Package 'bundle'

May 8, 2026

Title Serialize Model Objects with a Consistent Interface

Version 0.1.3

Description Typically, models in 'R' exist in memory and can be saved via regular 'R' serialization. However, some models store information in locations that cannot be saved using 'R' serialization alone. The goal of 'bundle' is to provide a common interface to capture this information, situate it within a portable object, and restore it for use in new settings.

License MIT + file LICENSE

URL https://github.com/rstudio/bundle, https://rstudio.github.io/bundle/

BugReports https://github.com/rstudio/bundle/issues

Depends R (>= 4.1)

Imports glue, lifecycle, purrr, rlang, utils, withr

Suggests bonsai, butcher (>= 0.4.0), callr, caret, covr, dbarts, embed, h2o, keras, kernlab, knitr, luz, MASS, modeldata, parsnip, recipes, renv, rmarkdown, stacks, tensorflow, testthat (>= 3.0.0), torch, torchvision, uwot, vetiver, workflows, xgboost (>= 1.6.0.1)

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Config/usethis/last-upkeep 2025-12-08

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Julia Silge [aut, cre] (ORCID: <https://orcid.org/0000-0002-3671-836X>), Simon Couch [aut], Qiushi Yan [aut], Max Kuhn [aut], Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Julia Silge <julia.silge@posit.co>

Repository CRAN

Date/Publication 2025-12-10 10:50:08 UTC

Contents

`bundle` *Bundling*

Description

`bundle()` methods provide a consistent interface to serialization methods for statistical model objects. The created bundle can be saved, then re-loaded and `unbundle()`d in a new R session for use in prediction.

Usage

```
bundle(x, ...)

unbundle(x)
```

Arguments

`x`	A model object to bundle.
`...`	Additional arguments to bundle methods.

Value

A bundle object with subclass referencing the modeling function. If a bundle method is not defined for the supplied object, `bundle.default` is the identity function.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

See Also

Other bundlers: `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

`bundle.bart` *Bundle a* `bart` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'bart'
bundle(x, ...)
```

Arguments

x	A bart object returned from dbarts::bart(). Notably, this ought not to be the output of parsnip::bart().
...	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Value

A bundle object with subclass bundled_bart.

Bundles are a list subclass with two components:

object	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
situate	A function. The situate() function is defined when bundle() is called, though is a loose analogue of an unbundle() S3 method for that object. Since the function is defined on bundle(), it has access to references and dependency information that can be saved alongside the object component. Calling unbundle() on a bundled object x calls x$situate(x$object), returning the unserialized version of object. situate() will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with base::saveRDS() is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object x in a new environment, load its bundle with base::readRDS() and run unbundle() on it. The output of unbundle() is a model object that is ready to predict() on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See vignette("bundle") for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model x, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of `unbundle(res)` in a new R session.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(dbarts)

mtcars$vs <- as.factor(mtcars$vs)

set.seed(1)
fit <- dbarts::bart(mtcars[c("disp", "hp")], mtcars$vs, keeptrees = TRUE)

fit_bundle <- bundle(fit)

# then, after saveRDS + readRDS or passing to a new session ----------
fit_unbundled <- unbundle(fit_bundle)

fit_unbundled_preds <- predict(fit_unbundled, mtcars)
```

---

bundle.H2OAutoML *Bundle an* h2o *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'H2OAutoML'
bundle(x, id = NULL, n = NULL, ...)

## S3 method for class 'H2OMultinomialModel'
bundle(x, ...)

## S3 method for class 'H2OBinomialModel'
```

```
bundle(x, ...)

## S3 method for class 'H2ORegressionModel'
bundle(x, ...)
```

Arguments

x	An object returned from modeling functions in the h2o package.
id	A single character. The model_id entry in the leaderboard. Applies to AutoML output only. Supply only one of this argument or n.
n	An integer giving the position in the leaderboard of the model to bundle. Applies to AutoML output only. Will be ignored if id is supplied.
...	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Value

A bundle object with subclass bundled_h2o.

Bundles are a list subclass with two components:

object	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
situate	A function. The situate() function is defined when bundle() is called, though is a loose analogue of an unbundle() S3 method for that object. Since the function is defined on bundle(), it has access to references and dependency information that can be saved alongside the object component. Calling unbundle() on a bundled object x calls x$situate(x$object), returning the unserialized version of object. situate() will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with base::saveRDS() is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object x in a new environment, load its bundle with base::readRDS() and run unbundle() on it. The output of unbundle() is a model object that is ready to predict() on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See vignette("bundle") for more information on bundling and its motivation.

See Also

These methods wrap h2o::h2o.save_mojo() and h2o::h2o.saveModel().

Other bundlers: bundle(), bundle.bart(), bundle.keras.engine.training.Model(), bundle.luz_module_fitted(), bundle.model_fit(), bundle.model_stack(), bundle.recipe(), bundle.step_umap(), bundle.train(), bundle.workflow(), bundle.xgb.Booster()

Examples

