Package: jrrosell (via r-universe)

September 13, 2024

Title Personal R package for Jordi Rosell

Version 0.0.0.9006

Description Useful functions for personal usage.

License CC0

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.3.0)

URL https://jrosell.github.io/jrrosell,
https://github.com/jrosell/jrrosell

Suggests beepr, blastula, devtools, doFuture, doParallel, dplyr, extrafont, future, ggplot2, glmnet, glue, hardhat, here, httr, httr2, janitor, modeldata, openxlsx, pak, parallel, parallelly, parsnip, pkgdown, purrr, readr, recipes, sf, styler, testthat, tibble, tictoc, tidymodels, tune, usethis, workflows, xgboost

Imports rlang

LazyData true

Repository https://jrosell.r-universe.dev

RemoteUrl https://github.com/jrosell/jrrosell

RemoteRef HEAD

RemoteSha 12abce5754794403ef6959277733f380f5d27a7c

Contents

`as.bitstring` *Data type utilities*

Description

Get the bit representation of a double number

Usage

```
as.bitstring(x)
```

Arguments

`x`	A numeric vetor.

Details

Get the bit representation of a double number Using rev() ensures that the bit order is correct, and the binary representation aligns with the usual convention of having the MSB first and the LSB last. This is because numToBits() returns the bits in the reverse order, and without rev(), we end up with the LSB first and the MSB last.

Source

https://youtu.be/J4DnzjIFj8w

Examples

```
0.1 + 0.2 == 0.3
as.bitstring(0.1 + 0.2)
as.bitstring(0.3)
```

`aside` *Multiple aside functions with base R pipe*

Description

Multiple aside functions with base R pipe

Usage

```
aside(x, ...)
```

Arguments

`x`	An object
`...`	functions to run aside

Examples

```
n_try <- 1
rnorm(200) |>
  matrix(ncol = 2) |>
  aside(
    print("Matrix prepared"),
    print(n_try)
  ) |>
  colSums()
```

`calc_split_prop` *Calculate split proportion*

Description

From a data frame, it returns the split proportion.

Usage

```
calc_split_prop(df)
```

Arguments

`df`	A data frame

Details

The calc_validation_size function returns the optimal split proportion according to the number of rows for your validation set.

Source

https://stats.stackexchange.com/a/305063/7387

Examples

```
calc_split_prop(data.frame(row = 1:891))
```

`calc_validation_size` *Calculate validation size*

Description

From and expected p and desired std_err, it returns the minimal validation size for your assesment sets or validation sets.

Usage

```
calc_validation_size(expected_p, desired_std_err)
```

Arguments

`expected_p`	An object
`desired_std_err`	An expresion

Details

The calc_validation_size function returns the minimal validation size for expected probabilities and desired error.

Source

https://stats.stackexchange.com/a/304996/7387

Examples

```
calc_validation_size(expected_p = 0.8, desired_std_err = 0.02)
```

check_installed_gihub

Check if the last github version is installed

Description

Get the version from the DESCRIPTION file of the master branch in the github package.

Usage

```
check_installed_gihub(repo)
```

Arguments

`repo`	a github repo/package. Ex: check_installed_gihub("tidyverse/dplyr")

last_metrics

Do the last fit and get the metrics

Description

Do the last fit and get the metrics

Usage

```
last_metrics(res, split, metric)
```

Arguments

`res`	Tune results
`split`	The initial split object
`metric`	What metric to use to select the best workflow

Examples

```
library(tidymodels)
library(xgboost)
library(modeldata)
data(cells)
split <- cells |>
  mutate(across(where(is.character), as.factor)) |>
  sample_n(500) |>
  initial_split(strata = class)
train <- training(split)
folds <- vfold_cv(train, v = 2, strata = class)
wf <- train |>
  recipe(case ~ .) |>
```

```
    step_integer(all_nominal_predictors()) |>
    workflow_boost_tree()
  res <- wf |>
    tune_grid(
      folds,
      grid = 2,
      metrics = metric_set(roc_auc),
      control = control_grid(save_workflow = TRUE, verbose = FALSE)
    )
  res |> collect_metrics()
  res |> last_metrics(split, "roc_auc")
  best <- res |> fit_best()
  best |>
    augment(testing(split)) |>
    roc_auc(case, .pred_Test) |>
    pull(.estimate)
```

`mapParallel` *Map parallel processing*

Description

Map data over a function in one parallel core

Usage

```
mapParallel(x, fun_list)
```

Arguments

`x`	An object
`fun_list`	A list of functions to run in parallel over the object

Details

The mapParallel function uses parallel package to run functions to run in parallel over the object. It doesn't returning anything.

