Package ‘mlr3proba’

April 16, 2020

Title Probabilistic Supervised Learning for ‘mlr3’

Version 0.1.5

Description Provides extensions for probabilistic supervised
learning for ‘mlr3’. This includes extending the regression task to
probabilistic and interval regression, adding a survival task, and
other specialized models, predictions, and measures.

License MIT + file LICENSE

URL https://mlr3proba.mlr-org.com,
https://github.com/mlr-org/mlr3proba

BugReports https://github.com/mlr-org/mlr3proba/issues

Depends R (>= 3.1.0)

Imports checkmate, data.table, distr6 (>= 1.3.6), mlr3 (>= 0.1.8),
mlr3misc (>= 0.1.7), mlr3pipelines, paradox (>= 0.1.0), R6,
survival

Suggests bibtex, CoxBoost, flexsurv, gbm, ggplot2, glmnet, gss,
kerdiest, knitr, ks, logspline, locfit, Matrix, mboost,
mlr3tuning, np, obliqueRSF, pec, penalized, pendensity,
plugdensity, randomForestSRC, ranger, riskRegression, rpart,
set6, simsurv, sm, survAUC, survivalsvm, testthat, xgboost

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.1.0

RdMacros mlr3misc

Collate 'LearnerDens.R' 'LearnerDensHistogram.R' 'LearnerDensKDE.R'
'LearnerDensKDEkd.R' 'LearnerDensKDEks.R' 'LearnerDensLocfit.R'
'LearnerDensLogspline.R' 'LearnerDensMixed.R'
'LearnerDensNonparametric.R' 'LearnerDensPenalized.R'
R topics documented:

'LearnerDensPlugin.R' 'LearnerDensSpline.R' 'LearnerProbreg.R'
'LearnerProbregGaussian.R' 'LearnerSurv.R'
'LearnerSurvBlackboost.R' 'LearnerSurvCVGlmnet.R'
'LearnerSurvCoxPH.R' 'predict_flexsurvreg.R'
'LearnerSurvFlexible.R' 'LearnerSurvGBM.R'
'LearnerSurvGamboost.R' 'LearnerSurvGlmboost.R'
'LearnerSurvGlmnet.R' 'LearnerSurvKaplan.R'
'LearnerSurvMboost.R' 'LearnerSurvObliqueRSF.R'
'LearnerSurvPenalized.R' 'LearnerSurvRandomForestSRC.R'
'LearnerSurvRanger.R' 'LearnerSurvRpart.R'
'LearnerSurvXGBoost.R' 'MeasureDens.R' 'MeasureDensLogloss.R'
'MeasureRegrLogloss.R' 'MeasureSurv.R'
'MeasureSurvIntegrated.R' 'MeasureSurvAUC.R'
'MeasureSurvBeggC.R' 'MeasureSurvChamblessAUC.R'
'MeasureSurvGonenHellersC.R' 'MeasureSurvGraf.R'
'MeasureSurvGrafSE.R' 'MeasureSurvHarrellC.R'
'MeasureSurvHungAUC.R' 'MeasureSurvIntLogloss.R'
'MeasureSurvIntLoglossSE.R' 'MeasureSurvLogloss.R'
'MeasureSurvLoglossSE.R' 'MeasureSurvMAE.R'
'MeasureSurvMAESE.R' 'MeasureSurvMSE.R' 'MeasureSurvMSESE.R'
'MeasureSurvNagelkR2.R' 'MeasureSurvOQuigleyR2.R'
'MeasureSurvRMSE.R' 'MeasureSurvRMSESE.R'
'MeasureSurvSongAUC.R' 'MeasureSurvSongTNR.R'
'MeasureSurvSongTPR.R' 'MeasureSurvUnoAUC.R'
'MeasureSurvUnoC.R' 'MeasureSurvUnoTNR.R' 'MeasureSurvUnoTPR.R'
'MeasureSurvXuR2.R' 'PipeOpCrankCompositor.R'
'PipeOpDistrCompositor.R' 'PredictionDens.R'
'PredictionProbreg.R' 'PredictionSurv.R' 'TaskDens.R'
'TaskDens_faithful.R' 'TaskDens_precip.R'
'TaskGeneratorSimdens.R' 'TaskGeneratorSimsurv.R'
'TaskProbreg.R' 'TaskSurv.R' 'TaskSurv_lung.R'
'TaskSurv_rats.R' 'TaskSurv_unemployment.R' 'assertions.R'
'autoplot.R' 'crankcompositor.R' 'distrcompositor.R'
'helpers.R' 'histogram.R' 'integrated_scores.R' 'pecs.R'
'surv_measures.R' 'weighted_survival_score.R' 'zzz.R'

Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>),
Franz Kiraly [aut],
Michel Lang [aut] (<https://orcid.org/0000-0001-9754-0393>),
Nurul Ain Toha [ctb],
Andreas Bender [ctb] (<https://orcid.org/0000-0001-5628-8611>)

Maintainer Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk>

Repository CRAN

Date/Publication 2020-04-16 16:10:02 UTC

R topics documented:
R topics documented:

- crankcompositor ........................................ 5
- distrcompositor ........................................... 6
- LearnerDens .............................................. 7
- LearnerDensHistogram .................................. 9
- LearnerDensKDE .......................................... 10
- LearnerDensKDEkd ......................................... 11
- LearnerDensKDEks .......................................... 12
- LearnerDensLocfit ......................................... 14
- LearnerDensLogspline .................................... 15
- LearnerDensMixed ......................................... 16
- LearnerDensNonparametric ................................ 17
- LearnerDensPenalized ..................................... 18
- LearnerDensPlugin ......................................... 19
- LearnerDensSpline ......................................... 20
- LearnerSurv ............................................... 21
- LearnerSurvBlackboost ................................... 23
- LearnerSurvCoxPH ......................................... 25
- LearnerSurvCVGlmnet ..................................... 27
- LearnerSurvFlexible ....................................... 28
- LearnerSurvGamboost ..................................... 30
- LearnerSurvGBM ........................................... 32
- LearnerSurvGlmboost ..................................... 34
- LearnerSurvGlmnet ......................................... 36
- LearnerSurvKaplan ......................................... 37
- LearnerSurvMboost ......................................... 38
- LearnerSurvObliqueRSF .................................... 40
- LearnerSurvPenalized ..................................... 42
- LearnerSurvRandomForestSRC ............................. 44
- LearnerSurvRanger ......................................... 46
- LearnerSurvRpart ........................................... 47
- LearnerSurvXgboost ....................................... 49
- MeasureDens ................................................ 50
- MeasureDensLogloss ....................................... 52
- MeasureSurv ............................................... 53
- MeasureSurvAUC ........................................... 54
- MeasureSurvBeggC ......................................... 55
- MeasureSurvChamblessAUC ................................ 57
- MeasureSurvGonenC ........................................ 58
- MeasureSurvGraf ........................................... 60
- MeasureSurvGrafSE ........................................ 62
- MeasureSurvHarrellIC ..................................... 64
- MeasureSurvHungAUC ...................................... 65
- MeasureSurvIntegrated .................................... 67
- MeasureSurvIntLogloss .................................... 68
- MeasureSurvIntLoglossSE .................................. 71
- MeasureSurvLogloss ....................................... 73
- MeasureSurvLoglossSE ..................................... 74
- MeasureSurvMAE ............................................ 76
mlr3proba-package

mlr3proba: Probabilistic Supervised Learning for 'mlr3'

Description

Provides extensions for probabilistic supervised learning for 'mlr3'. This includes extending the regression task to probabilistic and interval regression, adding a survival task, and other specialized models, predictions, and measures.
crankcompositor

Author(s)

Maintainer: Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk> (ORCID)
Authors:

• Franz Kiraly <f.kiraly@ucl.ac.uk>
• Michel Lang <michellang@gmail.com> (ORCID)

Other contributors:

• Nurul Ain Toha <nurul.toha.15@ucl.ac.uk> [contributor]
• Andreas Bender <bender.at.R@gmail.com> (ORCID) [contributor]

See Also

Useful links:

• https://mlr3proba.mlr-org.com
• https://github.com/mlr-org/mlr3proba
• Report bugs at https://github.com/mlr-org/mlr3proba/issues

Description

This is a wrapper around the PipeOpCrankCompositor pipe operation, which simplifies graph creation.

Usage

```
crankcompositor(
  learner,
  method = c("mean", "median", "mode"),
  param_vals = list()
)
```

Arguments

- **learner**: LearnerSurv object for which a crank is composed (or over-written)
- **method**: One of mean, mode, or median; abbreviations allowed. Used to determine how crank is estimated from the predicted distr. Default is mean.
- **param_vals**: Additional parameters to pass to the learner.

Details

For full details see PipeOpCrankCompositor.
distrcompositor

Value

mlr3pipelines::GraphLearner

Examples

```r
## Not run:
library(mlr3)
library(mlr3pipelines)

 task = tgen("simsurv")$generate(20)
ranger.crank = crankcompositor(learner = lrn("surv.coxph"),
                               method = "median")
resample(task, ranger.crank, rsmp("cv", folds = 2))$predictions()
## End(Not run)
```

distrcompositor  Compose a Distr Predict Type for Survival Learners

Description

This is a wrapper around the PipeOpDistrCompositor pipe operation, which simplifies graph creation.

Usage

```r
distrcompositor(
  learner,
  estimator = c("kaplan", "nelson"),
  form = c("aft", "ph", "po"),
  overwrite = FALSE,
  param_vals = list()
)
```

Arguments

- **learner**: LearnerSurv object for which a distr is composed (or over-written).
- **estimator**: One of kaplan or nelson, corresponding to the Kaplan-Meier and Nelson-Aalen estimators respectively. Used to estimate the baseline survival distribution. Abbreviations allowed. Default is kaplan.
- **form**: One of aft, ph, or po, corresponding to accelerated failure time, proportional hazards, and proportional odds respectively. Used to determine the form of the composed survival distribution. Default is aft.
- **overwrite**: logical. If FALSE (default) then if the learner already has a distr, the compositor does nothing. If TRUE then the distr is overwritten by the compositor if already present, which may be required for changing the prediction distr from one model form to another.
- **param_vals**: Additional parameters to pass to the learner.
LearnerDens

Details
For full details see PipeOpDistrCompositor.

Value
mlr3pipelines::GraphLearner

Examples
## Not run:
library("mlr3")
library("mlr3pipelines")

task = tgen("simsurv")$generate(20)
cvglm.distr = distrcompositor(learner = lrn("surv.cvglmnet"),
estimator = "kaplan",
 form = "aft")

resample(task, cvglm.distr, rsmp("cv", folds = 2))$predictions()
## End(Not run)

LearnerDens Density Learner

Description
This Learner specializes Learner for density estimation problems:

- task_type is set to "dens"
- Creates Predictions of class PredictionDens.
- Possible values for predict_types are:
  - "pdf": Evaluates the estimated probability density function for each value in the test set.
  - "se": Evaluates the estimated cumulative distribution function for each value in the test set.

Super class
mlr3::Learner -> LearnerDens

Methods
Public methods:
- LearnerDens$new()
- LearnerDens$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
LearnerDens$new(
  id,
  param_set = ParamSet$new(),
  predict_types = "cdf",
  feature_types = character(),
  properties = character(),
  data_formats = "data.table",
  packages = character()
)

Arguments:
id (character(1))
  Identifier for the new instance.
param_set (paradox::ParamSet)
  Set of hyperparameters.
predict_types (character())
  Supported predict types. Must be a subset of mlr_reflections$learner_predict_types.
feature_types (character())
  Feature types the learner operates on. Must be a subset of mlr_reflections$task_feature_types.
properties (character())
  Set of properties of the Learner. Must be a subset of mlr_reflections$learner_properties.

  The following properties are currently standardized and understood by learners in mlr3:
  • "missings": The learner can handle missing values in the data.
  • "weights": The learner supports observation weights.
  • "importance": The learner supports extraction of importance scores, i.e. comes with an $importance() extractor function (see section on optional extractors in Learner).
  • "selected_features": The learner supports extraction of the set of selected features, i.e. comes with a $selected_features() extractor function (see section on optional extractors in Learner).
  • "oob_error": The learner supports extraction of estimated out of bag error, i.e. comes with a oob_error() extractor function (see section on optional extractors in Learner).

data_formats (character())
  Set of supported data formats which can be processed during $train() and $predict(), e.g. "data.table".
packages (character())
  Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDens$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

See Also

Other Learner: LearnerSurv
library(mlr3)
# get all density learners from mlr_learners:
lrns = mlr_learners$mget(mlr_learners$keys("dens"))
names(lrns)

# get a specific learner from mlr_learners:
mlr_learners$get("dens.hist")
lrn("dens.hist")

---

**LearnerDensHistogram**  
*Histogram Density Estimator*

**Description**

Calls `graphics::hist()` and the result is coerced to a `distr6::Distribution`.

**Dictionary**

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
defineLearnerDensHistogram()
mlr_learners$get("dens.hist")
lrn("dens.hist")
```

**Meta Information**

- Type: "dens"
- Predict Types: pdf, cdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: `distr6`

**Super classes**

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensHistogram`

**Methods**

**Public methods:**

- `LearnerDensHistogram$new()`
- `LearnerDensHistogram$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

**Usage:**
LearnerDensKDE

LearnerDensHistogram$new()

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*
LearnerDensHistogram$clone(deep = FALSE)

*Arguments:*
deep Whether to make a deep clone.

**See Also**
Other density estimators: LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

---

**LearnerDensKDE**  
**Kernel Density Estimator**

**Description**
Calls kernels implemented in distr6 and the result is coerced to a distr6::Distribution.

**Details**
The default bandwidth uses Silverman’s rule-of-thumb for Gaussian kernels, however for non-Gaussian kernels it is recommended to use mlr3tuning to tune the bandwidth with cross-validation. Other density learners can be used for automated bandwidth selection.

**Dictionary**
This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerDensKDE$new()
mlr_learners$get("dens.kde")
lrn("dens.kde")

**Meta Information**
- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: missings
- Packages: distr6

**Super classes**
mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensKDE
Methods

Public methods:

• LearnerDensKDE$new()
• LearnerDensKDE$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensKDE$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDensKDE$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

References


See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

LearnerDensKDEkd  Kerdiest Kernel Density Estimator

Description

Calls kerdiest::kde() and the result is coerced to a distr6::Distribution.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerDensKDEkd$new()
mlr_learners$get("dens.kdeKD")
lrn("dens.kdeKD")
Meta Information

- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: kerdiest distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensKDEkd

Methods

Public methods:

- LearnerDensKDEkd$new()
- LearnerDensKDEkd$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensKDEkd$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDensKDEkd$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

LearnerDensKDEks | KS Kernel Density Estimator

Description

Calls ks::kde() and the result is coerced to a distr6::Distribution.
LearnerDensKDEeks

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerDensKDEeks$new()
mlr_learners$get("dens.kdeKS")
lrn("dens.kdeKS")

Meta Information

- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: ks distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensKDEeks

Methods

Public methods:

- LearnerDensKDEeks$new()
- LearnerDensKDEeks$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensKDEeks$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDensKDEeks$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline
Description

Calls `locfit::density.lf()` and the result is coerced to a `distr6::Distribution`.

Dictionary

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```
LearnerDensLocfit$new()
mlr_learners$get("dens.locfit")
lrn("dens.locfit")
```

Meta Information

- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: locfit distr6

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> LearnerDensLocfit

Methods

Public methods:

- `LearnerDensLocfit$new()`
- `LearnerDensLocfit$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```
LearnerDensLocfit$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
LearnerDensLocfit$clone(deep = FALSE)
```

Arguments:

- `deep`: Whether to make a deep clone.
**LearnerDensLogspline**

**Description**

Calls `logspline::logspline()` and the result is coerced to a `distr6::Distribution`.

**Dictionary**

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
LearnerDensLogspline$new()
mlr_learners$get("dens.logspline")
lrn("dens.logspline")
```

**Meta Information**

- Type: "dens"
- Predict Types: pdf, cdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties:
- Packages: `logspline distr6`

**Super classes**

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> LearnerDensLogspline

**Methods**

**Public methods:**

- `LearnerDensLogspline$new()`
- `LearnerDensLogspline$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

**Usage:**

```r
LearnerDensLogspline$new()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**

```r
LearnerDensLogspline$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

**See Also**

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline
LearnerDensMixed

See Also
Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

LearnerDensMixed Mixed Data Kernel Density Estimator

Description
Calls \( \text{np::npudens()} \) and the result is coerced to a \texttt{distr6::Distribution}.