```
# fit model and bundle ------------------------------------------------
library(h2o)

set.seed(1)

h2o.init()

cars_h2o <- as.h2o(mtcars)

cars_fit <-
  h2o.glm(
    x = colnames(cars_h2o)[2:11],
    y = colnames(cars_h2o)[1],
    training_frame = cars_h2o
  )

cars_bundle <- bundle(cars_fit)

# then, after saveRDS + readRDS or passing to a new session ----------
cars_unbundled <- unbundle(cars_fit)

predict(cars_unbundled, cars_h2o[, 2:11])

h2o.shutdown(prompt = FALSE)
```

---

bundle.keras.engine.training.Model

Bundle a keras *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'keras.engine.training.Model'
bundle(x, ...)
```

Arguments

x	An object returned from modeling functions in the keras package.
...	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

This bundler does not currently support custom keras extensions, such as use of a `keras::new_layer_class()` or custom metric function. In such situations, consider using `keras::with_custom_object_scope()`.

Value

A bundle object with subclass `bundled_keras`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

See Also

This method wraps `keras::save_model_tf()` and `keras::load_model_tf()`.

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(keras)

set.seed(1)

mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y

x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))

x_train <- x_train / 255
x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)

mod <- keras_model_sequential()

mod |>
  layer_dense(units = 128, activation = 'relu', input_shape = c(784)) |>
  layer_dropout(rate = 0.4) |>
  layer_dense(units = 64, activation = 'relu') |>
  layer_dropout(rate = 0.3) |>
  layer_dense(units = 10, activation = 'softmax')

mod |> compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)

mod |> fit(
  x_train, y_train,
  epochs = 5, batch_size = 128,
  validation_split = 0.2,
  verbose = 0
)

mod_bundle <- bundle(mod)

# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)

predict(mod_unbundled, x_test)
```

`bundle.luz_module_fitted` *Bundle a* `luz_module_fitted` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'luz_module_fitted'
bundle(x, ...)
```

Arguments

`x`	A `luz_module_fitted` object returned from `luz::fit.luz_module_generator()`.
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

For now, bundling methods for torch are only available via the luz package, "a higher level API for torch providing abstractions to allow for much less verbose training loops."

Value

A bundle object with subclass `bundled_luz_module_fitted`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()`

on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

See Also

This method wraps `luz::luz_save()` and `luz::luz_load()`.

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
if (torch::torch_is_installed()) {
# fit model and bundle ------------------------------------------------
library(torch)
library(torchvision)
library(luz)

set.seed(1)

# example adapted from luz pkgdown article "Autoencoder"
dir <- tempdir()

mnist_dataset2 <- torch::dataset(
  inherit = mnist_dataset,
  .getitem = function(i) {
    output <- super$.getitem(i)
    output$y <- output$x
    output
  }
)

train_ds <- mnist_dataset2(
  dir,
  download = TRUE,
  transform = transform_to_tensor
)

test_ds <- mnist_dataset2(
  dir,
  train = FALSE,
  transform = transform_to_tensor
)

train_dl <- dataloader(train_ds, batch_size = 128, shuffle = TRUE)
```

```
test_dl <- dataloader(test_ds, batch_size = 128)

net <- nn_module(
  "Net",
  initialize = function() {
    self$encoder <- nn_sequential(
      nn_conv2d(1, 6, kernel_size=5),
      nn_relu(),
      nn_conv2d(6, 16, kernel_size=5),
      nn_relu()
    )
    self$decoder <- nn_sequential(
      nn_conv_transpose2d(16, 6, kernel_size = 5),
      nn_relu(),
      nn_conv_transpose2d(6, 1, kernel_size = 5),
      nn_sigmoid()
    )
  },
  forward = function(x) {
    x |>
      self$encoder() |>
      self$decoder()
  },
  predict = function(x) {
    self$encoder(x) |>
      torch_flatten(start_dim = 2)
  }
)

mod <- net |>
  setup(
    loss = nn_mse_loss(),
    optimizer = optim_adam
  ) |>
  fit(train_dl, epochs = 1, valid_data = test_dl)

mod_bundle <- bundle(mod)


# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)

mod_unbundled_preds <- predict(mod_unbundled, test_dl)
}
```

`bundle.model_fit` *Bundle a parsnip* `model_fit` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'model_fit'
bundle(x, ...)
```

Arguments

x	A model_fit object returned from parsnip or other tidymodels packages.
...	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

Primarily, these methods call `bundle()` on the output of `parsnip::extract_fit_engine()`. See the class of the output of that function for more details on the bundling method for that object.