See Also

https://github.com/jrosell/jrrosell/blob/main/R/parallel.R

Examples

```
data.frame(x = 2) |> mapParallel(list(function(x) print(x), function(x) print(x * 2)))
```

`notify_finished` *Make a sound and send an email when a process finished*

Description

The notify_finished make a sound using beepr::beep, compose and email and send it returing the blastula::smtp_send call results.

Usage

```
notify_finished(name, body = "", ..., sound = 1, tictoc_result = NULL)
```

Arguments

`name`	The process name (Required)
`body`	The contents of the email (Default "")
`...`	Additional arguments to pass to the template function. If you're using the default template, you can use font_family to control the base font, and content_width to control the width of the main content; see blastula_template(). By default, the content_width is set to 1000px. Using widths less than 600px is generally not advised but, if necessary, be sure to test such HTML emails with a wide range of email clients before sending to the intended recipients. The Outlook mail client (Windows, Desktop) does not respect content_width.
`sound`	The sound for beepr::beep call (Default 1)
`tictoc_result`	the result from tictoc::toc (Default NULL)

Details

The following environment variables should be set:

- MY_SMTP_USER from
- MY_SMTP_RECIPIENT to
- MY_SMTP_PASSWORD service password (for gmail you can use https://myaccount.google.com/apppasswo
- MY_SMTP_PROVIDER blastula provider (gmail if not set)

Examples

```
if (exists("not_run")) {
  tictoc::tic()
  Sys.sleep(1)
  jrrosell::notify_finished("job", "Well done", sound = "fanfare", tictoc_result = tictoc::toc())
}
```

`package_github_name` *Github name of the package*

Description

Get the name of the package from the DESCRIPTION file of the master branch in the github repo

Usage

```
package_github_name(x, file_lines = NULL)
```

Arguments

`x`	a single repo/package to check Ex: package_github_name("tidyverse/dplyr")
`file_lines`	(default = NULL, internal)

`package_github_version` *Github version of the package*

Description

Get the version from the DESCRIPTION file of the master branch in the github repo

Usage

```
package_github_version(x, file_lines = NULL)
```

Arguments

`x`	a single repo/package to check Ex: package_github_version("tidyverse/dplyr")
`file_lines`	(default = NULL, internal)

prep_juice — *Prep, juice and glimpse a recipe or workflow*

Description

Prep, juice and glimpse a recipe or workflow

Usage

```
prep_juice(object)
```

Arguments

`object`	A recipe or a workflow object with a recipe

Source

https://recipes.tidymodels.org/reference/update.step.html

Examples

```
recipes::recipe(spray ~ ., data = InsectSprays) |>
  prep_juice()
recipes::recipe(spray ~ ., data = InsectSprays) |>
  workflows::workflow(parsnip::linear_reg()) |>
  prep_juice()
```

prep_juice_cols — *Prep, juice and get cols from a recipe or workflow*

Description

Prep, juice and get cols from a recipe or workflow

Usage

```
prep_juice_cols(object)
```

Arguments

`object`	A recipe or a workflow object with a recipe

Examples

```
recipes::recipe(spray ~ ., data = InsectSprays) |>
  prep_juice_cols()
```

`read_chr` *Read character columns with clean names*

Description

It's useful for reading the most common types of flat file data, comma separated values and tab separated values.

Usage

```
read_chr(file, delim = ",", locale, ...)
```

Arguments

`file`	Either a path to a file, a connection, or literal data (either a single string or a raw vector).
`delim`	Single character used to separate fields within a record.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use locale() to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`...`	Other parameters to readr::read_delim.

Details

The read_chr function works like `readr::read_delim`, except that column sreturned would be characters and with clean names. It requires readr and janitor packages installed.

Examples

```
es <- readr::locale("es", tz = "Europe/Madrid", decimal_mark = ",", grouping_mark = ".")
read_chr(readr::readr_example("mtcars.csv"), delim = ",", locale = es)
```

`read_url` *Read the html text of an url*

Description

It's useful for getting the text for webpages in a single character vector.

Usage

```
read_url(url, sleep = 0)
```

Arguments

`url`	Full url including http or https protocol and the page path.
`sleep`	Seconds to sleep after the request is done and before returning the result.

Details

The read_url function works uses rvest::read_html and purr::possibly and it's fault tolearnt.

