a variable created by this step, what analysis role should they be assigned?. By default, the function assumes that the new dummy variable column created by the original variable will be used as a predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`pattern`	A character string containing a regular expression (or character string for `fixed = TRUE`) to be matched in the given character vector. Coerced by `as.character` to a character string if possible.
`options`	A list of options to `grepl()` that should not include `x` or `pattern`.
`result`	A single character value for the name of the new variable. It should be a valid column name.

input	A single character value for the name of the variable being searched. This is NULL until computed by prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_regex object.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and result (the new column name).

Examples

```
data(covers)

rec <- recipe(~ description, covers) %>%
  step_regex(description, pattern = "(rock|stony)", result = "rocks") %>%
  step_regex(description, pattern = "ratake families")

rec2 <- prep(rec, training = covers)
rec2

with_dummies <- bake(rec2, newdata = covers)
with_dummies
tidy(rec, number = 1)
tidy(rec2, number = 1)
```

---

step_relu *Apply (Smoothed) Rectified Linear Transformation*

Description

step_relu creates a *specification* of a recipe step that will apply the rectified linear or softplus transformations to numeric data. The transformed data is added as new columns to the data matrix.

Usage

```
step_relu(recipe, ..., role = "predictor", trained = FALSE, shift = 0,
  reverse = FALSE, smooth = FALSE, prefix = "right_relu_", skip = FALSE)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
shift	A numeric value dictating a translation to apply to the data.
reverse	A logical to indicate if the left hinge should be used as opposed to the right hinge.
smooth	A logical indicating if the softplus function, a smooth appromixation to the rectified linear transformation, should be used.
prefix	A prefix for generated column names, default to "right_relu_" when right hinge transformation and "left_relu_" for reversed/left hinge transformations.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

Details

The rectified linear transformation is calculated as

$$max(0, x - c)$$

and is also known as the ReLu or right hinge function. If reverse is true, then the transformation is reflected about the y-axis, like so:

$$max(0, c - x)$$

Setting the smooth option to true will instead calculate a smooth approximation to ReLu according to

$$ln(1 + e^(x - c)$$

The reverse argument may also be applied to this transformation.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any).

Connection to MARS

The rectified linear transformation is used in the Multivariate Adaptive Regression Splines as a basis function to fit piecewise linear functions to data in a strategy similar to that employeed in tree based models. The transformation is a popular choice as an activation function in many neural networks, which could then be seen as a stacked generalization of MARS when making use of ReLu activations. The hinge function also appears in the loss function of Support Vector Machines, where it penalizes residuals only if they are within a certain margin of the decision boundary.

See Also

recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

transformed_te <- rec %>%
  step_relu(carbon, shift = 40) %>%
  prep(biomass_tr) %>%
  bake(biomass_te)

transformed_te
```

---

step_rm *General Variable Filter*

Description

step_rm creates a *specification* of a recipe step that will remove variables based on their name, type, or role.

Usage

```
step_rm(recipe, ..., role = NA, trained = FALSE, removals = NULL,
  skip = FALSE)

## S3 method for class 'step_rm'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables that will evaluated by the filtering bake. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.

removals	A character string that contains the names of columns that should be removed. These values are not determined until prep.recipe() is called.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_rm object.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be removed.

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

library(dplyr)
smaller_set <- rec %>%
  step_rm(contains("gen"))

smaller_set <- prep(smaller_set, training = biomass_tr)

filtered_te <- bake(smaller_set, biomass_te)
filtered_te

tidy(smaller_set, number = 1)
```

---

step_scale *Scaling Numeric Data*

Description

step_scale creates a *specification* of a recipe step that will normalize numeric data to have a standard deviation of one.

