

Package: plnr (via r-universe)

September 24, 2026

Title A Framework for Planning and Executing Analyses

Version 2026.9.24

Description Use a plan to run the same analysis function over many sets of arguments, or to run different functions on the same data. A plan holds the data sources, the argument sets and the functions. Running the plan loads the data once and passes it to every analysis, in sequence or in parallel. Typical uses are one method applied across strata such as locations or age groups, or across exposures and outcomes, and the tables and figures of a report.

License MIT + file LICENSE

URL <https://www.rwhite.no/plnr/>, <https://github.com/raubreywhite/plnr>

BugReports <https://github.com/raubreywhite/plnr/issues>

Encoding UTF-8

Config/Needs/website raubreywhite/rwtemplate

LazyData true

Depends R (>= 4.1.0)

Imports data.table, digest, fs, foreach, glue, pbmcapply, progress, purrr, R6, stats, tidyr, usethis, utils, uuid

Suggests testthat, doParallel, parallel, knitr, rmarkdown, progressr, ggplot2, readxl

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake git make libgit2-dev libicu-dev libuv1-dev libssl-dev libx11-dev

Repository <https://raubreywhite.r-universe.dev>

Date/Publication 2026-09-24 11:16:37 UTC

RemoteUrl <https://github.com/raubreywhite/plnr>

RemoteRef HEAD

RemoteSha 152c44c8364577d950e4434af826628f344a9d91

Contents

create_rmarkdown	2
example_action_fn	3
example_data_fn_nor_covid19_cases_by_time_location	4
expand_list	5
get_anything	5
is_run_directly	6
nor_covid19_cases_by_time_location	7
Plan	8
set_opts	22
test_action_fn	23
try_again	24

Index	26
--------------	-----------

create_rmarkdown	<i>Create an example R Markdown project that uses plnr</i>
------------------	--

Description

create_rmarkdown() writes a small project to home. Its run.R builds a [Plan](#) with one dataset and two analyses, then renders a report that runs them.

Usage

```
create_rmarkdown(home)
```

Arguments

home Character string. The directory for the project.

Details

The project holds:

- run.R, which calls org::initialize_project(), builds the plan and renders the report.
- R/table_death.R and R/figure_death.R, the two action functions.
- paper/paper.Rmd, the report.
- The empty directories results and raw.

run.R needs the org, ggplot2, huxtable, lubridate and rmarkdown packages. plnr does not install them.

Value

The project path, invisibly, as usethis::create_project() returns it.

See Also

`vignette("plnr")` for the concepts that `run.R` uses.

Other utilities: [try_again\(\)](#)

Examples

```
home <- tempfile("plnr_example_")
create_rmarkdown(home)
list.files(home, recursive = TRUE)
unlink(home, recursive = TRUE)
```

example_action_fn	<i>Print what an action function receives</i>
-------------------	---

Description

`example_action_fn()` is an example action function. It prints the names of the datasets in `data` and the names of the elements in `argset`.

Usage

```
example_action_fn(data, argset)
```

Arguments

<code>data</code>	A named list of datasets, as a Plan passes it.
<code>argset</code>	A named list of arguments for one analysis.

Value

`names(argset)`, invisibly.

See Also

`vignette("plnr")` for what an action function is.

Other example and test functions: [example_data_fn_nor_covid19_cases_by_time_location\(\)](#), [test_action_fn\(\)](#)

Examples

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
```

```
      list(name = "analysis_2", var_1 = 2)
    )
  )
  p$run_one("analysis_1")
```

```
example_data_fn_nor_covid19_cases_by_time_location
```

Return the example Covid-19 dataset

Description

`example_data_fn_nor_covid19_cases_by_time_location()` is an example data function: it takes no arguments and returns one dataset. Give its name to `Plan$add_data()` as `fn_name`.

Usage

```
example_data_fn_nor_covid19_cases_by_time_location()
```

Value

[nor_covid19_cases_by_time_location](#), a `data.table`.

See Also

`vignette("adding_analyses")`, which writes its own data functions.

Other example and test functions: [example_action_fn\(\)](#), [test_action_fn\(\)](#)

Examples