Dictionary
This Learner can be instantiated via the dictionary \texttt{mlr_learners} or with the associated sugar function \texttt{lrn()}:

LearnerDensMixed$new()
mlr_learners$get("dens.mixed")
lrn("dens.mixed")

Meta Information
- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: \texttt{np distr6}

Super classes
\texttt{mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensMixed}

Methods

Public methods:
- \texttt{LearnerDensMixed$new()}
- \texttt{LearnerDensMixed$clone()}

Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

Usage:
\texttt{LearnerDensMixed$new()}

Method \texttt{clone()}: The objects of this class are cloneable with this method.

Usage:
\texttt{LearnerDensMixed$clone(deep = FALSE)}

Arguments:
deepl Whether to make a deep clone.
LearnerDensNonparametric

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

LearnerDensNonparametric

Nonparametric Density Estimator

Description

Calls sm::sm.density() and the result is coerced to a distr6::Distribution.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerDensNonparametric$new()
mlr_learners$get("dens.nonpar")
lrn("dens.nonpar")

Meta Information

• Type: "dens"
• Predict Types: pdf
• Feature Types: logical, integer, numeric, character, factor, ordered
• Properties: weights
• Packages: sm distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensNonparametric

Methods

Public methods:
• LearnerDensNonparametric$new()
• LearnerDensNonparametric$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensNonparametric$new()

Method clone(): The objects of this class are cloneable with this method.
Usage:
LearnerDensNonparametric$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensPenalized, LearnerDensPlugin, LearnerDensSpline

---

LearnerDensPenalized  Penalized Density Estimator

Description

Calls `pendensity::pendensity()` and the result is coerced to a `distr6::Distribution`.

Dictionary

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

LearnerDensPenalized$new()
mlr_learners$get("dens.pen")
lrn("dens.pen")

Meta Information

- Type: "dens"
- Predict Types: pdf, cdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: -
- Packages: `pendensity distr6`

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> LearnerDensPenalized
LearnerDensPlugin

Methods

Public methods:

- LearnerDensPenalized$new()
- LearnerDensPenalized$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensPenalized$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDensPenalized$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspline, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPlugin, LearnerDensSpline

LearnerDensPlugin  Plug-In Kernel Density Estimator

Description

Calls plugdensity::plugin.density() and the result is coerced to a distr6::Distribution.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerDensPlugin$new()
mlr_learners$get("dens.plug")
lrn("dens.plug")

Meta Information

- Type: "dens"
- Predict Types: pdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: missings
- Packages: plugdensity distr6
Super classes

\[
\text{mlr3::Learner} \rightarrow \text{mlr3proba::Learner} \rightarrow \text{LearnerDensPlugin}
\]

Methods

Public methods:

- \text{LearnerDensPlugin}\$\text{new()}
- \text{LearnerDensPlugin}\$\text{clone()}

Method \text{new()}: Creates a new instance of this \text{R6} class.

Usage:

\text{LearnerDensPlugin}\$\text{new()}

Method \text{clone()}: The objects of this class are cloneable with this method.

Usage:

\text{LearnerDensPlugin}\$\text{clone(\text{deep} = \text{FALSE})}

Arguments:

- \text{deep} Whether to make a deep clone.

See Also

Other density estimators: \text{LearnerDensHistogram}, \text{LearnerDensKDEkd}, \text{LearnerDensKDEks}, \text{LearnerDensKDE}, \text{LearnerDensLocfit}, \text{LearnerDensLogspline}, \text{LearnerDensMixed}, \text{LearnerDensNonparametric}, \text{LearnerDensPenalized}, \text{LearnerDensSpline}

---

LearnerDensSpline  \textit{Smoothing Splines Density Estimator}

Description

Calls \text{gss::ssden()} and the result is coerced to a \text{distr6::Distribution}.

Dictionary

This \text{Learner} can be instantiated via the \text{dictionary mlr_learners} or with the associated sugar function \text{lrn()}:

\text{LearnerDensSpline}\$\text{new()}

\text{mlr_learners}$\text{get(“dens.spline”)}

\text{lrn(“dens.spline”)}
Meta Information

- Type: "dens"
- Predict Types: pdf, cdf
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: missings
- Packages: gss distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerDens -> LearnerDensSpline

Methods

Public methods:

- LearnerDensSpline$new()
- LearnerDensSpline$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerDensSpline$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerDensSpline$clone(deep = FALSE)

Arguments:
deepl Whether to make a deep clone.

See Also

Other density estimators: LearnerDensHistogram, LearnerDensKDEkd, LearnerDensKDEks, LearnerDensKDE, LearnerDensLocfit, LearnerDensLogspine, LearnerDensMixed, LearnerDensNonparametric, LearnerDensPenalized, LearnerDensPlugin
Description

This Learner specializes Learner for survival problems:

- task_type is set to "surv"
- Creates Predictions of class PredictionSurv.
- Possible values for predict_types are:
  - "distr": Predicts a probability distribution for each observation in the test set, uses distr6.
  - "lp": Predicts a linear predictor for each observation in the test set.
  - "crank": Predicts a continuous ranking for each observation in the test set.
  - "response": Predicts a survival time for each observation in the test set.

Super class

mlr3::Learner -> LearnerSurv

Methods

Public methods:

- LearnerSurv$new()
- LearnerSurv$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurv$new(
  id,
  param_set = ParamSet$new(),
  predict_types = "distr",
  feature_types = character(),
  properties = character(),
  packages = character(),
  man = NA_character_
)

Arguments:

id (character(1))
  Identifier for the new instance.
param_set (paradox::ParamSet)
  Set of hyperparameters.
predict_types (character())
  Supported predict types. Must be a subset of mlr_reflections$learner_predict_types.
feature_types (character())
  Feature types the learner operates on. Must be a subset of mlr_reflections$task_feature_types.
properties (character())
  Set of properties of the Learner. Must be a subset of mlr_reflections$learner_properties.
The following properties are currently standardized and understood by learners in mlr3:
• "missings": The learner can handle missing values in the data.
• "weights": The learner supports observation weights.
• "importance": The learner supports extraction of importance scores, i.e. comes with an $importance() extractor function (see section on optional extractors in Learner).
• "selected_features": The learner supports extraction of the set of selected features, i.e. comes with a $selected_features() extractor function (see section on optional extractors in Learner).
• "oob_error": The learner supports extraction of estimated out of bag error, i.e. comes with a oob_error() extractor function (see section on optional extractors in Learner).

packages (character())
Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().

man (character(1))
String in the format [pkg]:[topic] pointing to a manual page for this object. The referenced help package can be opened via method $help().

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurv$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also
Other Learner: LearnerDens

Examples

library(mlr3)
# get all survival learners from mlr_learners:
lrns = mlr_learners$mget(mlr_learners$keys("surv"))
names(lrns)

# get a specific learner from mlr_learners:
mlr_learners$get("surv.coxph")
lnr("surv.coxph")

LearnerSurvBlackboost Gradient Boosting with Regression Trees Survival Learner

Description

Calls mboost::blackboost().
• lp is predicted by mboost::predict.mboost()
• distr is predicted by mboost::survFit() which assumes a PH fit with a Breslow estimator
crank is identical to lp

The dist parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default "coxph".

If the value given to the Family parameter is "custom.family" then an object of class mboost::Family() needs to be passed to the custom.family parameter.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerSurvBlackboost$new()
mlr_learners$get("surv.blackboost")
lrn("surv.blackboost")

Meta Information

• Type: "surv"
• Predict Types: distr, crank, lp
• Feature Types: integer, numeric, factor
• Properties: -
• Packages: mboost distr6 survival partykit mvtnorm

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvBlackboost

Methods

Public methods:
• LearnerSurvBlackboost$new()
• LearnerSurvBlackboost$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvBlackboost$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvBlackboost$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
LearnerSurvCoxPH

References


See Also

Other survival learners: LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

Examples

```r
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.blackboost")
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)
```

**LearnerSurvCoxPH**

Cox Proportional Hazards Survival Learner

Description

Calls `survival::coxph()`.

- lp is predicted by `survival::predict.coxph()`
- distr is predicted by `survival::survfit.coxph()`
- crank is identical to lp

Dictionary

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
LearnerSurvCoxPH$new()
mlr_learners$get("surv.coxph")
lrn("surv.coxph")
```
Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: logical, integer, numeric, factor
- Properties: weights
- Packages: survival distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvCoxPH

Methods

Public methods:

- LearnerSurvCoxPH$new()
- LearnerSurvCoxPH$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvCoxPH$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvCoxPH$clone(deep = FALSE)

Arguments:
- deep Whether to make a deep clone.

References


See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost
**LearnerSurvCVGlmnet**

Cross-Validated GLM with Elastic Net Regularization Survival Learner

**Description**

Calls `glmnet::cv.glmnet()`.

- lp is predicted by `glmnet::predict.cv.glmnet()`
- crank is identical to lp

Use LearnerSurvGlmnet and LearnerSurvCVGlmnet for glmnets without and with internal cross-validation, respectively. Tuning using the internal optimizer in LearnerSurvCVGlmnet may be more efficient when tuning lambda only. However, for tuning multiple hyperparameters, mlr3tuning and LearnerSurvGlmnet will likely give better results.

**Dictionary**

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

```r
LearnerSurvCVGlmnet$new()
mlr_learners$get("surv.cvglmnet")
lrn("surv.cvglmnet")
```

**Meta Information**

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor
- Properties: weights
- Packages: glmnet survival

**Super classes**

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvCVGlmnet

**Methods**

**Public methods:**

- LearnerSurvCVGlmnet$new()
- LearnerSurvCVGlmnet$clone()

**Method** new(): Creates a new instance of this R6 class.

**Usage:**

LearnerSurvCVGlmnet$new()
**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**

```r
LearnerSurvCVGlmnet$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

**References**


**See Also**

Other survival learners: `LearnerSurvBlackboost`, `LearnerSurvCoxPH`, `LearnerSurvFlexible`, `LearnerSurvGBM`, `LearnerSurvGamboost`, `LearnerSurvGlmboost`, `LearnerSurvGlmnet`, `LearnerSurvKaplan`, `LearnerSurvMboost`, `LearnerSurvObliqueRSF`, `LearnerSurvPenalized`, `LearnerSurvRandomForestSRC`, `LearnerSurvRanger`, `LearnerSurvRpart`, `LearnerSurvXgboost`

---

**LearnerSurvFlexible**  
*Flexible Parametric Spline Survival Learner*

**Description**

Calls `flexsurv::flexsurvspline()`.

- `lp` is predicted by using an internally defined `predict` method, see details
- `distr` is predicted by using an internally defined `predict` method, see details
- `crank` is identical to `lp`

Parameter `k` is changed to 1 and `scale` is changed to `odds`, as these are more in line with the Royston/Parmar proposed models, and the package defaults are equivalent to fitting a parametric model and therefore `surv.parametric` should be used instead.

If fitting a model with `k = 0` then consider using `surv.parametric` as this is likely to have more optimal results, and has more options for tuning.

**Details**

The `distr` prediction is estimated using the fitted custom distributions from `flexsurv::flexsurvspline()` and the estimated coefficients.

As flexible spline models estimate the baseline hazard as the intercept, the linear predictor, `lp`, can be calculated as in the classical setting. i.e. For fitted coefficients, $\beta = (\beta_0, ..., \beta_P)$, and covariates $X^T = (X_0, ..., X_P)^T$, where $X_0$ is a column of 1s: $lp = \beta X$. 
This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function \texttt{lrn()}:

\begin{verbatim}
LearnerSurvFlexible$new()
mlr_learners$get("surv.flexible")
lrn("surv.flexible")
\end{verbatim}

**Meta Information**

- **Type**: "surv"
- Predict Types: distr, lp, crank
- Feature Types: logical, integer, factor, numeric
- Properties: weights
- Packages: \texttt{flexsurv survival distr6 set6}

**Super classes**

\begin{verbatim}
mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvFlexible
\end{verbatim}

**Methods**

**Public methods:**

- \texttt{LearnerSurvFlexible$\texttt{new}()}
- \texttt{LearnerSurvFlexible$\texttt{clone}()}

**Method \texttt{new}():** Creates a new instance of this \texttt{R6} class.

**Usage:**

\begin{verbatim}
LearnerSurvFlexible$new()
\end{verbatim}

**Method \texttt{clone}():** The objects of this class are cloneable with this method.

**Usage:**

\begin{verbatim}
LearnerSurvFlexible$\texttt{clone}(deep = FALSE)
\end{verbatim}

**Arguments:**

depth Whether to make a deep clone.

**References**

Gradient Boosting for Additive Models Survival Learner

Description

Calls `mboost::gamboost()`. 

- lp is predicted by `mboost::predict.mboost()`
- distr is predicted by `mboost::survFit()` which assumes a PH fit with a Breslow estimator
- crank is identical to lp

The dist parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default "coxph". If the value given to the Family parameter is "custom.family" then an object of class `mboost::Family()` needs to be passed to the custom.family parameter.

The only difference between LearnerSurvGamboost and LearnerSurvMboost is that the latter function allows one to specify default degrees of freedom for smooth effects specified via baselearner = "bbs". In all other cases, degrees of freedom need to be set manually via a specific definition of the corresponding base-learner.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```r
LearnerSurvGamboost$new()
mlr_learners$get("surv.gamboost")
lrn("surv.gamboost")
```

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Properties: -
- Packages: mboost distr6 survival
Super classes

\texttt{mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvGamboost}

Methods

**Public methods:**

- \texttt{LearnerSurvGamboost$new()}
- \texttt{LearnerSurvGamboost$clone()}

**Method** \texttt{new():} Creates a new instance of this \texttt{R6} class.

*Usage:*

\texttt{LearnerSurvGamboost$new()}

**Method** \texttt{clone():} The objects of this class are cloneable with this method.

*Usage:*

\texttt{LearnerSurvGamboost$clone(deep = FALSE)}

*Arguments:*

- \texttt{deep} Whether to make a deep clone.

References


See Also

Other survival learners: \texttt{LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost}
Examples

```r
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.gamboost")
learner$param_set$values = mlr3misc::insert_named(learner$param_set$values,
    list(dfbase = 3, center = TRUE, baselearner = "bols"))
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)
```

---

**LearnerSurvGBM**

*Generalized Boosting Regression Modeling Survival Learner*

Description

Calls `gbm::gbm()`.

- lp is predicted by `gbm::predict.gbm()`
- crank is identical to lp

Parameter distribution is set to coxph as this is the only distribution implemented in `gbm::gbm()` for survival analysis; parameter keep.data is set to FALSE for efficiency.

Dictionary

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
LearnerSurvGBM$new()
mlr_learners$get("surv.gbm")
lrn("surv.gbm")
```

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor, ordered
- Properties: importance, missings, weights
- Packages: **gbm**

Super classes

```
mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvGBM
```
Methods

Public methods:

- LearnerSurvGBM$new()
- LearnerSurvGBM$importance()
- LearnerSurvGBM$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
LearnerSurvGBM$new()

Method importance(): The importance scores are extracted from the model slot variable.importance.
Usage:
LearnerSurvGBM$importance()
Returns: Named numeric().

Method clone(): The objects of this class are cloneable with this method.
Usage:
LearnerSurvGBM$clone(deep = FALSE)
Arguments:
deeplWhether to make a deep clone.

References


See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

Description

Calls mboost::glmboost().

- `lp` is predicted by mboost::predict.mboost()
- `distr` is predicted by mboost::survFit() which assumes a PH fit with a Breslow estimator
- `crank` is identical to `lp`

The `dist` parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default "coxph".

If the value given to the `Family` parameter is "custom.family" then an object of class mboost::Family() needs to be passed to the custom.family parameter.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerSurvGlmboost$new()
mlr_learners$get("surv.glmboost")
lrn("surv.glmboost")

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Properties: -
- Packages: mboost distr6 survival

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvGlmboost
LearnerSurvGlmboost

Methods

Public methods:

- LearnerSurvGlmboost$new()
- LearnerSurvGlmboost$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvGlmboost$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvGlmboost$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References


See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

Examples

```r
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.glmboost")
resampling = rsmp("cv", folds = 3)
resample(task, learner, resampling)
```
LearnerSurvGlmnet

GLM with Elastic Net Regularization Survival Learner

Description

Calls \texttt{glmnet::glmnet()}.

- \texttt{lp} is predicted by \texttt{glmnet::predict.glmnet()}
- \texttt{crank} is identical to \texttt{lp}

Use \texttt{LearnerSurvGlmnet} and \texttt{LearnerSurvCVGlmnet} for glmnets without and with internal cross-validation, respectively. Tuning using the internal optimizer in \texttt{LearnerSurvCVGlmnet} may be more efficient when tuning lambda only. However, for tuning multiple hyperparameters, \texttt{mlr3tuning} and \texttt{LearnerSurvGlmnet} will likely give better results.

Parameter \( s \) (value of the regularization parameter used for predictions) is set to the median of the lambda sequence by default, but needs to be tuned by the user.