Usage

```
step_scale(recipe, ..., role = NA, trained = FALSE, sds = NULL,
  na.rm = TRUE, skip = FALSE)

## S3 method for class 'step_scale'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
sds	A named numeric vector of standard deviations This is NULL until computed by prep.recipe().
na.rm	A logical value indicating whether NA values should be removed when computing the standard deviation.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_scale object.

Details

Scaling data means that the standard deviation of a variable is divided out of the data. step_scale estimates the variable standard deviations from the data used in the training argument of prep.recipe. bake.recipe then applies the scaling to new data sets using these standard deviations.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value (the standard deviations).

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

scaled_trans <- rec %>%
  step_scale(carbon, hydrogen)

scaled_obj <- prep(scaled_trans, training = biomass_tr)

transformed_te <- bake(scaled_obj, biomass_te)
```

```
biomass_te[1:10, names(transformed_te)]
transformed_te
tidy(scaled_trans, number = 1)
tidy(scaled_obj, number = 1)
```

`step_shuffle` *Shuffle Variables*

Description

`step_shuffle` creates a *specification* of a recipe step that will randomly change the order of rows for selected variables.

Usage

```
step_shuffle(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'step_shuffle'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will permuted. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`columns`	A character string that contains the names of columns that should be shuffled. These values are not determined until `prep.recipe()` is called.
`skip`	A logical. Should the step be skipped when the recipe is baked by `bake.recipe()`? While all operations are baked when `prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`x`	A `step_shuffle` object.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` which is the columns that will be affected.

Examples

```
integers <- data.frame(A = 1:12, B = 13:24, C = 25:36)

library(dplyr)
rec <- recipe(~ A + B + C, data = integers) %>%
  step_shuffle(A, B)

rand_set <- prep(rec, training = integers)

set.seed(5377)
bake(rand_set, integers)

tidy(rec, number = 1)
tidy(rand_set, number = 1)
```

---

step_spatialsign *Spatial Sign Preprocessing*

Description

step_spatialsign is a *specification* of a recipe step that will convert numeric data into a projection on to a unit sphere.

Usage

```
step_spatialsign(recipe, ..., role = "predictor", na.rm = TRUE,
  trained = FALSE, columns = NULL, skip = FALSE)

## S3 method for class 'step_spatialsign'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be used for the normalization. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned?
na.rm	A logical: should missing data be removed from the norm computation?
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variable names that will be populated (eventually) by the terms argument.

skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_spatialsign object.

Details

The spatial sign transformation projects the variables onto a unit sphere and is related to global contrast normalization. The spatial sign of a vector w is w/norm(w).

The variables should be centered and scaled prior to the computations.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected.

References

Serneels, S., De Nolf, E., and Van Espen, P. (2006). Spatial sign preprocessing: a simple way to impart moderate robustness to multivariate estimators. *Journal of Chemical Information and Modeling*, 46(3), 1402-1409.

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

ss_trans <- rec %>%
  step_center(carbon, hydrogen) %>%
  step_scale(carbon, hydrogen) %>%
  step_spatialsign(carbon, hydrogen)

ss_obj <- prep(ss_trans, training = biomass_tr)

transformed_te <- bake(ss_obj, biomass_te)

plot(biomass_te$carbon, biomass_te$hydrogen)

plot(transformed_te$carbon, transformed_te$hydrogen)

tidy(ss_trans, number = 3)
tidy(ss_obj, number = 3)
```

step_sqrt *Square Root Transformation*

Description

step_sqrt creates a *specification* of a recipe step that will square root transform the data.

Usage

```
step_sqrt(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'step_sqrt'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables will be transformed. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_sqrt object.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the columns that will be affected.

See Also

step_logit() step_invlogit() step_log() step_hyperbolic() recipe() prep.recipe() bake.recipe()

Examples

```
set.seed(313)
examples <- matrix(rnorm(40)^2, ncol = 2)
examples <- as.data.frame(examples)

rec <- recipe(~ V1 + V2, data = examples)

sqrt_trans <- rec  %>%
  step_sqrt(all_predictors())

sqrt_obj <- prep(sqrt_trans, training = examples)

transformed_te <- bake(sqrt_obj, examples)
plot(examples$V1, transformed_te$V1)

tidy(sqrt_trans, number = 1)
tidy(sqrt_obj, number = 1)
```

---

step_string2factor *Convert Strings to Factors*

Description

`step_string2factor` will convert one or more character vectors to factors (ordered or unordered).