```
dim(example_data_fn_nor_covid19_cases_by_time_location())

p <- plnr::Plan$new()
p$add_data(
  name = "covid19_cases",
  fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location"
)
names(p$get_data())
```

expand_list	<i>Make a list of argsets from every combination of values</i>
-------------	--

Description

`expand_list()` returns one named list for each combination of the values of its arguments. It works like `tidyr::expand_grid()`, but it returns a list of lists, which `Plan$add_argset_from_list()` and `Plan$add_analysis_from_list()` take.

Usage

```
expand_list(...)
```

Arguments

... Named vectors or lists. Each holds the values of one argument.

Value

A list of named lists, one for each combination. The first argument varies slowest.

See Also

`vignette("adding_analyses")`, which builds argsets with `expand_list()`.

Other plan helpers: [Plan](#), [get_anything\(\)](#), [is_run_directly\(\)](#), [set_opts\(\)](#)

Examples

```
argsets <- plnr::expand_list(location = c("oslo", "bergen"), age = c("0-14", "15+"))
str(argsets)
```

get_anything	<i>Find an object by name</i>
--------------	-------------------------------

Description

`get_anything()` returns the object that a name refers to. [Plan](#) uses it to find the function that `fn_name` names.

Usage

```
get_anything(x, envir = parent.frame(), mode = "any")
```

Arguments

<code>x</code>	Character string. The name of the object.
<code>envir</code>	The environment where step 2 starts. The default is the environment that calls <code>get_anything()</code> .
<code>mode</code>	The type of object to find, as in <code>base::get()</code> . <code>Plan</code> uses "function".

Details

`get_anything()` looks in this order and returns the first match:

1. For "pkg::name", the object that package pkg exports. It looks nowhere else.
2. The environments from `envir` up to, but not including, the first global environment or package namespace above it.
3. The global environment, then the search path.
4. The `plnr` namespace, then the packages that `plnr` imports.

Value

The object that `x` names. `get_anything()` stops with an error when no object matches.

See Also

`vignette("plnr")`, whose "Function names" section shows how a `Plan` uses `fn_name`.

Other plan helpers: `Plan`, `expand_list()`, `is_run_directly()`, `set_opts()`

Examples

```
# Get a namespace-qualified object
head(plnr::get_anything("plnr::nor_covid19_cases_by_time_location"))

# Get an object from the calling environment
f <- function() {
  x <- 1
  get_anything("x")
}
f()
```

`is_run_directly`

Check whether code runs at the top level of the console

Description

`is_run_directly()` returns TRUE when you call it at the top level of the R console or of an Rscript file. It returns FALSE inside a function, and inside `source()`, `eval()` and a knitr chunk.

Usage

```
is_run_directly()
```

Details

Put it at the start of an action function to load data and argset while you develop the function line by line. When the plan runs the function, it returns FALSE, so the lines do nothing.

Value

TRUE or FALSE.

See Also

`vignette("plnr")`, whose "Debugging" section shows the pattern.

Other plan helpers: [Plan](#), [expand_list\(\)](#), [get_anything\(\)](#), [set_opts\(\)](#)

Examples

```
f <- function() {
  is_run_directly()
}
f()
```

`nor_covid19_cases_by_time_location`

Covid-19 cases confirmed by PCR in Norway, by nation and county

Description

The data come from the Norwegian Surveillance System for Communicable Diseases (MSIS). `date` is the date of the PCR test. The data were extracted on 2022-05-04.

Usage