Dictionary

This Learner can be instantiated via the dictionary \texttt{mlr_learners} or with the associated sugar function \texttt{lrn()}:

\begin{verbatim}
LearnerSurvGlmnet$new()
mlr_learners$get("surv.glmnet")
lrn("surv.glmnet")
\end{verbatim}

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: integer, numeric, factor
- Properties: weights
- Packages: \texttt{glmnet survival}

Super classes

\texttt{mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvGlmnet}

Methods

Public methods:

- \texttt{LearnerSurvGlmnet$new()}
- \texttt{LearnerSurvGlmnet$clone()}

Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

Usage:
LearnerSurvGlmnet$new()

**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**
LearnerSurvGlmnet$clone(deep = FALSE)

**Arguments:**
depth Whether to make a deep clone.

**References**

**See Also**
Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

---

**LearnerSurvKaplan**  
*Kaplan-Meier Estimator Survival Learner*

**Description**
Calls `survival::survfit()`.

- distr is predicted by estimating the survival function with `survival::survfit()`
- crank is predicted as the expectation of the survival distribution, distr

**Dictionary**
This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

LearnerSurvKaplan$new()
mlr_learners$get("surv.kaplan")
lrn("surv.kaplan")

**Meta Information**
- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: missings
- Packages: *survival, distr6*
Super classes

\texttt{mlr3::Learner} -> \texttt{mlr3proba::LearnerSurv} -> \texttt{LearnerSurvKaplan}

Methods

Public methods:

- \texttt{LearnerSurvKaplan\$new()}
- \texttt{LearnerSurvKaplan\$clone()}

Method \texttt{new()}: Creates a new instance of this R6 class.

\textit{Usage}:
\texttt{LearnerSurvKaplan\$new()}

Method \texttt{clone()}: The objects of this class are cloneable with this method.

\textit{Usage}:
\texttt{LearnerSurvKaplan\$clone(deep = FALSE)}

\textit{Arguments}:

- \texttt{deep} Whether to make a deep clone.

References


See Also

Other survival learners: \texttt{LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost}

\begin{verbatim}
LearnerSurvMboost
\end{verbatim}

\textit{Gradient Boosting for Generalized Additive Models Survival Learner}

Description

Calls \texttt{mboost::mboost()}.

- \texttt{lp} is predicted by \texttt{mboost::predict.mboost()}
- \texttt{distr} is predicted by \texttt{mboost::survFit()} which assumes a PH fit with a Breslow estimator
- \texttt{crank} is identical to \texttt{lp}
The dist parameter is specified slightly differently than in mboost. Whereas the latter takes in objects, in this learner instead a string is specified in order to identify which distribution to use. As the default in mboost is the Gaussian family, which is not compatible with survival models, instead we have by default "coxph".

If the value given to the Family parameter is "custom.family" then an object of class mboost::Family() needs to be passed to the custom.family parameter.

The only difference between LearnerSurvGamboost and LearnerSurvMboost is that the latter function allows one to specify default degrees of freedom for smooth effects specified via baselearner = "bbs". In all other cases, degrees of freedom need to be set manually via a specific definition of the corresponding base-learner.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerSurvMboost$new()
mlr_learners$get("surv.mboost")
lrn("surv.mboost")

Meta Information

- Type: "surv"
- Predict Types: distr, crank, lp
- Feature Types: integer, numeric, factor, logical
- Properties: -
- Packages: mboost distr6 survival

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvMboost

Methods

Public methods:

- LearnerSurvMboost$new()
- LearnerSurvMboost$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvMboost$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvMboost$clone(deep = FALSE)

Arguments:

depth Whether to make a deep clone.
References


See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

Examples

library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.mboost")
learner$param_set$values = mlr3misc::insert_named(learner$param_set$values,
  list(center = TRUE, baselearner = "bols"))
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)

LearnerSurvObliqueRSF  Oblique Random Survival Forest Survival Learner

Description

Calls obliqueRSF::ORSF().

- distr is predicted by obliqueRSF::predict.orsf()
- crank is predicted as the expectation of the survival distribution, distr

Parameter verbose is set to FALSE.
Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

```r
LearnerSurvObliqueRSF$new()
mlr_learners$get("surv.obliqueRSF")
lrn("surv.obliqueRSF")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: integer, numeric, factor, ordered
- Properties: missings
- Packages: obliqueRSF distr6

Super classes

```r
mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvObliqueRSF
```

Methods

Public methods:

- `LearnerSurvObliqueRSF$new()`
- `LearnerSurvObliqueRSF$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
LearnerSurvObliqueRSF$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
LearnerSurvObliqueRSF$clone(deep = FALSE)
```

Arguments:

depth Whether to make a deep clone.

References

See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

LearnerSurvPenalized  L1 and L2 Penalized Estimation in GLMs Survival Learner

Description

Calls penalized::penalized().

- distr is predicted using penalized::predict()
- crank is predicted as the expectation of the survival distribution, distr

The penalized and unpenalized arguments in the learner are implemented slightly differently than in penalized::penalized(). Here, there is no parameter for penalized but instead it is assumed that every variable is penalized unless stated in the unpenalized parameter, see examples.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerSurvPenalized$new()
mlr_learners$get("surv.penalized")
lrn("surv.penalized")

Meta Information

- Type: "surv"
- Predict Types: distr, crank
- Feature Types: integer, numeric, factor, ordered
- Properties: -
- Packages: penalized distr6

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvPenalized
Methods

Public methods:

- LearnerSurvPenalized$new()
- LearnerSurvPenalized$clone()

Method `new()`: Creates a new instance of this R6 class.

Usage:
LearnerSurvPenalized$new()

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
LearnerSurvPenalized$clone(deep = FALSE)

Arguments:
deeplWhether to make a deep clone.

References


See Also

Dictionary of Learners: mlr3::mlr_learners

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost

Examples

```r
library(mlr3)
task = tgen("simsurv")$generate(20)
learner = lrn("surv.penalized")
resampling = rsmp("cv", folds = 2)
resample(task, learner, resampling)

# specifying penalized and unpenalized variables
task = tgen("simsurv")$generate(20)
learner = lrn("surv.penalized", unpenalized = c("height"))
learner$train(task)
learner$model@penalized
learner$model@unpenalized
```
Description

Calls `randomForestSRC::rfsrc()`.

- `distr` is predicted using `randomForestSRC::predict.rfsrcc()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

`randomForestSRC::predict.rfsrcc()` returns both cumulative hazard function (chf) and survival function (surv) but uses different estimators to derive these. chf uses a bootstrapped Nelson-Aalen estimator, (Ishwaran, 2008) whereas surv uses a bootstrapped Kaplan-Meier estimator [https://kogalur.github.io/randomForestSRC/theory.html](https://kogalur.github.io/randomForestSRC/theory.html). The choice of which estimator to use is given by the extra estimator hyper-parameter, default is `nelson`.

Dictionary

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
LearnerSurvRandomForestSRC$new()
mlr_learners$get("surv.randomForestSRC")
lrn("surv.randomForestSRC")
```

Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, factor, ordered
- Properties: importance, missings, weights
- Packages: `randomForestSRC distr6`

Super classes

`mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvRandomForestSRC`

Methods

Public methods:

- `LearnerSurvRandomForestSRC$new()`
- `LearnerSurvRandomForestSRC$importance()`
- `LearnerSurvRandomForestSRC$selected_features()`
- `LearnerSurvRandomForestSRC$clone()`
**Method** `new()`: Creates a new instance of this R6 class.

*Usage:*

```r
LearnerSurvRandomForestSRC$new()
```

**Method** `importance()`: The importance scores are extracted from the model slot `variable.importance`.

*Usage:*

```r
LearnerSurvRandomForestSRC$importance()
```

*Returns*: Named numeric().

**Method** `selected_features()`: Selected features are extracted from the model slot `frame$var`.

*Usage:*

```r
LearnerSurvRandomForestSRC$selected_features()
```

*Returns*: character().

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```r
LearnerSurvRandomForestSRC$clone(deep = FALSE)
```

*Arguments:*

depth Whether to make a deep clone.

---

**References**


**See Also**

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRanger, LearnerSurvRpart, LearnerSurvXgboost
LearnerSurvRanger  
*Ranger Survival Forest Survival Learner*

**Description**

Calls `ranger::ranger()`.

- `distr` is predicted using `ranger::predict.ranger()`
- `crank` is predicted as the expectation of the survival distribution, `distr`

**Dictionary**

This Learner can be instantiated via the dictionary `mlr_learners` or with the associated sugar function `lrn()`:

```r
LearnerSurvRanger$new()
mlr_learners$get("surv.ranger")
lrn("surv.ranger")
```

**Meta Information**

- Type: "surv"
- Predict Types: distr, crank
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: importance, oob_error, weights
- Packages: ranger distr6

**Super classes**

```
mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvRanger
```

**Methods**

**Public methods:**

- `LearnerSurvRanger$new()`
- `LearnerSurvRanger$importance()`
- `LearnerSurvRanger$ooe_error()`
- `LearnerSurvRanger$clone()`

**Method `new()`**: Creates a new instance of this R6 class.

*Usage:*

```r
LearnerSurvRanger$new()
```

**Method `importance()`**: The importance scores are extracted from the model slot `variable.importance`.

*Usage:*

```r
```
LearnerSurvRpart

LearnerSurvRanger$importance()
_Returns:_ Named numeric().

**Method** oob_error(): The out-of-bag error is extracted from the model slot prediction.error.

_Usage:_
LearnerSurvRanger$oob_error()
_Returns:_ numeric(1).

**Method** clone(): The objects of this class are cloneable with this method.

_Usage:_
LearnerSurvRanger$clone(deep = FALSE)
_Arguments:_
depth  Whether to make a deep clone.

**References**


**See Also**

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRpart, LearnerSurvXgboost

---

**Description**

Calls _rpart::rpart_.

- crank is predicted using _rpart::predict.rpart_.

Parameter xval is set to 0 in order to save some computation time.

**Dictionary**

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

LearnerSurvRpart$new()
mlr_learners$get("surv.rpart")
lrn("surv.rpart")
Meta Information

- Type: "surv"
- Predict Types: crank, distr
- Feature Types: logical, integer, numeric, character, factor, ordered
- Properties: importance, missings, selected_features, weights
- Packages: rpart distr6 survival

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvRpart

Methods

Public methods:

- LearnerSurvRpart$new()
- LearnerSurvRpart$importance()
- LearnerSurvRpart$selected_features()
- LearnerSurvRpart$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvRpart$new()

Method importance(): The importance scores are extracted from the model slot variable.importance.

Usage:
LearnerSurvRpart$importance()

Returns: Named numeric().

Method selected_features(): Selected features are extracted from the model slot frame$var.

Usage:
LearnerSurvRpart$selected_features()

Returns: character().

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvRpart$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

References

See Also

Other survival learners:

LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH,
LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost,
LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan,
LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized,
LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvXgboost

LearnerSurvXgboost

 CoxPH models based on gradient boosted trees Survival Learner

Description

Calls xgboost::xgboost().

- distr is predicted by xgboost::predict.xgb.Booster()
- crank is predicted as the expectation of the survival distribution, distr

```
# eXtreme Gradient Boosting regression for Cox PH model. Calls xgboost::xgb.train() from package xgboost.

We changed the following defaults for this learner:

- Verbosity is reduced by setting verbose to 0.
- Number of boosting iterations nrounds is set to 1.
- The objective is set to survival:cox.
- The eval_metric is set to cox-nloglik
```

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function lrn():

```
LearnerSurvXgboost$new()
mldr_learners$get("surv.xgboost")
lrn("surv.xgboost")
```

Meta Information

- Type: "surv"
- Predict Types: crank, lp
- Feature Types: logical, integer, numeric
- Properties: importance, missings, weights
- Packages: xgboost

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvXgboost
Methods

Public methods:

• LearnerSurvXgboost$new()
• LearnerSurvXgboost$importance()
• LearnerSurvXgboost$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LearnerSurvXgboost$new()

Method importance(): The importance scores are calculated with xgboost::xgb.importance().

Usage:
LearnerSurvXgboost$importance()

Returns: Named numeric().

Method clone(): The objects of this class are cloneable with this method.

Usage:
LearnerSurvXgboost$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

References


See Also

Other survival learners: LearnerSurvBlackboost, LearnerSurvCVGlmnet, LearnerSurvCoxPH, LearnerSurvFlexible, LearnerSurvGBM, LearnerSurvGamboost, LearnerSurvGlmboost, LearnerSurvGlmnet, LearnerSurvKaplan, LearnerSurvMboost, LearnerSurvObliqueRSF, LearnerSurvPenalized, LearnerSurvRandomForestSRC, LearnerSurvRanger, LearnerSurvRpart

<table>
<thead>
<tr>
<th>MeasureDens</th>
<th>Density Measure</th>
</tr>
</thead>
</table>

Description

This measure specializes Measure for survival problems.

• task_type is set to "dens".
• Possible values for predict_type are "pdf" and "cdf".

Predefined measures can be found in the dictionary mlr3::mlr_measures.
**Super class**

`mlr3::Measure` -> `MeasureDens`

**Methods**

**Public methods:**

- `MeasureDens$new()`

**Method new():** Creates a new instance of this R6 class.

**Usage:**

```r
MeasureDens$new(
  id,
  range,
  minimize = NA,
  aggregator = NULL,
  properties = character(),
  predict_type = "pdf",
  task_properties = character(0L),
  packages = character(0L)
)
```

**Arguments:**

- `id` (character(1))
  - Identifier for the new instance.
- `range` (numeric(2))
  - Feasible range for this measure as `c(lower_bound,upper_bound)`. Both bounds may be infinite.
- `minimize` (logical(1))
  - Set to TRUE if good predictions correspond to small values, and to FALSE if good predictions correspond to large values. If set to NA (default), tuning this measure is not possible.
- `aggregator` (function(x))
  - Function to aggregate individual performance scores `x` where `x` is a numeric vector. If NULL, defaults to `mean()`.
- `properties` (character())
  - Properties of the measure. Must be a subset of `mlr_reflections$measure_properties`. Supported by mlr3:
    - "requires_task" (requires the complete `Task`),
    - "requires_learner" (requires the trained `Learner`),
    - "requires_train_set" (requires the training indices from the `Resampling`), and
    - "na_score" (the measure is expected to occasionally return NA or NaN).
- `predict_type` (character(1))
  - Required predict type of the `Learner`. Possible values are stored in `mlr_reflections$learner_predict_types`.
- `task_properties` (character())
  - Required task properties, see `Task`.
- `packages` (character())
  - Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via `requireNamespace()`.
MeasureDensLogloss

See Also

Default density measures: dens.logloss
Other Measure: MeasureSurv

Description

Calculates the cross-entropy, or logarithmic (log), loss.
The logloss, in the context of probabilistic predictions, is defined as the negative log probability density function, \( f \), evaluated at the observed value, \( y \),

\[
L(f, y) = -\log(f(y))
\]

Meta Information

- Type: "density"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: pdf

Super classes

mlr3::Measure -> mlr3proba::MeasureDens -> MeasureDensLogloss

Active bindings

eps  Returns eps parameter, see initialize.

Methods

Public methods:

- MeasureDensLogloss$new()
- MeasureDensLogloss$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
MeasureDensLogloss$new(eps = 1e-15)

Arguments:
eps (numeric(1))
  Very small number to set zero-valued predicted probabilities to in order to prevent errors in log(0) calculation.

Method clone(): The objects of this class are cloneable with this method.
MeasureSurv

Usage:
MeasureDensLogloss$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

Description
This measure specializes Measure for survival problems.

• task_type is set to "surv".
• Possible values for predict_type are "distr", "lp", "crank", and "response".

Predefined measures can be found in the dictionary mlr3::mlr_measures.

Super class
mlr3::Measure -> MeasureSurv

Methods

Public methods:

• MeasureSurv$new()

Method new(): Creates a new instance of this R6 class.

Usage:
MeasureSurv$new(
id,  
range,  
minimize = NA,  
aggregator = NULL,  
properties = character(),  
predict_type = "distr",  
task_properties = character(),  
packages = character()
)

Arguments:
id (character(1))  
  Identifier for the new instance.
range (numeric(2))  
  Feasible range for this measure as c(lower_bound, upper_bound). Both bounds may be infinite.
minimize (logical(1))
Set to TRUE if good predictions correspond to small values, and to FALSE if good predictions correspond to large values. If set to NA (default), tuning this measure is not possible.

aggregator (function(x))
Function to aggregate individual performance scores x where x is a numeric vector. If NULL, defaults to mean().

properties (character())
Properties of the measure. Must be a subset of mlr_reflections$measure_properties. Supported by mlr3:
- "requires_task" (requires the complete Task),
- "requires_learner" (requires the trained Learner),
- "requires_train_set" (requires the training indices from the Resampling), and
- "na_score" (the measure is expected to occasionally return NA or NaN).

predict_type (character(1))
Required predict type of the Learner. Possible values are stored in mlr_reflections$learner_predict_types.

task_properties (character())
Required task properties, see Task.

packages (character())
Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().