Examples

```
read_url("https://www.google.cat/", sleep = 1)
```

`read_xlsx` *Read a sheet from a xlsx file into a tibbles*

Description

It's useful for reading a single sheets from a Excel/Openoffice file.

Usage

```
read_xlsx(xlsxFile, ..., sheet = 1, startRow = 1)
```

Arguments

`xlsxFile`	The name of the file.
`...`	Other parameters to openxls::read.xlsx function
`sheet`	The name or index of the sheet (default 1)
`startRow`	The number of the starting reading row (default 1)

Details

The write_xlsx it's a wroapper for `openxls::write.xlsx`.

Examples

```
l <- list("IRIS" = iris, "MTCATS" = mtcars, matrix(runif(1000), ncol = 5))
write_xlsx(l, "/tmp/writeList1.xlsx", colWidths = c(NA, "auto", "auto"))
read_xlsx("/tmp/writeList1.xlsx")
file.remove("/tmp/writeList1.xlsx")
```

`spain_ccaas` *spain_ccaas*

Description

spain_ccaas

Usage

```
spain_ccaas
```

Format

`spain_ccaas`:
A sf object with 19 rows and 4 columns:

OBJECTID

codigo

nombre

geometry

Source

https://github.com/koldLight/curso-r-dataviz/blob/master/dat/spain_ccaas.geojson

Examples

```
library(sf)
data(spain_ccaas)
head(spain_ccaas)
```

`spain_provinces` *spain_provinces*

Description

spain_provinces

Usage

```
spain_provinces
```

Format

`spain_provinces`:
A sf object with 60 rows and 4 columns:

OBJECTID

codigo

nombre

geometry

Source

https://github.com/koldLight/curso-r-dataviz/blob/master/dat/spain_provinces.geojson

Examples

```
library(sf)
data(spain_provinces)
head(spain_provinces)
```

`startParallel` *Start parallel processing*

Description

Start timer and parallel processing in max/min of the available cores.

Usage

```
startParallel(msg = NULL, max = 10, min = 1)
```

Arguments

`msg`	A character vector with the tictoc timer message.
`max`	An integer with the max desired cores.
`min`	An integer with the min desired cores.

Details

The startParallel function uses tictoc, parallelly, parallel packages. It returns a parallel cluster to be passed as a stopParallel argument

Examples

```
cl <- startParallel("parallel processing", max = 3, min = 1)
print("parallel processing here")
stopParallel(cl)
```

`stopParallel` *Stop parallel processing*

Description

Stop started timer and parallel processing.

Usage

```
stopParallel(cl)
```

Arguments

`cl`	A cluster object returned by startParallel or parallel::makeCluster, parallel::makePSOCKcluster, parallel::makeForkCluster

Details

The stopParallel function uses tictoc and parallel packages. It returns the result of parallel::stopCluster(cl) method.

See Also

https://github.com/jrosell/jrrosell/blob/main/R/parallel.R

Examples

```
cl <- startParallel()
print("parallel processing here")
stopParallel(cl)
```

`tee` *Tee pipe that return the original value instead of the result*

Description

Pipe a value forward into a functio or call expression and return the original value instead of the result. This is useful when an expression is used for its side-effect, say plotting or printing.

Usage

```
tee(x, expr)
```

Arguments

`x`	An object
`expr`	An expresion

Details

The tee pipe works like `|>`, except the return value is `x` itself, and not the result of `expr` call.

Thanks

I want to give credit to Michael Milton and Matthew Kay for the idea and the code.

Source

https://mastodon.social/@multimeric@genomic.social/109555362766969210

Examples

```
rnorm(200) |>
  matrix(ncol = 2) |>
  as.data.frame() |>
  tee(\(x) {
    ggplot(x, aes(V1, V2)) +
      geom_point()
  }) |>
  colSums()
```

`theme_roboto` *Sets a minimal theme using the Roboto font family*

Description

It requires roboto fonts installed in your O.S. and run z

Usage

```
theme_roboto(
  base_size = 11,
  strip_text_size = 12,
  strip_text_margin = 5,
  subtitle_size = 13,
  subtitle_margin = 10,
  plot_title_size = 16,
  plot_title_margin = 10,
  ...
)
```

Arguments

`base_size`	= 11
`strip_text_size`	= 12
`strip_text_margin`	= 5
`subtitle_size`	= 13
`subtitle_margin`	= 10
`plot_title_size`	= 16
`plot_title_margin`	= 10
`...`	Other parameters passed to theme_set