Usage

```
step_string2factor(recipe, ..., role = NA, trained = FALSE, levels = NULL,
  ordered = FALSE, skip = FALSE)

## S3 method for class 'step_string2factor'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables will converted to factors. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`levels`	An options specification of the levels to be used for the new factor. If left `NULL`, the sorted unique values present when `bake` is called will be used.
`ordered`	A single logical value; should the factor(s) be ordered?

skip — A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

x — A step_string2factor object.

Details

If levels is given, step_string2factor will convert all factors to have the same levels. Also, note that prep has an option stringsAsFactors that defaults to TRUE. This should be changed so that raw character data will be applied to step_string2factor.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and ordered.

See Also

step_factor2string() step_dummy() step_other() step_novel()

Examples

```
data(okc)

rec <- recipe(~ diet + location, data = okc)

make_factor <- rec %>%
  step_string2factor(diet)
make_factor <- prep(make_factor,
                    training = okc,
                    stringsAsFactors = FALSE,
                    retain = TRUE)

# note that `diet` is a factor
juice(make_factor) %>% head
okc %>% head
tidy(make_factor, number = 1)
```

---

step_unorder *Convert Ordered Factors to Unordered Factors*

Description

step_unorder creates a *specification* of a recipe step that will transform the data.

Usage

```
step_unorder(recipe, ..., role = NA, trained = FALSE, columns = NULL,
  skip = FALSE)

## S3 method for class 'step_unorder'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_unorder object.

Details

The factors level order is preserved during the transformation.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the columns that will be affected).

See Also

step_ordinalscore() recipe() prep.recipe() bake.recipe()

Examples

```
lmh <- c("Low", "Med", "High")

examples <- data.frame(X1 = factor(rep(letters[1:4], each = 3)),
                       X2 = ordered(rep(lmh, each = 4),
                                    levels = lmh))

rec <- recipe(~ X1 + X2, data = examples)
```

```
factor_trans <- rec  %>%
  step_unorder(all_predictors())

factor_obj <- prep(factor_trans, training = examples)

transformed_te <- bake(factor_obj, examples)
table(transformed_te$X2, examples$X2)

tidy(factor_trans, number = 1)
tidy(factor_obj, number = 1)
```

`step_upsample` *Up-Sample a Data Set Based on a Factor Variable*

Description

`step_upsample` creates a *specification* of a recipe step that will replicate rows of a data set to make the occurrence of levels in a specific factor level equal.

Usage

```
step_upsample(recipe, ..., ratio = 1, role = NA, trained = FALSE,
  column = NULL, target = NA, skip = TRUE)

## S3 method for class 'step_upsample'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variable is used to sample the data. See `selections()` for more details. The selection should result in *single factor variable*. For the `tidy` method, these are not currently used.
`ratio`	A numeric value for the ratio of the majority-to-minority frequencies. The default value (1) means that all other levels are sampled up to have the same frequency as the most occurring level. A value of 0.5 would mean that the minority levels will have (at most) (approximately) half as many rows than the majority level.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`column`	A character string of the variable name that will be populated (eventually) by the `...` selectors.
`target`	An integer that will be used to subsample. This should not be set by the user and will be populated by `prep`.

skip — A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

x — A step_upsample object.

Details

Up-sampling is intended to be performed on the *training* set alone. For this reason, the default is skip = TRUE. It is advisable to use prep(recipe, retain = TRUE) when preparing the recipe; in this way juice() can be used to obtain the up-sampled version of the data.

If there are missing values in the factor variable that is used to define the sampling, missing data are selected at random in the same way that the other factor levels are sampled. Missing values are not used to determine the amount of data in the majority level (see example below).

For any data with factor levels occurring with the same frequency as the majority level, all data will be retained.

All columns in the data are sampled and returned by juice() and bake().

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms which is the variable used to sample.

Examples