See Also
Default survival measures: surv.harrellC
Other Measure: MeasureDens

---

Description
This is an abstract class that should not be constructed directly.

Super classes
mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> MeasureSurvAUC

Methods
Public methods:
- MeasureSurvAUC$new()
- MeasureSurvAUC$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
MeasureSurvBeggC$new(integrated = TRUE, times, id, properties)

**Arguments:**

- `integrated` (logical(1))
  - If TRUE (default), returns the integrated score; otherwise, not integrated.
- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
- `id` (character(1))
  - Identifier for the new instance.
- `properties` (character())
  - Properties of the measure. Must be a subset of `mlr_reflections$measure_properties`. Supported by `mlr3`:
    - "requires_task" (requires the complete `Task`),
    - "requires_learner" (requires the trained `Learner`),
    - "requires_train_set" (requires the training indices from the `Resampling`), and
    - "na_score" (the measure is expected to occasionally return NA or NaN).

**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**

```r
MeasureSurvAUC$clone(deep = FALSE)
```

**Arguments:**

- `deep`  Whether to make a deep clone.

---

**MeasureSurvBeggC**  **Begg’s C-Index Survival Measure**

**Description**

Calls `survAUC::BeggC()`.

Assumes Cox PH model specification.

**Details**

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

**Dictionary**

This `Measure` can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvBeggC$new()
mlr_measures$get("surv.beggC")
msr("surv.beggC")
```
MeasureSurvBeggC

Meta Information

• Type: "surv"
• Range: [0, 1]
• Minimize: FALSE
• Required prediction: lp

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvBeggC

Methods

Public methods:

• MeasureSurvBeggC$new()
• MeasureSurvBeggC$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
MeasureSurvBeggC$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
MeasureSurvBeggC$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

References


See Also

Other survival measures: MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvGonenC, MeasureSurvHarrellIC, MeasureSurvUnoC

Other lp survival measures: MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2
MeasureSurvChamblessAUC

Chambless and Diao’s AUC Survival Measure

Description

Calls survAUC::AUC.cd().
Assumes Cox PH model specification.

Details

All measures implemented from survAUC should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvChamblessAUC$new()
mlr_measures$get("surv.chamblessAUC")
msr("surv.chamblessAUC")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> mlr3proba::MeasureSurvAUC -> MeasureSurvChamblessAUC

Methods

Public methods:

- `MeasureSurvChamblessAUC$new()`
- `MeasureSurvChamblessAUC$clone()`

Method new(): Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvChamblessAUC$new(integrated = TRUE, times)
```
MeasureSurvGonenC

Arguments:
integrated (logical(1))
    If TRUE (default), returns the integrated score; otherwise, not integrated.
times (numeric())
    If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

Method clone(): The objects of this class are cloneable with this method.

Usage:
MeasureSurvChamblessAUC$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

See Also
Other survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2
Other AUC survival measures: MeasureSurvHungAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR
Other lp survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

MeasureSurvGonenC  Gonen and Heller's C-Index Survival Measure

Description
Calls survAUC::GHCI()
Assumes Cox PH model specification.

Details
All measures implemented from survAUC should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.
Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvGonenC$new()
mlr_measures$get("surv.gonenC")
msr("surv.gonenC")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

```r
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvGonenC
```

Methods

Public methods:

- `MeasureSurvGonenC$new()`
- `MeasureSurvGonenC$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvGonenC$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
MeasureSurvGonenC$clone(deep = FALSE)
```

Arguments:

depth Whether to make a deep clone.

References

MeasureSurvGraf

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMASE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvHarrellC, MeasureSurvUnoC

Other lp survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

MeasureSurvGraf  Integrated Graf Score Survival Measure

Description

Calculates the Integrated Graf Score, aka integrated Brier score or squared loss.

For an individual who dies at time \( t \), with predicted Survival function, \( S \), the Graf Score at time \( t^* \) is given by

\[
L(S, t|t^*) = [(S(t^*)^2)I(t \leq t^*, \delta = 1)(1/G(t))] + [[[1 - S(t^*)^2]I(t > t^*)(1/G(t^*))]
\]

where \( G \) is the Kaplan-Meier estimate of the censoring distribution.

Note: If comparing the integrated graf score to other packages, e.g. `pec::pec()`, then method = 2 should be used, however the results may still be very slightly different as this package uses `survfit` to estimate the censoring distribution, in line with the Graf 1999 paper. Whereas some other packages use `prodlim` with `reverse = TRUE` (meaning Kaplan-Meier is not used).

If integrated == FALSE then the sample mean is taken for the single specified times, \( t^* \), and the returned score is given by

\[
L(S, t|t^*) = \frac{1}{N} \sum_{i=1}^{N} L(S_i, t_i|t^*)
\]

where \( N \) is the number of observations, \( S_i \) is the predicted survival function for individual \( i \) and \( t_i \) is their true survival time.

If integrated == TRUE then an approximation to integration is made by either taking the sample mean over all \( T \) unique time-points (method == 1), or by taking a mean weighted by the difference between time-points (method == 2). Then the sample mean is taken over all \( N \) observations.

\[
L(S) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{j=1}^{T} L(S_i, t_i|t_j^*)
\]
**Dictionary**

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvGraf$new()
mlr_measures$get("surv.graf")
msr("surv.graf")
```

**Meta Information**

- **Type:** "surv"
- **Range:** $[0, \infty)$
- **Minimize:** TRUE
- **Required prediction:** distr

**Super classes**

`mlr3::Measure` $\rightarrow$ `mlr3proba::MeasureSurv` $\rightarrow$ `mlr3proba::MeasureSurvIntegrated` $\rightarrow$ `MeasureSurvGraf`

**Methods**

**Public methods:**

- `MeasureSurvGraf$new()`
- `MeasureSurvGraf$clone()`

**Method new():** Creates a new instance of this R6 class.

Usage:

`MeasureSurvGraf$new(integrated = TRUE, times, method = 2)`

Arguments:

- `integrated` (logical(1))
  - If TRUE (default), returns the integrated score; otherwise, not integrated.
- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
- `method` (integer(1))
  - If integrate == TRUE selects the integration weighting method. method == 1 corresponds to weighting each time-point equally and taking the mean score over discrete time-points. method == 2 corresponds to calculating a mean weighted by the difference between time-points. method == 2 is default to be in line with other packages.

**Method clone():** The objects of this class are cloneable with this method.

Usage:

`MeasureSurvGraf$clone(deep = FALSE)`

Arguments:

- `deep` Whether to make a deep clone.
References


See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongAUC, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss

Other distr survival measures: MeasureSurvGrafSE, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss

---

**MeasureSurvGrafSE**

*Standard Error of Integrated Graf Score Survival Measure*

**Description**

Calculates the standard error of `MeasureSurvGraf`.

If `integrated` == FALSE then the standard error of the loss, L, is approximated via,

\[
se(L) = sd(L)/\sqrt{N}
\]

where \( N \) are the number of observations in the test set, and \( sd \) is the standard deviation.

If `integrated` == TRUE then correlations between time-points need to be taken into account, therefore

\[
se(L) = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{M} \frac{\Sigma_{i,j}}{NT^2}}
\]

where \( \Sigma_{i,j} \) is the sample covariance matrix over \( M \) distinct time-points.

**Dictionary**

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```
MeasureSurvGrafSE$new()
mlr_measures$get("surv.grafSE")
msr("surv.grafSE")
```
Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: distr

Super classes

```
mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> MeasureSurvGrafSE
```

Methods

Public methods:

- `MeasureSurvGrafSE$new()`
- `MeasureSurvGrafSE$clone()`

Method `new()`: Creates a new instance of this R6 class.

**Usage:**

`MeasureSurvGrafSE$new(integrated = TRUE, times)`

**Arguments:**

- `integrated` (logical(1))
  - If TRUE (default), returns the integrated score; otherwise, not integrated.
- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.

Method `clone()`: The objects of this class are cloneable with this method.

**Usage:**

`MeasureSurvGrafSE$clone(deep = FALSE)`

**Arguments:**

- `deep` Whether to make a deep clone.

References


See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvHungAUC, MeasureSurvMSE integr, MeasureSurvMAE integr, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvLogMSE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSE integr, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2
Other Probabilistic survival measures: \texttt{MeasureSurvGraf}, \texttt{MeasureSurvIntLoglossSE}, \texttt{MeasureSurvIntLogloss}, \texttt{MeasureSurvLoglossSE}, \texttt{MeasureSurvLogloss}

Other distr survival measures: \texttt{MeasureSurvGraf}, \texttt{MeasureSurvIntLoglossSE}, \texttt{MeasureSurvIntLogloss}, \texttt{MeasureSurvLoglossSE}, \texttt{MeasureSurvLogloss}

\begin{center}
\begin{tabular}{ll}
\textbf{MeasureSurvHarrellC} & \textit{Harrell's C-Index Survival Measure} \\
\end{tabular}
\end{center}

\textbf{Description}

Calculates Harrell's C, equivalent to the Fortran implementation in \texttt{Hmisc}.

\textbf{Dictionary}

This \texttt{Measure} can be instantiated via the dictionary \texttt{mlr_measures} or with the associated sugar function \texttt{msr():}

\begin{verbatim}
MeasureSurvHarrellC$new()
mlr_measures$get("surv.harrellC")
msr("surv.harrellC")
\end{verbatim}

\textbf{Meta Information}

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: crank

\textbf{Super classes}

\texttt{mlr3::Measure} $\rightarrow$ \texttt{mlr3proba::MeasureSurv} $\rightarrow$ \texttt{MeasureSurvHarrellC}

\textbf{Methods}

\textbf{Public methods:}

- \texttt{MeasureSurvHarrellC$new()}
- \texttt{MeasureSurvHarrellC$clone()}

\textbf{Method} \texttt{new()}: Creates a new instance of this \texttt{R6} class.

\textit{Usage:}
\begin{verbatim}
MeasureSurvHarrellC$new()
\end{verbatim}

\textbf{Method} \texttt{clone()}: The objects of this class are cloneable with this method.

\textit{Usage:}
\begin{verbatim}
MeasureSurvHarrellC$clone(deep = FALSE)
\end{verbatim}

\textit{Arguments:}
- \texttt{deep} Whether to make a deep clone.
References

See Also
Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2
Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvUnoC
Other crank survival measures: MeasureSurvUnoC

---

**MeasureSurvHungAUC**  
*Hung and Chiang’s AUC Survival Measure*

**Description**
Calls *survAUC::AUC.hc()*.
Assumes random censoring.

**Details**
All measures implemented from *survAUC* should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in *mlr3proba*.

**Dictionary**
This *Measure* can be instantiated via the dictionary *mlr_measures* or with the associated sugar function *msr()*:

```r
MeasureSurvHungAUC$new()
mlr_measures$get("surv.hungAUC")
msr("surv.hungAUC")
```

**Meta Information**
- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp
Super classes

\texttt{mlr3::Measure} -> \texttt{mlr3proba::MeasureSurv} -> \texttt{mlr3proba::MeasureSurvIntegrated} -> \texttt{mlr3proba::MeasureSurvAUC} -> \texttt{MeasureSurvHungAUC}

Methods

Public methods:

- \texttt{MeasureSurvHungAUC\$new()}
- \texttt{MeasureSurvHungAUC\$clone()}

Method \texttt{new()}: Creates a new instance of this R6 class.

Usage:

\texttt{MeasureSurvHungAUC\$new(integrated = TRUE, times)}

Arguments:

- \texttt{integrated (logical())}
  - If \texttt{TRUE} (default), returns the integrated score; otherwise, not integrated.
- \texttt{times (numeric())}
  - If integrate == \texttt{TRUE} then a vector of time-points over which to integrate the score. If integrate == \texttt{FALSE} then a single time point at which to return the score.

Method \texttt{clone()}: The objects of this class are cloneable with this method.

Usage:

\texttt{MeasureSurvHungAUC\$clone(deep = FALSE)}

Arguments:

- \texttt{deep} Whether to make a deep clone.

References


See Also

Other survival measures: \texttt{MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2}

Other AUC survival measures: \texttt{MeasureSurvChamblessAUC, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR}

Other lp survival measures: \texttt{MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2}
MeasureSurvIntegrated

Abstract Class for Integrated Measures

Description

This is an abstract class that should not be constructed directly.

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvIntegrated

Active bindings

integrated (logical(1)) Returns if the measure should be integrated or not. Settable.
times (numeric()) Returns the times at which the measure should be evaluated at, or integrated over. Settable.
method (integer(1)) Returns which method is used for approximating integration. Settable.

Methods

Public methods:

• MeasureSurvIntegrated$new()
• MeasureSurvIntegrated$clone()

Method new(): This is an abstract class that should not be constructed directly.

Usage:
MeasureSurvIntegrated$new(
    integrated = TRUE,
times,
method = 2,
id,
range,
minimize,
packages,
predict_type,
properties
)

Arguments:

integrated (logical(1))
    If TRUE (default), returns the integrated score; otherwise, not integrated.
times (numeric())
    If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
method (integer(1))
  If integrate == TRUE selects the integration weighting method. method == 1 corresponds to weighting each time-point equally and taking the mean score over discrete time-points. method == 2 corresponds to calculating a mean weighted by the difference between time-points. method == 2 is default to be in line with other packages.

id (character(1))
  Identifier for the new instance.

range (numeric(2))
  Feasible range for this measure as c(lower_bound,upper_bound). Both bounds may be infinite.

minimize (logical(1))
  Set to TRUE if good predictions correspond to small values, and to FALSE if good predictions correspond to large values. If set to NA (default), tuning this measure is not possible.

packages (character())
  Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().

predict_type (character(1))
  Required predict type of the Learner. Possible values are stored in mlr_reflections$learner_predict_types.

properties (character())
  Properties of the measure. Must be a subset of mlr_reflections$measure_properties. Supported by mlr3:
  • "requires_task" (requires the complete Task),
  • "requires_learner" (requires the trained Learner),
  • "requires_train_set" (requires the training indices from the Resampling), and
  • "na_score" (the measure is expected to occasionally return NA or NaN).

Method clone(): The objects of this class are cloneable with this method.

Usage:
MeasureSurvIntegrated$clone(deep = FALSE)

Arguments:
deepe Whether to make a deep clone.

---

MeasureSurvIntLogloss Integrated Log loss Survival Measure

Description

Calculates the integrated logarithmic (log), loss, aka integrated cross entropy.

For an individual who dies at time \( t \), with predicted Survival function, \( S \), the probabilistic log loss at time \( t^* \) is given by

\[
L(S, t|t^*) = -[\log(1 - S(t^*))I(t \leq t^*, \delta = 1)(1/G(t))] - [\log(S(t^*))I(t > t^*)(1/G(t^*))]
\]

where \( G \) is the Kaplan-Meier estimate of the censoring distribution.
If integrated == FALSE then the sample mean is taken for the single specified times, \( t^* \), and the returned score is given by

\[
L(S, t|t^*) = \frac{1}{N} \sum_{i=1}^{N} L(S_i, t_i|t^*)
\]

where \( N \) is the number of observations, \( S_i \) is the predicted survival function for individual \( i \) and \( t_i \) is their true survival time.

If integrated == TRUE then an approximation to integration is made by either taking the sample mean over all \( T \) unique time-points (method == 1), or by taking a mean weighted by the difference between time-points (method == 2). Then the sample mean is taken over all \( N \) observations.

\[
L(S) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{j=1}^{T} L(S_i, t_i|t_j^*)
\]

**Dictionary**

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvIntLogloss$new()
mlr_measures$get("surv.intlogloss")
msr("surv.intlogloss")
```

**Meta Information**

- **Type:** "surv"
- **Range:** \([0, \infty)\)
- **Minimize:** TRUE
- **Required prediction:** distr

**Super classes**

mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> MeasureSurvIntLogloss

**Active bindings**

- **eps** (numeric(1))
  - Very small number used to prevent log(0) error.

**Methods**

**Public methods:**

- `MeasureSurvIntLogloss$new()`
- `MeasureSurvIntLogloss$clone()`

**Method new():** Creates a new instance of this R6 class.

*Usage:*
MeasureSurvIntLogloss$new(integrated = TRUE, times, eps = 1e-15, method = 2)

**Arguments:**

- **integrated** (*logical(1)*)
  - If `TRUE` (default), returns the integrated score; otherwise, not integrated.

- **times** (*numeric()*)
  - If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.

- **eps** (*numeric(1)*)
  - Very small number to set zero-valued predicted probabilities to in order to prevent errors in `log(0)` calculation.

- **method** (*integer(1)*)
  - If `integrate == TRUE` selects the integration weighting method. `method == 1` corresponds to weighting each time-point equally and taking the mean score over discrete time-points. `method == 2` corresponds to calculating a mean weighted by the difference between time-points. `method == 2` is default to be in line with other packages.