`theme_set_roboto_darkblue` *Sets a dark blue colored dark minimal theme using the Roboto font family*

Description

Sets a dark blue colored dark minimal theme using the Roboto font family

Usage

```
theme_set_roboto_darkblue(...)
```

Arguments

`...`	Other parameters passed to theme_set

Examples

```
library(ggplot2)
theme_set_roboto_darkblue()
ggplot(iris, aes(Species)) +
  geom_bar()
```

`update_step` *Update recipe step values by id*

Description

Update the vaules of a specific recipe step located by id

Usage

```
update_step(object, target_id, ...)
```

Arguments

`object`	A recipe or a workflow object with a recipe
`target_id`	The id name of the step
`...`	The arguments to update the step.

Examples

```
recipes::recipe(spray ~ ., data = InsectSprays) |>
  recipes::step_ns(count, deg_free = hardhat::tune(), id = "ns") |>
  update_step("ns", deg_free = 1)
```

`workflow_boost_tree` *Create an xgboost tunable workflow for regression and classification*

Description

Create an xgboost tunable workflow for regression and classification

Usage

```
workflow_boost_tree(rec, engine = "xgboost", counts = TRUE, ...)
```

Arguments

`rec`	prerocessing recipe to build the workflow
`engine`	optional, xgboost by default
`counts`	Optional logic argument wether mtry use counts or not
`...`	optional engine arguments

Examples

```
library(tidymodels)
library(xgboost)
library(modeldata)
library(future)
data(cells)
split <- cells |>
  mutate(across(where(is.character), as.factor)) |>
  sample_n(500) |>
  initial_split(strata = class)
train <- training(split)
folds <- vfold_cv(train, v = 2, strata = class)
wf <- train |>
  recipe(case ~ .) |>
  step_integer(all_nominal_predictors()) |>
  workflow_boost_tree()
doFuture::registerDoFuture()
plan(sequential)
res <- wf |>
  tune_grid(
    folds,
    grid = 2,
    metrics = metric_set(roc_auc),
    control = control_grid(save_workflow = TRUE, verbose = FALSE)
  )
res |> collect_metrics()
res |> last_metrics(split, "roc_auc")
best <- res |> fit_best()
best |>
  augment(testing(split)) |>
  roc_auc(case, .pred_Test) |>
  pull(.estimate)
```

`workflow_elasticnet` *Create a tuneable glmnet worfklow for regression and classification*

Description

Create a tuneable glmnet worfklow for regression and classification

Usage

```
workflow_elasticnet(rec, engine = "glmnet", ...)
```

Arguments

`rec`	prerocessing recipe to build the workflow
`engine`	optional, glmnet by default
`...`	Optional engine arguments

Examples

```
library(tidymodels)
library(glmnet)
library(modeldata)
library(future)
data(cells)
split <- cells |>
  mutate(across(where(is.character), as.factor)) |>
  sample_n(500) |>
  initial_split(strata = class)
train <- training(split)
folds <- vfold_cv(train, v = 2, strata = class)
wf <- train |>
  recipe(case ~ .) |>
  step_integer(all_nominal_predictors()) |>
  workflow_elasticnet()
doFuture::registerDoFuture()
plan(sequential)
res <- wf |>
  tune_grid(
    folds,
    grid = 2,
    metrics = metric_set(roc_auc),
    control = control_grid(save_workflow = TRUE, verbose = FALSE)
  )
res |> collect_metrics()
res |> last_metrics(split, "roc_auc")
best <- res |> fit_best()
best |>
  augment(testing(split)) |>
  roc_auc(case, .pred_Test) |>
  pull(.estimate)
```

`write_xlsx` *Write a list of tibbles to a xlsx file*

Description

It's useful for saving multiple data to a multiple sheets of a single Excel/Openoffice/libreoffice file.

Usage

```
write_xlsx(data, distfile, ...)
```

Arguments

`data`	A named list of tibbles

`distfile`	The name of the destination file.
`...`	Other parameters to openxls::write.xlsx function

Details

The write_xlsx it's a wroapper for `openxls::write.xlsx`.

Examples

```
l <- list("IRIS" = iris, "MTCATS" = mtcars, matrix(runif(1000), ncol = 5))
write_xlsx(l, "/tmp/writeList1.xlsx", colWidths = c(NA, "auto", "auto"))
file.remove("/tmp/writeList1.xlsx")
```

Index