Value

A bundle object with subclass `bundled_model_fit`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model `x`, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of `unbundle(res)` in a new R session.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(parsnip)
library(xgboost)

set.seed(1)

mod <-
  boost_tree(trees = 5, mtry = 3) |>
  set_mode("regression") |>
  set_engine("xgboost") |>
  fit(mpg ~ ., data = mtcars)

mod_bundle <- bundle(mod)

# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)

mod_unbundled_preds <- predict(mod_unbundled, new_data = mtcars)
```

`bundle.model_stack` *Bundle a tidymodels* `model_stack` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'model_stack'
bundle(x, ...)
```

Arguments

`x`	A model_stack object returned from fit_members().
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

This bundler wraps `bundle.model_fit()` and `bundle.workflow()`. Both the fitted members and the meta-learner (in `x$coefs`) are bundled.

Value

A bundle object with subclass `bundled_model_stack`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(stacks)

set.seed(1)

mod <-
  stacks() |>
  add_candidates(reg_res_lr) |>
  add_candidates(reg_res_svm) |>
  blend_predictions(times = 10) |>
  fit_members()

mod_bundle <- bundle(mod)

# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)
```

---

bundle.recipe *Bundle a* `recipe` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

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

Arguments

x	A recipe object returned from recipes.
...	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

The method call `bundle()` on every step in the recipe object. See the classes of individual steps for more details on the bundling method for that object.

Value

A bundle object with subclass `bundled_recipe`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.step_umap()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

`bundle.step_umap` *Bundle a* `step_umap` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'step_umap'
bundle(x, ...)
```

Arguments

`x`	A step_umap object returned from embed.
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Value

A bundle object with subclass `bundled_step_umap`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle

package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model `x`, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of `unbundle(res)` in a new R session.

See Also

This method wraps `uwot::save_uwot()` and `uwot::load_uwot()`.

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.train()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(recipes)
library(embed)

set.seed(1)

rec <- recipe(Species ~ ., data = iris) |>
  step_normalize(all_predictors()) |>
  step_umap(all_predictors(), outcome = vars(Species), num_comp = 2) |>
  prep()

rec_bundle <- bundle(rec)

# then, after saveRDS + readRDS or passing to a new session ----------
rec_unbundled <- unbundle(rec_bundle)

bake(rec_unbundled, new_data = iris)
```

`bundle.train` *Bundle a caret* `train` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'train'
bundle(x, ...)
```

Arguments

`x`	A train object returned from `caret::train()`.
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

Primarily, these methods call `bundle()` on the output of `train_model_object$finalModel`. See the class of the output of that slot for more details on the bundling method for that object.

Value

A bundle object with subclass `bundled_train`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model `x`, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of `unbundle(res)` in a new R session.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.workflow()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(caret)

predictors <- mtcars[, c("cyl", "disp", "hp")]

set.seed(1)

mod <-
  train(
    x = predictors,
    y = mtcars$mpg,
    method = "glm"
  )

mod_bundle <- bundle(mod)

# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)

mod_unbundled_preds <- predict(mod_unbundled, new_data = mtcars)
```

`bundle.workflow` *Bundle a tidymodels* `workflow` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'workflow'
bundle(x, ...)
```

Arguments

`x`	A workflow object returned from workflows or other tidymodels packages.
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Details

This bundler wraps `bundle.model_fit()` and `bundle.recipe()`.

Value

A bundle object with subclass `bundled_workflow`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object `x` in a new environment, load its bundle with `base::readRDS()` and run `unbundle()` on it. The output of `unbundle()` is a model object that is ready to `predict()` on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See `vignette("bundle")` for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model `x`, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of `unbundle(res)` in a new R session.