**Method** clone(): The objects of this class are cloneable with this method.

**Usage:**

MeasureSurvIntLogloss$clone(deep = FALSE)

**Arguments:**

- **deep** Whether to make a deep clone.

**References**


**See Also**

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Probabilistic survival measures: MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss

Other distr survival measures: MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss
MeasureSurvIntLoglossSE

Standard Error of Integrated Log loss Survival Measure

Description

Calculates the standard error of MeasureSurvIntLogloss.

If \texttt{integrated} == \texttt{FALSE} then the standard error of the loss, \( L \), is approximated via,

\[
se(L) = \frac{sd(L)}{\sqrt{N}}
\]

where \( N \) are the number of observations in the test set, and \( sd \) is the standard deviation.

If \texttt{integrated} == \texttt{TRUE} then correlations between time-points need to be taken into account, therefore

\[
se(L) = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{M} \Sigma_{i,j} \over NT^2}
\]

where \( \Sigma_{i,j} \) is the sample covariance matrix over \( M \) distinct time-points.

Dictionary

This \texttt{Measure} can be instantiated via the dictionary \texttt{mlr_measures} or with the associated sugar function \texttt{msr()}:

\begin{verbatim}
MeasureSurvIntLoglossSE$new()
mlr_measures$get("surv.intloglossSE")
msr("surv.intloglossSE")
\end{verbatim}

Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: \texttt{TRUE}
- Required prediction: \texttt{distr}

Super classes

\texttt{mlr3::Measure} -> \texttt{mlr3proba::MeasureSurv} -> \texttt{mlr3proba::MeasureSurvIntegrated} -> \texttt{MeasureSurvIntLoglossSE}

Active bindings

\texttt{eps} (numeric(1))

Very small number used to prevent log(0) error.
Methods

Public methods:

- `MeasureSurvIntLoglossSE$new()`
- `MeasureSurvIntLoglossSE$clone()`

Method `new()`: Creates a new instance of this R6 class.

**Usage:**

`MeasureSurvIntLoglossSE$new(integrated = TRUE, times, eps = 1e-15)`

**Arguments:**

- `integrated` (logical(1))
  - If TRUE (default), returns the integrated score; otherwise, not integrated.
- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
- `eps` (numeric(1))
  - Very small number to set zero-valued predicted probabilities to in order to prevent errors in log(0) calculation.

Method `clone()`: The objects of this class are cloneable with this method.

**Usage:**

`MeasureSurvIntLoglossSE$clone(deep = FALSE)`

**Arguments:**

- `deep` Whether to make a deep clone.

References


See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellIC`, `MeasureSurvHungAUC`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other Probabilistic survival measures: `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`

Other distr survival measures: `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`
**MeasureSurvLogloss**

*Log loss Survival Measure*

**Description**

Calculates the cross-entropy, or logarithmic (log), loss. The logloss, in the context of probabilistic predictions, is defined as the negative log probability density function, \( f \), evaluated at the observation time, \( t \),

\[
L(f, t) = -\log(f(t))
\]

Censored observations in the test set are ignored.

**Dictionary**

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvLogloss$new()
mlr_measures$get("surv.logloss")
msr("surv.logloss")
```

**Meta Information**

- **Type:** "surv"
- **Range:** \([0, \infty)\)
- **Minimize:** TRUE
- **Required prediction:** distr

**Super classes**

```
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvLogloss
```

**Active bindings**

- **eps** (numeric(1))
  
  Very small number used to prevent log(0) error.

**Methods**

- **Public methods:**
  - `MeasureSurvLogloss$new()`
  - `MeasureSurvLogloss$clone()`

**Method** new(): Creates a new instance of this R6 class.

*Usage:*
MeasureSurvLogloss$\texttt{new}(\texttt{eps} = 1e-15, \texttt{id} = "surv.logloss")

**Arguments:**
- **\texttt{eps}** (numeric(1))
  - Very small number to set zero-valued predicted probabilities to in order to prevent errors in log(0) calculation.
- **\texttt{id}** (character(1))
  - Identifier for the new instance.

**Method** \texttt{clone()}: The objects of this class are cloneable with this method.

**Usage:**
- \texttt{MeasureSurvLogloss$\texttt{clone}(\texttt{deep} = \texttt{FALSE})}

**Arguments:**
- **\texttt{deep}** Whether to make a deep clone.

---

**Description**

Calculates the standard error of \texttt{MeasureSurvLogloss}.

The standard error of the Logloss, \(L\), is approximated via,

\[
se(L) = \frac{sd(L)}{\sqrt{N}}
\]

where \(N\) are the number of observations in the test set, and \(sd\) is the standard deviation.

Censored observations in the test set are ignored.

**Dictionary**

This \texttt{Measure} can be instantiated via the dictionary \texttt{mlr_measures} or with the associated sugar function \texttt{msr()}:

\[
\begin{align*}
\texttt{MeasureSurvLoglossSE$\texttt{new}()} \\
\texttt{mlr_measures$\texttt{get("surv.loglossSE")}} \\
\texttt{msr("surv.loglossSE")}
\end{align*}
\]
Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: distr

Super classes

`mlr3::Measure` -> `mlr3proba::MeasureSurv` -> `mlr3proba::MeasureSurvLogloss` -> `MeasureSurvLoglossSE`

Methods

**Public methods:**

- `MeasureSurvLoglossSE$new()`
- `MeasureSurvLoglossSE$clone()`

**Method new():** Creates a new instance of this R6 class.

**Usage:**

```r
MeasureSurvLoglossSE$new(eps = 1e-15)
```

**Arguments:**

- `eps` (numeric(1))
  
  Very small number to set zero-valued predicted probabilities to in order to prevent errors in log(0) calculation.

**Method clone():** The objects of this class are cloneable with this method.

**Usage:**

```r
MeasureSurvLoglossSE$clone(deep = FALSE)
```

**Arguments:**

- `deep`  Whether to make a deep clone.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellIC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvMAESE`, `MeasureSurvLoglossSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other Probabilistic survival measures: `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLogloss`

Other distr survival measures: `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLogloss`
MeasureSurvMAE  

Mean Absolute Error Survival Measure

Description

Calculates the mean absolute error (MAE).

The MAE is defined by

$$ \frac{1}{n} \sum |t - \hat{t}| $$

where $t$ is the true value and $\hat{t}$ is the prediction.
Censored observations in the test set are ignored.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvMAE$new()
mlr_measures$get("surv.mae")
msr("surv.mae")
```

Meta Information

- **Type**: "surv"
- **Range**: $[0, \infty)$
- **Minimize**: TRUE
- **Required prediction**: response

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvMAE

Methods

Public methods:
- `MeasureSurvMAE$new()`
- `MeasureSurvMAE$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:
```r
MeasureSurvMAE$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
```r
MeasureSurvMAE$clone(deep = FALSE)
```
Arguments:
- `deep` Whether to make a deep clone.
MeasureSurvMAESE

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other response survival measures: MeasureSurvMAESE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvRMSESE, MeasureSurvRMSE

---

MeasureSurvMAESE  Standard Error of Mean Absolute Error Survival Measure

Description

Calculates the standard error of MeasureSurvMAE.

The standard error of the MAE, L, is approximated via

\[ se(L) = \frac{sd(L)}{\sqrt{N}} \]

Censored observations in the test set are ignored.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```
MeasureSurvMAESE$new()
mlr_measures$get("surv.maeSE")
msr("surv.maeSE")
```

Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: response

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvMAESE
MeasureSurvMSE

Methods

Public methods:

• MeasureSurvMAESE$new()  
• MeasureSurvMAESE$clone()

Method new(): Creates a new instance of this R6 class.

Usage:

MeasureSurvMAESE$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:

MeasureSurvMAESE$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOquigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other response survival measures: MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvRMSESE, MeasureSurvRMSE

---*MeasureSurvMSE  Mean Squared Error Survival Measure*

Description

Calculates the mean squared error (MSE).

The MSE is defined by

$$\frac{1}{n} \sum ((t - \hat{t})^2)$$

where $t$ is the true value and $\hat{t}$ is the prediction.

Censored observations in the test set are ignored.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

MeasureSurvMSE$new()

mlr_measures$get("surv.mse")

msr("surv.mse")
MeasureSurvMSE

Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: response

Super classes

```
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvMSE
```

Methods

Public methods:

- `MeasureSurvMSE$new()`
- `MeasureSurvMSE$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:
```
MeasureSurvMSE$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
```
MeasureSurvMSE$clone(deep = FALSE)
```

Arguments:

- `deep` Whether to make a deep clone.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellIC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other response survival measures: `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvXuR2`
MeasureSurvMSESE

Standard Error of Mean Squared Error Survival Measure

Description
Calculates the standard error of MeasureSurvMSE.

The standard error of the MSE, \( L \), is approximated via

\[
se(L) = \frac{sd(L)}{\sqrt{N}}
\]

Censored observations in the test set are ignored.

Dictionary
This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvMSESE$new()
mlr_measures$get("surv.mseSE")
msr("surv.mseSE")
```

Meta Information
- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: response

Super classes
```
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvMSESE
```

Methods

Public methods:
- `MeasureSurvMSESE$new()`
- `MeasureSurvMSESE$clone()`

Method `new()`:
Creates a new instance of this R6 class.

Usage:
`MeasureSurvMSESE$new()`

Method `clone()`:
The objects of this class are cloneable with this method.

Usage:
`MeasureSurvMSESE$clone(deep = FALSE)`

Arguments:
depth Whether to make a deep clone.
See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other response survival measures: MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSE, MeasureSurvRMSESE, MeasureSurvRMSE

---

MeasureSurvNagelkR2  
Nagelkerke’s R2 Survival Measure

**Description**

Calls `survAUC::Nagelk()`.

Assumes Cox PH model specification.

**Details**

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

**Dictionary**

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvNagelkR2$new()
mlr_measures$get("surv.nagelkR2")
msr("surv.nagelkR2")
```

**Meta Information**

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

**Super classes**

`mlr3::Measure` -> `mlr3proba::MeasureSurv` -> `MeasureSurvNagelkR2`
Methods

**Public methods:**

- `MeasureSurvNagelkR2$new()`
- `MeasureSurvNagelkR2$clone()`

**Method `new()`:** Creates a new instance of this R6 class.

*Usage:*

`MeasureSurvNagelkR2$new()`

**Method `clone()`:** The objects of this class are cloneable with this method.

*Usage:*

`MeasureSurvNagelkR2$clone(deep = FALSE)`

*Arguments:*

- `deep` Whether to make a deep clone.

References


See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellIC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other R2 survival measures: `MeasureSurvOQuigleyR2`, `MeasureSurvXuR2`

Other lp survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvHungAUC`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

---

MeasureSurvOQuigleyR2  *O’Quigley, Xu, and Stare’s R2 Survival Measure*

Description

Calls `survAUC::OXS()`.

Assumes Cox PH model specification.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.
Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvOQuigleyR2$new()
mlr_measures$get("surv.oquigleyR2")
msr("surv.oquigleyR2")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

```r
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvOQuigleyR2
```

Methods

Public methods:

```r
• MeasureSurvOQuigleyR2$new()
• MeasureSurvOQuigleyR2$clone()
```

Method new(): Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvOQuigleyR2$new()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```r
MeasureSurvOQuigleyR2$clone(deep = FALSE)
```

Arguments:

- deep Whether to make a deep clone.

References

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other R2 survival measures: MeasureSurvNagelkR2, MeasureSurvXuR2

Other lp survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

---

**MeasureSurvRMSE**

**Root Mean Squared Error Survival Measure**

**Description**

Calculates the root mean squared error (RMSE).

The RMSE is defined by

\[
\sqrt{\frac{1}{n} \sum (t - \hat{t})^2}
\]

where \( t \) is the true value and \( \hat{t} \) is the prediction.

Censored observations in the test set are ignored.

**Dictionary**

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvRMSE$new()
multi_measures$get("surv.rmse")
msr("surv.rmse")
```

**Meta Information**

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: response

**Super classes**

mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvRMSE
Methods

Public methods:

- `MeasureSurvRMSE$new()`
- `MeasureSurvRMSE$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:
`MeasureSurvRMSE$new()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
`MeasureSurvRMSE$clone(deep = FALSE)`

Arguments:
- `deep` Whether to make a deep clone.

See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAE`, `MeasureSurvMSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other response survival measures: `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvRMSESE`

---

**Description**

Calculates the standard error of `MeasureSurvRMSE`.

The standard error of the RMSE, $L$, is approximated via first order approximation by

$$sd(RMSE) = Var(\sqrt{MSE})$$

Censored observations in the test set are ignored.

**Dictionary**

This `Measure` can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

`MeasureSurvRMSESE$new()`

`mlr_measures$get("surv.rmseSE")`

`msr("surv.rmseSE")`
Meta Information

- Type: "surv"
- Range: \([0, \infty)\)
- Minimize: TRUE
- Required prediction: response

Super classes

```
mlr3::Measure \rightarrow mlr3proba::MeasureSurv \rightarrow MeasureSurvRMSESE
```

Methods

Public methods:

- MeasureSurvRMSESE$new()
- MeasureSurvRMSESE$clone()

Method `new()`: Creates a new instance of this R6 class.

Usage:

```
MeasureSurvRMSESE$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
MeasureSurvRMSESE$clone(deep = FALSE)
```

Arguments:

deep  Whether to make a deep clone.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other response survival measures: MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvRMSE
MeasureSurvSongAUC  

Song and Zhou’s AUC Survival Measure

Description

Calls `survAUC::AUC.sh()`.
Assumes Cox PH model specification.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Dictionary

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvSongAUC$new()
mlr_measures$get("surv.songAUC")
msr("surv.songAUC")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

```
mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> mlr3proba::MeasureSurvAUC
-> MeasureSurvSongAUC
```

Active bindings

- `type` (character(1))
  Type of measure, one of: 'cumulative', 'incident'.

Methods

Public methods:
- `MeasureSurvSongAUC$new()`
- `MeasureSurvSongAUC$clone()`

Method `new()`: Creates a new instance of this R6 class.
**Usage:**

```r
MeasureSurvSongAUC$new(
  integrated = TRUE,
  times,
  type = c("incident", "cumulative")
)
```

**Arguments:**

- `integrated` (logical(1))
  - If TRUE (default), returns the integrated score; otherwise, not integrated.
- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
- `type` (character(1))
  - Determines the type of score, one of: 'cumulative', 'incident'.

**Method** `clone()`: The objects of this class are cloneable with this method.

**Usage:**

```r
MeasureSurvSongAUC$clone(deep = FALSE)
```

**Arguments:**

- `deep`  Whether to make a deep clone.

**References**


**See Also**

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other AUC survival measures: `MeasureSurvChamblessAUC`, `MeasureSurvHungAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`

Other lp survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvHungAUC`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`
MeasureSurvSongTNR  

Song and Zhou's TNR Survival Measure

Description

Calls `survAUC::spec.sh()`.

Assumes Cox PH model specification.

times and lp_thesh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Dictionary

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvSongTNR$new()
mlr_measures$get("surv.songTNR")
msr("surv.songTNR")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

```
mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> mlr3proba::MeasureSurvAUC -> MeasureSurvSongTNR
```

Active bindings

`lp_thesh` numeric(1)

Threshold for linear predictor when calculating TPR/TNR.
MeasureSurvSongTNR

Methods

Public methods:

- MeasureSurvSongTNR$new()
- MeasureSurvSongTNR$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
MeasureSurvSongTNR$new(times = 0, lp_thresh = 0)

Arguments:
- times (numeric())
  If integrate == TRUE then a vector of time-points over which to integrate the score. If
  integrate == FALSE then a single time point at which to return the score.
- lp_thresh numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Method clone(): The objects of this class are cloneable with this method.

Usage:
MeasureSurvSongTNR$clone(deep = FALSE)

Arguments:
- deep Whether to make a deep clone.

References

Song X, Zhou X (2008). “A semiparametric approach for the covariate specific ROC curve with

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC,
MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrellIIC, MeasureSurvHungAUC, MeasureSurvIntLoglossSE,
MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE,
MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2,
MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTPR, MeasureSurvUnoAUC,
MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC,
MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

Other lp survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC,
MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC,
MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2
MeasureSurvSongTPR

**Song and Zhou’s TPR Survival Measure**

**Description**

Calls `survAUC::sens.sh()`.
Assumes Cox PH model specification.
times and lp_thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

**Details**

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

**Dictionary**

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvSongTPR$new()
mlr_measures$get("surv.songTPR")
msr("surv.songTPR")
```

**Meta Information**

- **Type**: "surv"
- **Range**: [0, 1]
- **Minimize**: FALSE
- **Required prediction**: lp

**Super classes**

`mlr3::Measure` -> `mlr3proba::MeasureSurv` -> `mlr3proba::MeasureSurvIntegrated` -> `mlr3proba::MeasureSurvAUC` -> MeasureSurvSongTPR

**Active bindings**

- **lp_thresh** numeric(1)
  Threshold for linear predictor when calculating TPR/TNR.
- **type** (character(1))
  Type of measure, one of: 'cumulative', 'incident'.
Methods

Public methods:

- `MeasureSurvSongTPR$new()`
- `MeasureSurvSongTPR$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvSongTPR$new(
  times = 0,
  lp_thresh = 0,
  type = c("incident", "cumulative")
)
```

Arguments:

- `times` (numeric())
  - If integrate == TRUE then a vector of time-points over which to integrate the score. If integrate == FALSE then a single time point at which to return the score.
- `lp_thresh` (numeric(1))
  - Determines where to threshold the linear predictor for calculating the TPR/TNR.
- `type` (character(1))
  - Determines the type of score, one of: 'cumulative', 'incident'.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
MeasureSurvSongTPR$clone(deep = FALSE)
```

Arguments:

- `deep` Whether to make a deep clone.