```
data(okc)

orig <- table(okc$diet, useNA = "always")

sort(orig, decreasing = TRUE)

up_rec <- recipe( ~ ., data = okc) %>%
  # Bring the minority levels up to about 200 each
  # 200/16562 is approx 0.0121
  step_upsample(diet, ratio = 0.0121) %>%
  prep(training = okc, retain = TRUE)

training <- table(juice(up_rec)$diet, useNA = "always")

# Since `skip` defaults to TRUE, baking the step has no effect
baked_okc <- bake(up_rec, newdata = okc)
baked <- table(baked_okc$diet, useNA = "always")

# Note that if the original data contained more rows than the
# target n (= ratio * majority_n), the data are left alone:
data.frame(
  level = names(orig),
  orig_freq = as.vector(orig),
  train_freq = as.vector(training),
```

```
  baked_freq = as.vector(baked)
)
```

---

step_window *Moving Window Functions*

Description

step_window creates a *specification* of a recipe step that will create new columns that are the results of functions that compute statistics across moving windows.

Usage

```
step_window(recipe, ..., role = NA, trained = FALSE, size = 3,
  na.rm = TRUE, statistic = "mean", columns = NULL, names = NULL,
  skip = FALSE)

## S3 method for class 'step_window'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	For model terms created by this step, what analysis role should they be assigned? If names is left to be NULL, the rolling statistics replace the original columns and the roles are left unchanged. If names is set, those new columns will have a role of NULL unless this argument has a value.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
size	An odd integer >= 3 for the window size.
na.rm	A logical for whether missing values should be removed from the calculations within each window.
statistic	A character string for the type of statistic that should be calculated for each moving window. Possible values are: 'max', 'mean', 'median', 'min', 'prod', 'sd', 'sum', 'var'
columns	A character string that contains the names of columns that should be processed. These values are not determined until prep.recipe() is called.
names	An optional character string that is the same length of the number of terms selected by terms. If you are not sure what columns will be selected, use the summary function (see the example below). These will be the names of the new columns created by the step.

skip — A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations

x — A step_window object.

Details

The calculations use a somewhat atypical method for handling the beginning and end parts of the rolling statistics. The process starts with the center justified window calculations and the beginning and ending parts of the rolling values are determined using the first and last rolling values, respectively. For example if a column x with 12 values is smoothed with a 5-point moving median, the first three smoothed values are estimated by median(x[1:5]) and the fourth uses median(x[2:6]).

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and statistic (the summary function name), and size.

Examples

```
library(recipes)
library(dplyr)
library(rlang)
library(ggplot2, quietly = TRUE)

set.seed(5522)
sim_dat <- data.frame(x1 = (20:100) / 10)
n <- nrow(sim_dat)
sim_dat$y1 <- sin(sim_dat$x1) + rnorm(n, sd = 0.1)
sim_dat$y2 <- cos(sim_dat$x1) + rnorm(n, sd = 0.1)
sim_dat$x2 <- runif(n)
sim_dat$x3 <- rnorm(n)

rec <- recipe(y1 + y2 ~ x1 + x2 + x3, data = sim_dat) %>%
  step_window(starts_with("y"), size = 7, statistic = "median",
              names = paste0("med_7pt_", 1:2),
              role = "outcome") %>%
  step_window(starts_with("y"),
              names = paste0("mean_3pt_", 1:2),
              role = "outcome")
rec <- prep(rec, training = sim_dat)

# If you aren't sure how to set the names, see which variables are selected
# and the order that they are selected:
terms_select(info = summary(rec), terms = quos(starts_with("y")))

smoothed_dat <- bake(rec, sim_dat, everything())
```

```
ggplot(data = sim_dat, aes(x = x1, y = y1)) +
  geom_point() +
  geom_line(data = smoothed_dat, aes(y = med_7pt_1)) +
  geom_line(data = smoothed_dat, aes(y = mean_3pt_1), col = "red") +
  theme_bw()

tidy(rec, number = 1)
tidy(rec, number = 2)

# If you want to replace the selected variables with the rolling statistic
# don't set `names`
sim_dat$original <- sim_dat$y1
rec <- recipe(y1 + y2 + original ~ x1 + x2 + x3, data = sim_dat) %>%
  step_window(starts_with("y"))
rec <- prep(rec, training = sim_dat)
smoothed_dat <- bake(rec, sim_dat, everything())
ggplot(smoothed_dat, aes(x = original, y = y1)) +
  geom_point() +
  theme_bw()
```