See Also

Other bundlers: `bundle()`, `bundle.H2OAutoML()`, `bundle.bart()`, `bundle.keras.engine.training.Model()`, `bundle.luz_module_fitted()`, `bundle.model_fit()`, `bundle.model_stack()`, `bundle.recipe()`, `bundle.step_umap()`, `bundle.train()`, `bundle.xgb.Booster()`

Examples

```
# fit model and bundle ------------------------------------------------
library(workflows)
library(recipes)
library(parsnip)
library(xgboost)

set.seed(1)

spec <-
  boost_tree(trees = 5, mtry = 3) |>
  set_mode("regression") |>
  set_engine("xgboost")

rec <-
  recipe(mpg ~ ., data = mtcars) |>
  step_log(hp)

mod <-
  workflow() |>
  add_model(spec) |>
  add_recipe(rec) |>
```

```
  fit(data = mtcars)

mod_bundle <- bundle(mod)

# then, after saveRDS + readRDS or passing to a new session ----------
mod_unbundled <- unbundle(mod_bundle)
```

`bundle.xgb.Booster` *Bundle an* `xgb.Booster` *object*

Description

Bundling a model prepares it to be saved to a file and later restored for prediction in a new R session. See the 'Value' section for more information on bundles and their usage.

Usage

```
## S3 method for class 'xgb.Booster'
bundle(x, ...)
```

Arguments

`x`	An `xgb.Booster` object returned from `xgboost::xgboost()` or `xgboost::xgb.train()`.
`...`	Not used in this bundler and included for compatibility with the generic only. Additional arguments passed to this method will return an error.

Value

A bundle object with subclass `bundled_xgb.Booster`.

Bundles are a list subclass with two components:

`object`	An R object. Gives the output of native serialization methods from the model-supplying package, sometimes with additional classes or attributes that aid portability. This is often a raw object.
`situate`	A function. The `situate()` function is defined when `bundle()` is called, though is a loose analogue of an `unbundle()` S3 method for that object. Since the function is defined on `bundle()`, it has access to references and dependency information that can be saved alongside the `object` component. Calling `unbundle()` on a bundled object `x` calls `x$situate(x$object)`, returning the unserialized version of `object`. `situate()` will also restore needed references, such as server instances and environmental variables.

Bundles are R objects that represent a "standalone" version of their analogous model object. Thus, bundles are ready for saving to a file; saving with `base::saveRDS()` is our recommended serialization strategy for bundles, unless documented otherwise for a specific method.

To restore the original model object x in a new environment, load its bundle with base::readRDS() and run unbundle() on it. The output of unbundle() is a model object that is ready to predict() on new data, and other restored functionality (like plotting or summarizing) is supported as a side effect only.

The bundle package wraps native serialization methods from model-supplying packages. Between versions, those model-supplying packages may change their native serialization methods, possibly introducing problems with re-loading objects serialized with previous package versions. The bundle package does not provide checks for these sorts of changes, and ought to be used in conjunction with tooling for managing and monitoring model environments like vetiver or renv.

See vignette("bundle") for more information on bundling and its motivation.

bundle and butcher

The butcher package allows you to remove parts of a fitted model object that are not needed for prediction.

This bundle method is compatible with pre-butchering. That is, for a fitted model x, you can safely call:

```
res <-
  x |>
  butcher() |>
  bundle()
```

and predict with the output of unbundle(res) in a new R session.

See Also

This method adapts the xgboost functions xgboost::xgb.save.raw() and xgboost::xgb.load.raw().

Other bundlers: bundle(), bundle.H2OAutoML(), bundle.bart(), bundle.keras.engine.training.Model(), bundle.luz_module_fitted(), bundle.model_fit(), bundle.model_stack(), bundle.recipe(), bundle.step_umap(), bundle.train(), bundle.workflow()

Examples

```
# fit model and bundle ------------------------------------------------
library(xgboost)

set.seed(1)

data(agaricus.train)
data(agaricus.test)

if (utils::packageVersion("xgboost") >= "2.0.0.0") {
  xgb <- xgboost(x = agaricus.train$data, y = agaricus.train$label,
                 max_depth = 2, learning_rate = 1, nthread = 2, nrounds = 2,
                 objective = "reg:squarederror")
} else {
  xgb <- xgboost(data = agaricus.train$data, label = agaricus.train$label,
                 max_depth = 2, eta = 1, nthread = 2, nrounds = 2,
```

```
                  objective = "binary:logistic")
}

xgb_bundle <- bundle(xgb)

# then, after saveRDS + readRDS or passing to a new session ----------
xgb_unbundled <- unbundle(xgb_bundle)

xgb_unbundled_preds <- predict(xgb_unbundled, agaricus.test$data)
```

Index