References


See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`

Other AUC survival measures: `MeasureSurvChamblessAUC`, `MeasureSurvHungAUC`, `MeasureSurvSongAUC`, `MeasureSurvSongTPR`, `MeasureSurv UnoAUC`, `MeasureSurv UnoC`, `MeasureSurv UnoTNR`, `MeasureSurv UnoTPR`

Other lp survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvHungAUC`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`, `MeasureSurvUnoTPR`, `MeasureSurvXuR2`
MeasureSurvUnoAUC

Uno’s AUC Survival Measure

Description

Calls `survAUC::AUC.uno()`.
Assumes random censoring.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Dictionary

This `Measure` can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvUnoAUC$new()
mlr_measures$get("surv.unoAUC")
msr("surv.unoAUC")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

`mlr3::Measure` -> `mlr3proba::MeasureSurv` -> `mlr3proba::MeasureSurvIntegrated` -> `mlr3proba::MeasureSurvAUC` -> `MeasureSurvUnoAUC`

Methods

Public methods:

- `MeasureSurvUnoAUC$new()`
- `MeasureSurvUnoAUC$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvUnoAUC$new(integrated = TRUE, times)
```

Arguments:
MeasureSurvUnoC

Uno's C-Index Survival Measure

**Description**

Calls `survAUC::UnoC()`.

Assumes random censoring.

**Details**

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`. 
MeasureSurvUnoC

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvUnoC$new()
mldr_measures$get("surv.unoC")
msr("surv.unoC")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: crank

Super classes

`mlr3::Measure` $\rightarrow$ `mlr3proba::MeasureSurv` $\rightarrow$ `MeasureSurvUnoC`

Methods

Public methods:

- `MeasureSurvUnoC$new()`
- `MeasureSurvUnoC$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvUnoC$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
MeasureSurvUnoC$clone(deep = FALSE)
```

Arguments:

- `deep` Whether to make a deep clone.

References

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrell1C, MeasureSurvHungAUC, MeasureSurvIntLoglossSE, MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTNR, MeasureSurvUnoTPR, MeasureSurvXuR2

Other Concordance survival measures: MeasureSurvBeggC, MeasureSurvGonenC, MeasureSurvHarrell1C

Other crank survival measures: MeasureSurvHarrell1C

Description

Calls survAUC::spec.uno().

Assumes random censoring.

times and lp_thres are arbitrarily set to 0 to prevent crashing, these should be further specified.

Details

All measures implemented from survAUC should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

MeasureSurvUnoTNR$new()
mlr_measures$get("surv.unoTNR")
msr("surv.unoTNR")

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> mlr3proba::MeasureSurvAUC -> MeasureSurvUnoTNR
Active bindings

lp_thresh numeric(1)
Threshold for linear predictor when calculating TPR/TNR.

Methods

Public methods:

- MeasureSurvUnoTNR$new()
- MeasureSurvUnoTNR$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
MeasureSurvUnoTNR$new(times = 0, lp_thresh = 0)

Arguments:

times (numeric())
  If integrate == TRUE then a vector of time-points over which to integrate the score. If
  integrate == FALSE then a single time point at which to return the score.

lp_thresh numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Method clone(): The objects of this class are cloneable with this method.

Usage:
MeasureSurvUnoTNR$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.

References

Censored Regression Models.” Journal of the American Statistical Association, 102(478), 527–
537. doi: 10.1198/016214507000000149.

See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC,
MeasureSurvGrafSE, MeasureSurvGraf, MeasureSurvHarrell1C, MeasureSurvHungAUC, MeasureSurvIntLoglossSE,
MeasureSurvIntLogloss, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAE,
MeasureSurvMAESE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2,
MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR,
MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTPR, MeasureSurvXuR2

Other lp survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC,
MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC,
MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTPR, MeasureSurvXuR2

Other AUC survival measures: MeasureSurvChamblessAUC, MeasureSurvHungAUC, MeasureSurvSongAUC,
MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoTPR
MeasureSurvUnoTPR  Uno's TPR Survival Measure

Description

Calls `survAUC::sens.uno()`.

Assumes random censoring.

times and lp.thresh are arbitrarily set to 0 to prevent crashing, these should be further specified.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in mlr3proba.

Dictionary

This Measure can be instantiated via the dictionary mlr_measures or with the associated sugar function msr():

```r
MeasureSurvUnoTPR$new()
mlr_measures$get("surv.unoTPR")
msr("surv.unoTPR")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

`mlr3::Measure -> mlr3proba::MeasureSurv -> mlr3proba::MeasureSurvIntegrated -> mlr3proba::MeasureSurvAUC -> MeasureSurvUnoTPR`

Active bindings

lp.thresh numeric(1)

Threshold for linear predictor when calculating TPR/TNR.
Methods

Public methods:

• `MeasureSurvUnoTPR$new()`
• `MeasureSurvUnoTPR$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:
`MeasureSurvUnoTPR$new(times = 0, lp_thresh = 0)`

Arguments:
- `times` (numeric())
  If `integrate == TRUE` then a vector of time-points over which to integrate the score. If `integrate == FALSE` then a single time point at which to return the score.
- `lp_thresh` numeric(1)
  Determines where to threshold the linear predictor for calculating the TPR/TNR.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
`MeasureSurvUnoTPR$clone(deep = FALSE)`

Arguments:
- `deep` Whether to make a deep clone.

References


See Also

Other survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvGrafSE`, `MeasureSurvGraf`, `MeasureSurvHarrellC`, `MeasureSurvHungAUC`, `MeasureSurvIntLoglossSE`, `MeasureSurvIntLogloss`, `MeasureSurvLoglossSE`, `MeasureSurvLogloss`, `MeasureSurvMAESE`, `MeasureSurvMAE`, `MeasureSurvMSESE`, `MeasureSurvMSE`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvRMSESE`, `MeasureSurvRMSE`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoC`, `MeasureSurvUnoTNR`, `MeasureSurvXuR2`

Other AUC survival measures: `MeasureSurvChamblessAUC`, `MeasureSurvHungAUC`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`

Other lp survival measures: `MeasureSurvBeggC`, `MeasureSurvChamblessAUC`, `MeasureSurvGonenC`, `MeasureSurvHungAUC`, `MeasureSurvNagelkR2`, `MeasureSurvOQuigleyR2`, `MeasureSurvSongAUC`, `MeasureSurvSongTNR`, `MeasureSurvSongTPR`, `MeasureSurvUnoAUC`, `MeasureSurvUnoTNR`, `MeasureSurvXuR2`
Description

Calls `survAUC::XO()`.
Assumes Cox PH model specification.

Details

All measures implemented from `survAUC` should be used with care, we are aware of problems in implementation that sometimes cause fatal errors in R. In future updates these measures will all be re-written and implemented directly in `mlr3proba`.

Dictionary

This Measure can be instantiated via the dictionary `mlr_measures` or with the associated sugar function `msr()`:

```r
MeasureSurvXuR2$new()
mlr_measures$get("surv.xuR2")
msr("surv.xuR2")
```

Meta Information

- Type: "surv"
- Range: [0, 1]
- Minimize: FALSE
- Required prediction: lp

Super classes

```r
mlr3::Measure -> mlr3proba::MeasureSurv -> MeasureSurvXuR2
```

Methods

Public methods:

- `MeasureSurvXuR2$new()`
- `MeasureSurvXuR2$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
MeasureSurvXuR2$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
```
MeasureSurvXuR2$clone(deep = FALSE)

Arguments:
- deep: Whether to make a deep clone.

References


See Also

Other survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvGrafSE, MeasureSurvIntLoglossSE, MeasureSurvLoglossSE, MeasureSurvLogloss, MeasureSurvMAESE, MeasureSurvMAE, MeasureSurvMSESE, MeasureSurvMSE, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvRMSESE, MeasureSurvRMSE, MeasureSurvAUC, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

Other R2 survival measures: MeasureSurvNagelkR2, MeasureSurvOQuigleyR2

Other lp survival measures: MeasureSurvBeggC, MeasureSurvChamblessAUC, MeasureSurvGonenC, MeasureSurvHungAUC, MeasureSurvNagelkR2, MeasureSurvOQuigleyR2, MeasureSurvSongAUC, MeasureSurvSongTNR, MeasureSurvSongTPR, MeasureSurvUnoAUC, MeasureSurvUnoC, MeasureSurvUnoTNR, MeasureSurvUnoTPR

mlr_tasks_faithful

Old Faithful Eruptions Density Task

Description

A density task for the faithful data set.

Format

R6::R6Class inheriting from TaskDens.

Details

Only the eruptions column is kept in this task.

Construction

mlr3::mlr_tasks$get("faithful")
mlr3::tsk("faithful")
**mlr_tasks_lung**  
*Lung Cancer Survival Task*

**Description**

A survival task for the `lung` data set.

**Format**

*R6::R6Class* inheriting from *TaskSurv*.

**Details**

Column "sex" has been converted to a factor, all others have been converted to integer.

**Construction**

```r
mlr3::mlr_tasks$get("lung")
mlr3::tsk("lung")
```

---

**mlr_tasks_precip**  
*Annual Precipitation Density Task*

**Description**

A density task for the `precip` data set.

**Format**

*R6::R6Class* inheriting from *TaskDens*.

**Construction**

```r
mlr3::mlr_tasks$get("precip")
mlr3::tsk("precip")
```
### Rats Survival Task

**Description**
A survival task for the rats data set.

**Format**
R6::R6Class inheriting from TaskSurv.

**Details**
Column "sex" has been converted to a factor, all others have been converted to integer.

**Construction**
```r
mlr3::mlr_tasks$get("rats")
mlr3::tsk("rats")
```

### Unemployment Duration Survival Task

**Description**
A survival task for the UnempDur data set.

**Format**
R6::R6Class inheriting from TaskSurv.

**Details**
A survival task for the "UnempDur" data set in package Ecdat. Contains the following columns of the original data set: "spell" (time), "censor1" (status), "age", "ui", "logwage", and "tenure".

**Construction**
```r
mlr3::mlr_tasks$get("unemployment")
mlr3::tsk("unemployment")
```

**See Also**
Dictionary of Tasks: mlr3::mlr_tasks
mlr_task_generators_simdens

Density Task Generator for Package 'distr6'

Description

A mlr3::TaskGenerator calling distr6::distrSimulate() from package simsurv. See distr6::distrSimulate() for an explanation of the hyperparameters.

Dictionary

This TaskGenerator can be instantiated via the dictionary mlr_task_generators or with the associated sugar function tgen():

mlr_task_generators$get("simdens")
tgen("simdens")

Super class

mlr3::TaskGenerator -> TaskGeneratorSimdens

Methods

Public methods:

• TaskGeneratorSimdens$new()
• TaskGeneratorSimdens$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
TaskGeneratorSimdens$new()

Method clone(): The objects of this class are cloneable with this method.

Usage:
TaskGeneratorSimdens$clone(deep = FALSE)

Arguments:

deepp Whether to make a deep clone.

See Also

Dictionary of TaskGenerators: mlr3::mlr_task_generators

Examples

generator = mlr3::mlr_task_generators$get("simdens")
task = generator$generate(20)
task$head()
mlr3::TaskGenerator calling `simsurv::simsurv()` from package `simsurv`.

This generator currently only exposes a small subset of the flexibility of `simsurv`, and just creates a small data set with the following numerical covariates:

- `treatment`: Bernoulli distributed with log hazard ratio $-0.5$.
- `height`: Normally distributed with log hazard ratio $1$.
- `weight`: normally distributed with log hazard ratio $0$.

See `simsurv::simsurv()` for an explanation of the hyperparameters.

Dictionary

This `TaskGenerator` can be instantiated via the dictionary `mlr_task_generators` or with the associated sugar function `tgen()`:

```r
mlr_task_generators$get("simsurv")
tgen("simsurv")
```

Super class

`mlr3::TaskGenerator` $\rightarrow$ `TaskGeneratorSimsurv`

Methods

**Public methods:**

- `TaskGeneratorSimsurv$new()`
- `TaskGeneratorSimsurv$clone()`

**Method** `new()`: Creates a new instance of this `R6` class.

Usage:

```r
TaskGeneratorSimsurv$new()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

Usage:

```r
TaskGeneratorSimsurv$clone(deep = FALSE)
```

Arguments:

- `deep` Whether to make a deep clone.
pecs

Prediction Error Curves for PredictionSurv and LearnerSurv

Description

Methods to plot prediction error curves (pecs) for either a PredictionSurv object or a list of trained LearnerSurvs.

Usage

```r
pecs(x, measure = c("graf", "logloss"), times, n, eps = 1e-15, ...)
```

## S3 method for class 'list'
```r
pecs(
  x,
  measure = c("graf", "logloss"),
  times,
  n,
  eps = 1e-15,
  task = NULL,
  row_ids = NULL,
  newdata,
  ...
)
```

## S3 method for class 'PredictionSurv'
```r
pecs(x, measure = c("graf", "logloss"), times, n, eps = 1e-15, ...)
```

Arguments

- **x**
  - (PredictionSurv or list of LearnerSurvs)
- **measure**
  - (character(1))
  - Either "graf" for MeasureSurvGraf, or "logloss" for MeasureSurvIntLogloss
- **times**
  - (numeric())
  - If provided then either a vector of time-points to evaluate measure or a range of time-points.

Examples

```r
generator = mlr3::mlr_task_generators$get("simsurv")
task = generator$generate(20)
task$head()
```
pecs

n
(integer())
If times is missing or given as a range, then n provide number of time-points to evaluate measure over.

eps
(numeric())
Small error value to pass to MeasureSurvIntLogloss to prevent errors resulting from a log(0) calculation.

... Additional arguments.

task
(TaskSurv)

row_ids
(integer())
Passed to Learner$predict.

newdata
(data.frame())
If not missing Learner$predict_newdata is called instead of Learner$predict.

Details

If times and n are missing then measure is evaluated over all observed time-points from the PredictionSurv or TaskSurv object. If a range is provided for times without n, then all time-points between the range are returned.

Examples

## Not run:
library(mlr3)
task = tsk("rats")

# Prediction Error Curves for prediction object
learn = lrn("surv.coxph")
p = learn$train(task)$predict(task)
pecs(p)
pecs(p, measure = "logloss", times = c(20, 40, 60, 80)) +
  ggplot2::geom_point() +
  ggplot2::ggtitle("Logloss Prediction Error Curve for Cox PH")

# Access underlying data
x = pecs(p)
x$data

# Prediction Error Curves for fitted learners
learns = lrns(c("surv.kaplan", "surv.coxph", "surv.ranger"))
lapply(learns, function(x) x$train(task))
pecs(learns, task = task, measure = "logloss", times = c(20, 90), n = 10)

## End(Not run)
PipeOpCrankCompositor

Description
Uses a predicted distr in a PredictionSurv to estimate (or 'compose') a crank prediction.

Dictionary
This PipeOp can be instantiated via the dictionary mlr3pipelines::mlr_pipeops or with the associated sugar function mlr3pipelines::po():

PipeOpCrankCompositor$new()
mlr_pipeops$get("crankcompose")
po("crankcompose")

Input and Output Channels
PipeOpCrankCompositor has one input channel named "input", which takes NULL during training and PredictionSurv during prediction.
PipeOpCrankCompositor has one output channel named "output", producing NULL during training and a PredictionSurv during prediction.
The output during prediction is the PredictionSurv from the "pred" input but with the crank predict type overwritten by the given estimation method.

State
The $state is left empty (list()).

Parameters
• method :: character(1)
  Determines what method should be used to produce a continuous ranking from the distribution. One of median, mode, or mean corresponding to the respective functions in the predicted survival distribution. Note that for models with a proportional hazards form, the ranking implied by mean and median will be identical (but not the value of crank itself). Default is mean.

Internals
The median, mode, or mean will use analytical expressions if possible but if not they are calculated using distr6::median.Distribution, distr6::mode, or distr6::mean.Distribution respectively.

Fields
Only fields inherited from PipeOp.
PipeOpCrankCompositor

Methods
Only fields inherited from PipeOp.

Super class
mlr3pipelines::PipeOp -> PipeOpCrankCompositor

Methods

Public methods:
• PipeOpCrankCompositor$new()
• PipeOpCrankCompositor$train_internal()
• PipeOpCrankCompositor$predict_internal()
• PipeOpCrankCompositor$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
PipeOpCrankCompositor$new(id = "crankcompose",
                          param_vals = list(method = "mean")
)

Arguments:
id (character(1))
   Identifier of the resulting object.
param_vals (list())
   List of hyperparameter settings, overwriting the hyperparameter settings that would otherwise be set during construction.

Method train_internal(): train_internal Internal train function, will be moved to private in a near-future update, should be ignored.

Usage:
PipeOpCrankCompositor$train_internal(inputs)

Arguments:
inputs Ignore.

Method predict_internal(): predict_internal Internal predict function, will be moved to private in a near-future update, should be ignored.

Usage:
PipeOpCrankCompositor$predict_internal(inputs)

Arguments:
inputs Ignore.

Method clone(): The objects of this class are cloneable with this method.

Usage:
PipeOpCrankCompositor$clone(deep = FALSE)

Arguments:
dee Whether to make a deep clone.
See Also

- mlr3pipelines::PipeOp and crankcompositor
- Other survival compositors: PipeOpDistrCompositor

Examples