`step_YeoJohnson` *Yeo-Johnson Transformation*

Description

`step_YeoJohnson` creates a *specification* of a recipe step that will transform data using a simple Yeo-Johnson transformation.

Usage

```
step_YeoJohnson(recipe, ..., role = NA, trained = FALSE, lambdas = NULL,
  limits = c(-5, 5), nunique = 5, na.rm = TRUE, skip = FALSE)

## S3 method for class 'step_YeoJohnson'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables are affected by the step. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`lambdas`	A numeric vector of transformation values. This is `NULL` until computed by `prep.recipe()`.

limits	A length 2 numeric vector defining the range to compute the transformation parameter lambda.
nunique	An integer where data that have less possible values will not be evaluate for a transformation.
na.rm	A logical value indicating whether NA values should be removed during computations.
skip	A logical. Should the step be skipped when the recipe is baked by bake.recipe()? While all operations are baked when prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
x	A step_YeoJohnson object.

Details

The Yeo-Johnson transformation is very similar to the Box-Cox but does not require the input variables to be strictly positive. In the package, the partial log-likelihood function is directly optimized within a reasonable set of transformation values (which can be changed by the user).

This transformation is typically done on the outcome variable using the residuals for a statistical model (such as ordinary least squares). Here, a simple null model (intercept only) is used to apply the transformation to the *predictor* variables individually. This can have the effect of making the variable distributions more symmetric.

If the transformation parameters are estimated to be very closed to the bounds, or if the optimization fails, a value of NA is used and no transformation is applied.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any). For the tidy method, a tibble with columns terms (the selectors or variables selected) and value (the lambda estimate).

References

Yeo, I. K., and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*.

See Also

step_BoxCox() recipe() prep.recipe() bake.recipe()

Examples

```
data(biomass)

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]
```

```
rec <- recipe(HHV ~ carbon + hydrogen + oxygen + nitrogen + sulfur,
              data = biomass_tr)

yj_trans <- step_YeoJohnson(rec,  all_numeric())

yj_estimates <- prep(yj_trans, training = biomass_tr)

yj_te <- bake(yj_estimates, biomass_te)

plot(density(biomass_te$sulfur), main = "before")
plot(density(yj_te$sulfur), main = "after")

tidy(yj_trans, number = 1)
tidy(yj_estimates, number = 1)
```

`step_zv` *Zero Variance Filter*

Description

`step_zv` creates a *specification* of a recipe step that will remove variables that contain only a single value.

Usage

```
step_zv(recipe, ..., role = NA, trained = FALSE, removals = NULL,
  skip = FALSE)

## S3 method for class 'step_zv'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose which variables that will evaluated by the filtering. See `selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`removals`	A character string that contains the names of columns that should be removed. These values are not determined until `prep.recipe()` is called.
`skip`	A logical. Should the step be skipped when the recipe is baked by `bake.recipe()`? While all operations are baked when `prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`x`	A `step_zv` object.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` which is the columns that will be removed.