```r
library(mlr)
library(mlr3pipelines)
set.seed(1)

# Three methods to predict a 'crank' from 'surv.rpart'
task = tgen("simsurv")$generate(30)

# Method 1 - Train and predict separately then compose
learn = lrn("surv.coxph")$train(task)$predict(task)
poc = po("crankcompose", param_vals = list(method = "mean"))
poc$predict(list(learn))

# Examples not run to save run-time.
## Not run:
# Method 2 - Create a graph manually
gr = Graph$new()
  add_pipeop(po("learner", lrn("surv.ranger")))
  add_pipeop(po("crankcompose"))
  add_edge("surv.ranger", "crankcompose")
gr$train(task)
gr$predict(task)

# Method 3 - Syntactic sugar: Wrap the learner in a graph
ranger.crank = crankcompositor(learner = lrn("surv.ranger"),
  method = "median")
resample(task, ranger.crank, rsmp("cv", folds = 2))$predictions()
## End(Not run)
```

Description

Estimates (or 'composes') a survival distribution from a predicted baseline distr and a crank or lp from two PredictionSurvs.

Compositor Assumptions:

- The baseline distr is a discrete estimator, i.e. surv.kaplan or surv.nelson.
- The composed distr is of a linear form
- If lp is missing then crank is equivalent

These assumptions are strong and may not be reasonable. Future updates will upgrade this compositor to be more flexible.
PipeOpDistrCompositor

Dictionary

This PipeOp can be instantiated via the dictionary mlr3pipelines::mlr_pipeops or with the associated sugar function mlr3pipelines::po()

```r
PipeOpDistrCompositor$new()
mlr_pipeops$get("distrcompose")
po("distrcompose")
```

Input and Output Channels

PipeOpDistrCompositor has two input channels, "base" and "pred". Both input channels take NULL during training and PredictionSurv during prediction.

PipeOpDistrCompositor has one output channel named "output", producing NULL during training and a PredictionSurv during prediction.

The output during prediction is the PredictionSurv from the "pred" input but with an extra (or overwritten) column for distr predict type; which is composed from the distr of "base" and lp or crank of "pred".

State

The $state is left empty (list()).

Parameters

The parameters are:

- **form**: character(1)
  - Determines the form that the predicted linear survival model should take. This is either, accelerated-failure time, aft, proportional hazards, ph, or proportional odds, po. Default aft.

- **overwrite**: logical(1)
  - If FALSE (default) then if the "pred" input already has a distr, the compositor does nothing and returns the given PredictionSurv. If TRUE then the distr is overwritten with the distr composed from lp/crank - this is useful for changing the prediction distr from one model form to another.

Internals

The respective forms above have respective survival distributions:

```
aft : S(t) = S_0(\frac{t}{\exp(lp)})
ph : S(t) = S_0(t)^{\exp(lp)}
po : S(t) = \frac{S_0(t)}{\exp(-lp) + (1 - \exp(-lp))S_0(t)}
```

where $S_0$ is the estimated baseline survival distribution, and $lp$ is the predicted linear predictor. If the input model does not predict a linear predictor then crank is assumed to be the lp - **this may be a strong and unreasonable assumption**.
Fields

Only fields inherited from PipeOp.

Methods

Only methods inherited from PipeOp.

Super class

mlr3pipelines::PipeOp -> PipeOpDistrCompositor

Methods

Public methods:

- `PipeOpDistrCompositor$new()`
- `PipeOpDistrCompositor$train_internal()`
- `PipeOpDistrCompositor$predict_internal()`
- `PipeOpDistrCompositor$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
PipeOpDistrCompositor$new(
  id = "distrcompose",
  param_vals = list(form = "aft", overwrite = FALSE)
)
```

Arguments:

- `id` (character(1))
  - Identifier of the resulting object.
- `param_vals` (list())
  - List of hyperparameter settings, overwriting the hyperparameter settings that would otherwise be set during construction.

Method `train_internal()`: `train_internal` Internal train function, will be moved to private in a near-future update, should be ignored.

Usage:

```r
PipeOpDistrCompositor$train_internal(inputs)
```

Arguments:

- `inputs` Ignore.

Method `predict_internal()`: `predict_internal` Internal predict function, will be moved to private in a near-future update, should be ignored.

Usage:

```r
PipeOpDistrCompositor$predict_internal(inputs)
```

Arguments:

- `inputs` Ignore.
Method clone(): The objects of this class are cloneable with this method.

Usage:
PipeOpDistrCompositor$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
mlr3pipelines::PipeOp and distrcompositor
Other survival compositors: PipeOpCrankCompositor

Examples

library(mlr3)
library(mlr3pipelines)
set.seed(42)

# Three methods to transform the cox ph predicted \`distr\' to an
# accelerated failure time model
task = tgen("simsurv")$generate(30)

# Method 1 - Train and predict separately then compose
base = lrn("surv.kaplan")$train(task)$predict(task)
pred = lrn("surv.coxph")$train(task)$predict(task)
pod = po("distrcompose", param_vals = list(form = "aft", overwrite = TRUE))
pod$predict(list(base = base, pred = pred))

# Method 2 - Create a graph manually
gr = Graph$new()
add_pipeop(po("learner", lrn("surv.kaplan")))
add_pipeop(po("learner", lrn("surv.glmnet")))
add_pipeop(po("distrcompose"))
add_edge("surv.kaplan", "distrcompose", dst_channel = "base")
add_edge("surv.glmnet", "distrcompose", dst_channel = "pred")
gr$train(task)$predict(task)

# Method 3 - Syntactic sugar: Wrap the learner in a graph.
cvglm.distr = distrcompositor(learner = lrn("surv.cvglmnet"),
 estimator = "kaplan",
 form = "aft")
cvglm.distr$fit(task)$predict(task)
plot.LearnerSurv

Visualization of fitted LearnerSurv objects

Description

Wrapper around predict.LearnerSurv and plot.VectorDistribution.

Usage

## S3 method for class 'LearnerSurv'
plot(
  x,
  task,
  fun = c("survival", "pdf", "cdf", "quantile", "hazard", "cumhazard"),
  row_ids = NULL,
  newdata,
  ...
)

Arguments

x (LearnerSurv)  
task (TaskSurv)  
fun (character)  
  Passed to distr6::plot.VectorDistribution
row_ids (integer())  
  Passed to Learner$predict
newdata (data.frame())  
  If not missing Learner$predict_newdata is called instead of Learner$predict.
...  
  Additional arguments passed to distr6::plot.VectorDistribution

Examples

library(mlr3)
task = tsk("rats")

# Prediction Error Curves for prediction object
learn = lrn("surv.coxph")
learn$train(task)

plot(learn, task, "survival", ind = 10)
plot(learn, task, "survival", fun = c("survival", "pdf", "cdf", "quantile", "hazard", "cumhazard"), row_ids = NULL, newdata, ...)
plot(learn, task, "survival", newdata = task$data()[1:5,])
plot(learn, task, "survival", newdata = task$data()[1:5,], ylim=c(0, 1))
Description

This object stores the predictions returned by a learner of class LearnerDens.
The task_type is set to "dens".

Super class

mlr3::Prediction -> PredictionDens

Active bindings

pdf (numeric())
Access the stored predicted probability density function.
cdf (numeric())
Access the stored predicted cumulative distribution function.
missing (integer())
Returns row_ids for which the predictions are missing or incomplete.

Methods

Public methods:
• PredictionDens$new()
• PredictionDens$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
PredictionDens$new(
  task = NULL,
  row_ids = task$row_ids,
  truth = task$truth(),
  pdf = NULL,
  cdf = NULL
)

Arguments:
task (TaskSurv)
  Task, used to extract defaults for row_ids and truth.
row_ids (integer())
  Row ids of the predicted observations, i.e. the row ids of the test set.
truth (numeric())
  True (observed) response.
pdf (numeric())
  Numeric vector of estimated probability density function, evaluated at 'target' column of test set. One element for each observation in the test set.

cdf (numeric())
  Numeric vector of estimated cumulative distribution function, evaluated at 'target' column of test set. One element for each observation in the test set.

Method clone(): The objects of this class are cloneable with this method.

Usage:
PredictionDens$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

See Also
Other Prediction: PredictionSurv

Examples
library(mlr3)
task = mlr_tasks$get("precip")
learner = mlr_learners$get("dens.hist")
p = learner$train(task)$predict(task)
head(as.data.table(p))
PredictionSurv

- lp (numeric())
  Access the stored predicted linear predictor.
- response (numeric())
  Access the stored predicted survival time.
- missing (integer())
  Returns row_ids for which the predictions are missing or incomplete.

Methods

Public methods:
- PredictionSurv$new()
- PredictionSurv$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
PredictionSurv$new(
  task = NULL,
  row_ids = task$row_ids,
  truth = task$truth(),
  crank = NULL,
  distr = NULL,
  lp = NULL,
  response = NULL
)

Arguments:
- task (TaskSurv)
  Task, used to extract defaults for row_ids and truth.
- row_ids (integer())
  Row ids of the predicted observations, i.e. the row ids of the test set.
- truth (numeric())
  True (observed) response.
- crank (numeric())
  Numeric vector of predicted continuous rankings (or relative risks). One element for each observation in the test set. For a pair of continuous ranks, a higher rank indicates that the observation is more likely to experience the event.
- distr (VectorDistribution)
  VectorDistribution from distr6. Each individual distribution in the vector represents the random variable ‘survival time’ for an individual observation.
- lp (numeric())
  Numeric vector of linear predictor scores. One element for each observation in the test set. \( lp = X\beta \) where \( X \) is a matrix of covariates and \( \beta \) is a vector of estimated coefficients.
- response (numeric())
  Numeric vector of predicted survival times. One element for each observation in the test set.

Method clone(): The objects of this class are cloneable with this method.
Usage:
PredictionSurv$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also
Other Prediction: PredictionDens

Examples

library(mlr3)
task = tgen("simsurv")$generate(20)
learner = mlr_learners$get("surv.rpart")
p = learner$train(task)$predict(task)
head(as.data.table(p))

TaskDens  Density Task

Description
This task specializes Task for density estimation problems. The target column is assumed to be
numeric. The task_type is set to "density".
Predefined tasks are stored in the dictionary mlr_tasks.

Super class
mlr3::Task -> TaskDens

Methods

Public methods:
• TaskDens$new()
• TaskDens$truth()
• TaskDens$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
TaskDens$new(id, backend, target)

Arguments:
id (character(1))
  Identifier for the new instance.
backend (DataBackend)
  Either a DataBackend, or any object which is convertible to a DataBackend with as_data_backend().
  E.g., a data.frame() will be converted to a DataBackendDataTable.
target (character(1))
   Name of the target column.

**Method**  truth(): Returns the target column for specified row_ids, this is unsupervised so should not be thought of as a 'true' prediction. Defaults to all rows with role "use".

*Usage:*
TaskDens$truth(rows = NULL)

*Arguments:*
rows integer()
   Row indices.

*Returns:* numeric().

**Method**  clone(): The objects of this class are cloneable with this method.

*Usage:*
TaskDens$clone(deep = FALSE)

*Arguments:*
deep Whether to make a deep clone.

**See Also**
Other Task: TaskSurv

**Examples**
```r
task = TaskDens$new("precip", backend = data.frame(target = precip), target = "target")
task$task_type
task$truth()
```

---

**Description**

This task specializes mlr3::Task and mlr3::TaskSupervised for possibly-censored survival problems. The target is comprised of survival times and an event indicator. Predefined tasks are stored in mlr3::mlr_tasks. The task_type is set to "surv".

**Super classes**

mlr3::Task -> mlr3::TaskSupervised -> TaskSurv

**Active bindings**

censtype character(1)
   Returns the type of censoring, one of "right", "left", "counting", "interval", "interval2" or "mstate".

---

**TaskSurv**

*Survival Task*

---

**Description**

This task specializes mlr3::Task and mlr3::TaskSupervised for possibly-censored survival problems. The target is comprised of survival times and an event indicator. Predefined tasks are stored in mlr3::mlr_tasks. The task_type is set to "surv".

**Super classes**

mlr3::Task -> mlr3::TaskSupervised -> TaskSurv

**Active bindings**

censtype character(1)
   Returns the type of censoring, one of "right", "left", "counting", "interval", "interval2" or "mstate".

---
Methods

Public methods:

• TaskSurv$new()
• TaskSurv$truth()
• TaskSurv$formula()
• TaskSurv$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
TaskSurv$new(
  id,
  backend,
  time,
  event,
  time2,
  type = c("right", "left", "counting", "interval", "interval2", "mstate")
)

Arguments:

id (character(1))
  Identifier for the new instance.

backend (DataBackend)
  Either a DataBackend, or any object which is convertible to a DataBackend with as_data_backend().
  E.g., a data.frame() will be converted to a DataBackendDataTable.

time (character(1))
  Name of the column for event time if data is right censored, otherwise starting time if interval censored.

event (character(1))
  Name of the column giving the event indicator. If data is right censored then "0"/FALSE
  means alive (no event), "1"/TRUE means dead (event). If type is "interval" then "0"
  means right censored, "1" means dead (event), "2" means left censored, and "3" means
  interval censored. If type is "interval2" then event is ignored.

time2 (character(1))
  Name of the column for ending time for interval censored data, otherwise ignored.

type (character(1))
  Name of the column giving the type of censoring. Default is 'right' censoring.

Method truth(): True response for specified row_ids. Format depends on the task type. Defaults to all rows with role "use".

Usage:
TaskSurv$truth(rows = NULL)

Arguments:

rows integer()
  Row indices.

Returns: numeric().
Method `formula()`: Creates a formula for survival models with `survival::Surv` on the LHS.

Usage:
TaskSurv$formula(rhs = NULL)

Arguments:
`rhs` If `NULL` RHS is `..`, otherwise gives RHS of formula.

Returns: numeric().

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
TaskSurv$clone(deep = FALSE)

Arguments:
`deep` Whether to make a deep clone.

See Also
Other Task: `TaskDens`

Examples

```r
library(mlr3)
lung = mlr3misc::load_dataset("lung", package = "survival")
lung$status = (lung$status == 2L)
b = as_data_backend(lung)
task = TaskSurv$new("lung", backend = b, time = "time",
    event = "status")

task$target_names
task$feature_names
task$formula()
task$truth()
```
Index

crankcompositor, 5, 110
DataBackend, 118, 120
DataBackendDataTable, 118, 120
dens.logloss, 52
Dictionary, 30, 43, 103, 104, 106
dictionary, 9–11, 13–20, 24, 25, 27, 29, 30, 32, 34, 36, 37, 39, 41, 42, 44, 46, 47, 49, 50, 53, 55, 57, 59, 61, 62, 64, 65, 69, 71, 73, 74, 76–78, 80, 81, 83–85, 87, 89, 91, 93, 95, 96, 100, 104, 105, 108, 111, 118
distr6::Distribution, 9–12, 14–20
distr6::distrSimulate(), 104
distr6::mean.Distribution, 108
distr6::median.Distribution, 108
distr6::mode, 108
distrcompositor, 6, 113
faithful, 101
flexsurv::flexsurvspline(), 28
gbm::gbm(), 32
gbm::predict.gbm(), 32
glmnet::cv.glmnet(), 27
glmnet::glmnet(), 36
glmnet::predict.cv.glmnet(), 27
glmnet::predict.glmnet(), 36
graphics::hist(), 9
gss::ssden(), 20
kerdiest::kde(), 11
ks::kde(), 12
Learner, 7–11, 13–20, 22–25, 27, 29, 30, 32, 34, 36, 37, 39, 41, 42, 44, 46, 47, 49, 51, 54, 55, 68
LearnerDens, 7, 23, 115
LearnerDensHistogram, 9, 11–13, 15–21
LearnerDensKDE, 10, 10, 12, 13, 15–21
LearnerDensKDEkd, 10, 11, 13, 15–21
LearnerDensKDEks, 10–12, 13, 15–21
LearnerDensLocfit, 10–13, 14, 16–21
LearnerDensLogspline, 10–13, 15, 17–21
LearnerDensMixed, 10–13, 15, 16, 18–21
LearnerDensNonparametric, 10–13, 15–17, 19–21
LearnerDensPenalized, 10–13, 15–18, 20, 21
LearnerDensPlugin, 10–13, 15–19, 19, 21
LearnerDensSpline, 10–13, 15–20, 20
Learners, 30, 43
LearnerSurv, 5, 6, 8, 21, 106, 114, 116
LearnerSurvBlackboost, 23, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvCoxPH, 25, 25, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvCVGlmnet, 25–27, 27, 30, 31, 34–38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvFlexible, 25, 26, 28, 28, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvGamboost, 25, 26, 28, 30, 30, 34, 35, 37–40, 42, 43, 45, 47, 49, 50
LearnerSurvGBM, 25, 26, 28, 30, 31, 32, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvGlmboost, 25, 26, 28, 30, 31, 34, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvGlmnet, 25–28, 30, 31, 34–36, 36, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvKaplan, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvMboost, 25, 26, 28, 30, 31, 34, 35, 37, 38, 39, 42, 43, 45, 47, 49, 50
LearnerSurvObliqueRSF, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 40, 43, 45, 47, 49, 50
LearnerSurvPenalized, 25, 26, 28, 30, 31,
INDEX

34, 35, 37, 38, 40, 42, 45, 47, 49, 50
LearnerSurvRandomForestSRC, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 44, 47, 49, 50
LearnerSurvRanger, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 46, 49, 50
LearnerSurvRpart, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 50
LearnerSurvXgboost, 25, 26, 28, 30, 31, 34, 35, 37, 38, 40, 42, 43, 45, 47, 49, 49
locfit::density.lf(), 14
logspline::logspline(), 15
lrn(), 9–11, 13–20, 24, 25, 27, 29, 30, 32, 34, 36, 37, 39, 41, 42, 44, 46, 47, 49
lung, 102
mboost::blackboost(), 23
mboost::Family(), 24, 30, 34, 39
mboost::gamboost(), 30
mboost::glmboost(), 34
mboost::mboost(), 38
mboost::predict.mboost(), 23, 30, 34, 38
mboost::survFit(), 23, 30, 34, 38
mean(), 51, 54
Measure, 50, 53, 55, 57, 59, 61, 62, 64, 65, 69, 71, 73, 74, 76–78, 80, 81, 83–85, 87, 89, 91, 93, 95, 96, 98, 100
MeasureDens, 50, 54
MeasureDensLogloss, 52
MeasureSurv, 52, 53
MeasureSurvAUC, 54
MeasureSurvBeggC, 55, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvBrier (MeasureSurvGraf), 60
MeasureSurvChamblessAUC, 56, 57, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvGonenC, 56, 58, 58, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvGravf, 56, 58, 60, 62–66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvGravfSE, 56, 58, 60, 62, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvHarrellIC, 56, 58, 60, 62, 63, 64, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvHungAUC, 56, 58, 60, 62, 63, 65, 66, 67, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvIntegrated, 67
MeasureSurvIntLogloss, 56, 58, 60, 62–66, 68, 71, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvLogloss, 56, 58, 60, 62–66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvLoglossSE, 56, 58, 60, 62–66, 70, 72, 74, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvMAE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvMAESE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvMSE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvMSESE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvNagelkR2, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvQuigleyR2, 56, 58, 60, 62, 63, 65, 66, 67, 70, 72, 74, 75, 77–79, 81, 82, 82, 85, 86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvRMSE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–84, 85, 86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvRMSESE, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvSongAUC, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 87, 90, 92, 94, 96, 97, 99, 101
MeasureSurvUnoAUC, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 89, 92, 94, 96, 97, 99, 101
MeasureSurvUnoTPR, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvUnoAUC, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvUnoC, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvUnoTPR, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
MeasureSurvXuR2, 56, 58, 60, 62, 63, 65, 66, 70, 72, 74, 75, 77–79, 81, 82, 84–86, 88, 90, 92, 94, 96, 97, 99, 101
mlr3::Learner, 7, 9, 10, 12–18, 20–22, 24, 26, 27, 29, 31, 32, 34, 36, 38, 39, 41, 42, 44, 46, 48, 49
mlr3::Measure, 51–54, 56, 57, 59, 61, 63, 64, 66, 67, 69, 71, 73, 75–77, 79–81, 83, 84, 86, 87, 89, 91, 93, 95, 96, 98, 100
mlr3::mlr_learners, 30, 43
mlr3::mlr_measures, 50, 53
mlr3::mlr_task_generators, 104, 106
mlr3::mlr_tasks, 103, 119
mlr3::Prediction, 115, 116
mlr3::Task, 118, 119
mlr3::TaskGenerator, 104, 105
mlr3::TaskSupervised, 119
mlr3pipelines::GraphLearner, 6, 7
mlr3pipelines::mlr_pipeops, 108, 111
mlr3pipelines::PipeOp, 109, 110, 112, 113
mlr3pipelines::po(), 108, 111
mlr3proba::mlr3proba-package, 4
mlr3proba-package, 4
mlr3proba::LearnerDens, 9, 10, 12–18, 20, 21
mlr3proba::LearnerSurv, 24, 26, 27, 29, 31, 32, 34, 36, 38, 39, 41, 42, 44, 46, 48, 49
mlr3proba::MeasureDens, 52
mlr3proba::MeasureSurv, 54, 56, 57, 59, 61, 63, 64, 66, 67, 69, 71, 73, 75–77, 79–81, 83, 84, 86, 87, 89, 91, 93, 95, 96, 98, 100
mlr3proba::MeasureSurvAUC, 57, 66, 87, 89, 91, 93, 95, 96, 98
mlr3proba::MeasureSurvIntegrated, 54, 57, 61, 63, 66, 69, 71, 87, 89, 91, 93, 96, 98
mlr3proba::MeasureSurvLogloss, 75
mlr_learners, 9–11, 13–20, 24, 25, 27, 29, 30, 32, 34, 36, 37, 39, 41, 42, 44, 46, 47, 49
mlr_learnersdens.hist (LearnerDensHistogram), 9
mlr_learnersdens.kde (LearnerDensKDE), 10
mlr_learnersdens.kdekde (LearnerDensKDEkde), 11
mlr_learnersdens.kdekks (LearnerDensKDEKks), 12
mlr_learnersdens.locfit (LearnerDensLocfit), 14
mlr_learnersdens.logsppline (LearnerDensLogsppline), 15
mlr_learnersdens.mixed (LearnerDensMixed), 16
mlr_learnersdens.nonpar (LearnerDensNonparametric), 17
mlr_learnersdens.pen (LearnerDensPenalized), 18
mlr_learnersdens.plug (LearnerDensPlugin), 19
mlr_learnersdens.spline (LearnerDensSpline), 20
mlr_learnersurv.blackboost (LearnerSurvBlackboost), 23
mlr_learnersurv.coxph (LearnerSurvCoxPH), 25
mlr_learnersurv.cvglmnet (LearnerSurvCVGLMnet), 27
mlr_learnersurv.flexible (LearnerSurvFlexible), 28
mlr_learnersurv.gamboost (LearnerSurvGamboost), 30
mlr_learnersurv.gbm (LearnerSurvGBM), 32
mlr_learnersurv.glmboost (LearnerSurvGLMboost), 34
<table>
<thead>
<tr>
<th>mlr_learners_surv glmnet</th>
<th>LearnerSurvGlmnet, 36</th>
</tr>
</thead>
<tbody>
<tr>
<td>mlr_learners_surv kaplan</td>
<td>LearnerSurvKaplan, 37</td>
</tr>
<tr>
<td>mlr_learners_surv mboost</td>
<td>LearnerSurvMboost, 38</td>
</tr>
<tr>
<td>mlr_learners_surv obliqueRSF</td>
<td>LearnerSurvObliqueRSF, 40</td>
</tr>
<tr>
<td>mlr_learners_surv penalized</td>
<td>LearnerSurvPenalized, 42</td>
</tr>
<tr>
<td>mlr_learners_surv randomForestSRC</td>
<td>LearnerSurvRandomForestSRC, 44</td>
</tr>
<tr>
<td>mlr_learners_surv ranger</td>
<td>LearnerSurvRanger, 46</td>
</tr>
<tr>
<td>mlr_learners_surv rpart</td>
<td>LearnerSurvRpart, 47</td>
</tr>
<tr>
<td>mlr_learners_surv xgboost</td>
<td>LearnerSurvXgboost, 49</td>
</tr>
<tr>
<td>mlr_measures dens logloss</td>
<td>MeasureDensLogloss, 52</td>
</tr>
<tr>
<td>mlr_measures surv beggC</td>
<td>MeasureSurvBeggC, 55</td>
</tr>
<tr>
<td>mlr_measures surv brier</td>
<td>MeasureSurvBrier, 60</td>
</tr>
<tr>
<td>mlr_measures surv chambleSSauc</td>
<td>MeasureSurvChamblessAUC, 57</td>
</tr>
<tr>
<td>mlr_measures surv gonenC</td>
<td>MeasureSurvGonenC, 58</td>
</tr>
<tr>
<td>mlr_measures surv graf</td>
<td>MeasureSurvGraf, 60</td>
</tr>
<tr>
<td>mlr_measures surv grafSE</td>
<td>MeasureSurvGrafSE, 62</td>
</tr>
<tr>
<td>mlr_measures surv harrell1c</td>
<td>MeasureSurvHarrell1C, 64</td>
</tr>
<tr>
<td>mlr_measures surv hungAUC</td>
<td>MeasureSurvHungAUC, 65</td>
</tr>
<tr>
<td>mlr_measures surv intlogloss</td>
<td>MeasureSurvIntLogloss, 68</td>
</tr>
<tr>
<td>mlr_measures surv intloglossSE</td>
<td>MeasureSurvIntLoglossSE, 71</td>
</tr>
<tr>
<td>mlr_measures surv logloss</td>
<td>MeasureSurvLogloss, 73</td>
</tr>
<tr>
<td>mlr_measures surv loglossSE</td>
<td>MeasureSurvLoglossSE, 74</td>
</tr>
<tr>
<td>mlr_measures surv mae</td>
<td>MeasureSurvMAE, 76</td>
</tr>
<tr>
<td>mlr_measures surv maeSE</td>
<td>MeasureSurvMAESE, 77</td>
</tr>
<tr>
<td>mlr_measures surv mse</td>
<td>MeasureSurvMSE, 78</td>
</tr>
<tr>
<td>mlr_measures surv mseSE</td>
<td>MeasureSurvMSESE, 80</td>
</tr>
<tr>
<td>mlr_measures surv nagelkR2</td>
<td>MeasureSurvNagelkR2, 81</td>
</tr>
<tr>
<td>mlr_measures surv oquigleyR2</td>
<td>MeasureSurvOQuigleyR2, 82</td>
</tr>
<tr>
<td>mlr_measures surv rmse</td>
<td>MeasureSurvRMSE, 84</td>
</tr>
<tr>
<td>mlr_measures surv rmseSE</td>
<td>MeasureSurvRMSESE, 85</td>
</tr>
<tr>
<td>mlr_measures surv songAUC</td>
<td>MeasureSurvSongAUC, 87</td>
</tr>
<tr>
<td>mlr_measures surv songTNR</td>
<td>MeasureSurvSongTNR, 89</td>
</tr>
<tr>
<td>mlr_measures surv songTPR</td>
<td>MeasureSurvSongTPR, 91</td>
</tr>
<tr>
<td>mlr_measures surv unoAUC</td>
<td>MeasureSurvUnoAUC, 93</td>
</tr>
<tr>
<td>mlr_measures surv unoC</td>
<td>MeasureSurvUnoC, 94</td>
</tr>
<tr>
<td>mlr_measures surv unoTNR</td>
<td>MeasureSurvUnoTNR, 96</td>
</tr>
<tr>
<td>mlr_measures surv unoTPR</td>
<td>MeasureSurvUnoTPR, 98</td>
</tr>
<tr>
<td>mlr_pipeops crankcompose</td>
<td>PipeOpCrankCompositor, 108</td>
</tr>
<tr>
<td>mlr_pipeops distrcompose</td>
<td>PipeOpDistrCompositor, 110</td>
</tr>
<tr>
<td>mlr_reflections$learner_predict_types, 8, 22, 51, 54, 68</td>
<td></td>
</tr>
<tr>
<td>mlr_reflections$learner_properties, 8, 22</td>
<td></td>
</tr>
<tr>
<td>mlr_reflections$measure_properties, 51, 54, 55, 68</td>
<td></td>
</tr>
<tr>
<td>mlr_reflections$task_feature_types, 8, 22</td>
<td></td>
</tr>
<tr>
<td>mlr_task_generators, 104, 105</td>
<td></td>
</tr>
<tr>
<td>mlr_task_generators_simdens, 104</td>
<td></td>
</tr>
<tr>
<td>mlr_task_generators_simsurv, 105</td>
<td></td>
</tr>
<tr>
<td>mlr_tasks, 118</td>
<td></td>
</tr>
</tbody>
</table>
mlr_tasks_faithful, 101
mlr_tasks_lung, 102
mlr_tasks_precip, 102
mlr_tasks_rats, 103
mlr_tasks_unemployment, 103
msr(), 55, 57, 61, 62, 64, 65, 69, 71, 73,
74, 76–78, 80, 81, 83–85, 87, 89, 91,
93, 95, 96, 98, 100

np::npudens(), 16

obliqueRSF::ORSF(), 40
obliqueRSF::predict.orsf(), 40

paradox::ParamSet, 8, 22
pec::pecc(), 60
pecs, 106
penalized::penalized(), 42
penalized::predict(), 42
pendensity::pendensity(), 18
PipeOp, 108, 109, 111, 112
PipeOpCrankCompositor, 5, 108, 108, 113
PipeOpDistrCompositor, 6, 7, 110, 110, 111
plot.LearnerSurv, 114
plugdensity::plugin.density(), 19
precip, 102
Prediction, 7, 22
PredictionDens, 7, 115, 118
PredictionSurv, 22, 106–108, 110, 111, 116,
116

R6, 7, 9, 11–17, 19–22, 24, 26, 27, 29, 31, 33,
35, 36, 38, 39, 41, 43, 45, 46, 48,
50–54, 56, 57, 59, 61, 63, 64, 66, 69,
72, 73, 75, 76, 78–80, 82, 83, 85–87,
90, 92, 93, 95, 97, 99, 100, 104, 105,
109, 112, 115, 117, 118, 120

R6::R6Class, 101–103
randomForestSRC::predict.rfsrc(), 44
randomForestSRC::rfsrc(), 44
ranger::predict.ranger(), 46
ranger::ranger(), 46
rats, 103
requireNamespace(), 8, 23, 51, 54, 68
Resampling, 51, 54, 55, 68
rpart::predict.rpart(), 47
rpart::rpart(), 47

simsurv::simsurv(), 105

sm::sm.density(), 17
surv.harrellC, 54
surv.kaplan, 110
survAUC::AUC.cd(), 57
survAUC::AUC.hc(), 65
survAUC::AUC.sh(), 87
survAUC::AUC.uno(), 93
survAUC::BeggC(), 55
survAUC::GHCI(), 58
survAUC::Nagelk(), 81
survAUC::OXS(), 82
survAUC::sens.sh(), 91
survAUC::sens.uno(), 98
survAUC::spec.sh(), 89
survAUC::spec.uno(), 96
survAUC::UnOC(), 94
survAUC::X0(), 100
survival::coxph(), 25
survival::predict.coxph(), 25
survival::Surv, 121
survival::survfit(), 37
survival::survfit.coxph(), 25

Task, 51, 54, 55, 68, 118
TaskDens, 101, 102, 118, 121
TaskGenerator, 104, 105
TaskGenerators, 104, 106
TaskGeneratorSimdens
(mlr_task_generators_simdens), 104
TaskGeneratorSimsurv
(mlr_task_generators_simsurv), 105

Tasks, 103
TaskSurv, 102, 103, 107, 114, 115, 117, 119,
119
tgen(), 104, 105

VectorDistribution, 116, 117

xgboost::predict.xgb.Booster(), 49
xgboost::xgb.importance(), 50
xgboost::xgb.train(), 49
xgboost::xgboost(), 49