See Also

`step_nzv()` `step_corr()` `recipe()` `prep.recipe()` `bake.recipe()`

Examples

```
data(biomass)

biomass$one_value <- 1

biomass_tr <- biomass[biomass$dataset == "Training",]
biomass_te <- biomass[biomass$dataset == "Testing",]

rec <- recipe(HHV ~ carbon + hydrogen + oxygen +
                    nitrogen + sulfur + one_value,
              data = biomass_tr)

zv_filter <- rec %>%
  step_zv(all_predictors())

filter_obj <- prep(zv_filter, training = biomass_tr)

filtered_te <- bake(filter_obj, biomass_te)
any(names(filtered_te) == "one_value")

tidy(zv_filter, number = 1)
tidy(filter_obj, number = 1)
```

---

summary.recipe *Summarize a Recipe*

Description

This function prints the current set of variables/features and some of their characteristics.

Usage

```
## S3 method for class 'recipe'
summary(object, original = FALSE, ...)
```

Arguments

`object`	A `recipe` object
`original`	A logical: show the current set of variables or the original set when the recipe was defined.
`...`	further arguments passed to or from other methods (not currently used).

Details

Note that, until the recipe has been trained, the current and original variables are the same.

Value

A tibble with columns `variable`, `type`, `role`, and `source`.

See Also

`recipe()` `prep.recipe()`

Examples

```
rec <- recipe( ~ ., data = USArrests)
summary(rec)
rec <- step_pca(rec, all_numeric(), num = 3)
summary(rec) # still the same since not yet trained
rec <- prep(rec, training = USArrests)
summary(rec)
```

---

terms_select *Select Terms in a Step Function.*

Description

This function bakes the step function selectors and might be useful when creating custom steps.

Usage

```
terms_select(terms, info)
```

Arguments

`terms`	A list of formulas whose right-hand side contains quoted expressions. See `rlang::quos()` for examples.
`info`	A tibble with columns `variable`, `type`, `role`, and `source` that represent the current state of the data. The function `summary.recipe()` can be used to get this information from a recipe.

Value

A character string of column names or an error of there are no selectors or if no variables are selected.

See Also

`recipe()` `summary.recipe()` `prep.recipe()`

Examples

```
library(rlang)
data(okc)
rec <- recipe(~ ., data = okc)
info <- summary(rec)
terms_select(info = info, quos(all_predictors()))
```

`tidy.recipe` *Tidy the Result of a Recipe*

Description

`tidy` will return a data frame that contains information regarding a recipe or operation within the recipe (when a `tidy` method for the operation exists).

Usage

```
## S3 method for class 'recipe'
tidy(x, number = NA, ...)
```

Arguments

`x`	A `recipe` object (trained or otherwise).
`number`	An integer or `NA`. If missing, the return value is a list of the operation in the recipe. If a number is given, a `tidy` method is executed for that operation in the recipe (if it exists).
`...`	Not currently used.

Value

A tibble with columns that would vary depending on what `tidy` method is executed. When `x` is `NA`, a tibble with columns `number` (the operation iteration), `operation` (either "step" or "check"), `type` (the method, e.g. "nzv", "center"), a logical column called `trained` for whether the operation has been estimated using `prep`, and a logical for `skip`.

Examples

```
data(okc)

okc_rec <- recipe(~ ., data = okc) %>%
  step_other(all_nominal(), threshold = 0.05) %>%
  step_date(date, features = "dow") %>%
  step_center(all_numeric()) %>%
  step_dummy(all_nominal()) %>%
  check_cols(starts_with("date"), age, height)

tidy(okc_rec)
```

```
tidy(okc_rec, number = 2)
tidy(okc_rec, number = 3)

okc_rec_trained <- prep(okc_rec, training = okc)

tidy(okc_rec_trained)
tidy(okc_rec_trained, number = 3)
```

Index