```
nor_covid19_cases_by_time_location
```

Format

A `data.table` with 11028 rows and 18 variables:

granularity_time "day" or "isoweek".

granularity_geo "nation" or "county".

country_iso3 "nor".

location_code "nation_nor", or one of 11 county codes such as "county_nor03".

border The year of the county borders, 2020.

age "total".
sex "total".
isoyear ISO year.
isoweek ISO week.
isoyearweek ISO year and week, such as "2020-08".
season Season, such as "2020/2021".
seasonweek Week within the season.
calyear Calendar year. NA in weekly rows.
calmonth Calendar month. NA in weekly rows.
calyearmonth Calendar year and month, such as "2020-M02". NA in weekly rows.
date The date. In weekly rows, the Sunday that ends the ISO week.
covid19_cases_testdate_n Number of confirmed cases.
covid19_cases_testdate_pr100000 Number of confirmed cases per 100 000 population.

Source

https://github.com/folkehelseinstituttet/surveillance_data/blob/master/covid19/_DOCUMENTATION_data_covid19_msis_by_time_location.txt

Plan

Plan and run analyses

Description

A Plan holds data sources and analyses, and runs the analyses on the data. An analysis is one argset plus one action function.

Details

An argset is a named list of arguments for one analysis. When an analysis runs, its argset also holds `index_analysis`: the position or name that ran it.

An action function **MUST** take the data as its first argument and the argset as its second. It **MAY** take further arguments. The run methods pass their ... to it in one of two ways:

- A function given as `fn` gets the argset by position, and every argument in ...
- A function given as `fn_name` gets data and argset by name. It gets ... only when it has more than two formal arguments.

`get_anything()` finds the function that `fn_name` names when the analysis runs. If the name resolved in a local environment when you set `fn_name`, the plan stores that environment and looks there first.

Value

`Plan$new()` returns a new Plan object.

Public fields

`analyses` A named list with one element for each analysis. Each element holds `argset` and, once set, `fn`, `fn_name` and `fn_env`.

Methods**Public methods:**

- `Plan$new()`
- `Plan$add_data()`
- `Plan$add_argset()`
- `Plan$add_argset_from_df()`
- `Plan$add_argset_from_list()`
- `Plan$add_analysis()`
- `Plan$add_analysis_from_df()`
- `Plan$add_analysis_from_list()`
- `Plan$apply_action_fn_to_all_argsets()`
- `Plan$apply_analysis_fn_to_all()`
- `Plan$x_length()`
- `Plan$x_seq_along()`
- `Plan$set_progress()`
- `Plan$set_progressor()`
- `Plan$set_verbose()`
- `Plan$set_use_foreach()`
- `Plan$get_data()`
- `Plan$get_analysis()`
- `Plan$get_argset()`
- `Plan$get_argsets_as_dt()`
- `Plan$run_one_with_data()`
- `Plan$run_one()`
- `Plan$run_all_with_data()`
- `Plan$run_all()`
- `Plan$run_all_progress()`
- `Plan$run_all_parallel()`
- `Plan$clone()`

`Plan$new()`: Create a new plan.

Usage:

```
Plan$new(verbose = interactive() | config$force_verbose, use_foreach = FALSE)
```

Arguments:

`verbose` Logical. TRUE makes `run_all()` show progress. The default is TRUE in an interactive session, or after `set_opts(force_verbose = TRUE)`.

`use_foreach` Logical or NULL. TRUE makes `run_all()` use `foreach`, and FALSE a loop. NULL uses `foreach` when a backend with more than one worker is registered and `progressr` is installed. The default is FALSE.

Returns: A new Plan object.

`Plan$add_data()`: Add a data source. `get_data()` loads every data source. Give one of `fn`, `fn_name` and `direct`. If you give more, `direct` wins over `fn_name`, and `fn_name` wins over `fn`.

Usage:

```
Plan$add_data(name, fn = NULL, fn_name = NULL, direct = NULL)
```

Arguments:

`name` Character string. The name of the dataset in the list that `get_data()` returns.

`fn` Optional. A function with no arguments that returns the dataset.

`fn_name` Optional. The name of such a function, as a character string. `"pkg::fn"` also works.

`direct` Optional. The dataset itself.

Returns: The stored data source, invisibly: a list that holds `fn`, `fn_name`, `direct`, `name` and `fn_env`.

Examples:

```
p <- plnr::Plan$new()
data_fn <- function() {
  return(plnr::nor_covid19_cases_by_time_location)
}
p$add_data("data_1", fn = data_fn)
p$add_data("data_2", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_data("data_3", direct = plnr::nor_covid19_cases_by_time_location)
names(p$get_data())
```

`Plan$add_argset()`: Add an argset: an analysis that has no action function yet. If an analysis with this name exists, `add_argset()` replaces its argset and keeps its action function.

Usage:

```
Plan$add_argset(name = uuid::UUIDgenerate(), ...)
```

Arguments:

`name` Character string. The name of the analysis. The default is a UUID.

`...` Named arguments. They make up the argset.

Returns: The argset, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_argset("argset_1", var_1 = 3, var_b = "hello")
p$add_argset("argset_2", var_1 = 8, var_c = "hello2")
p$get_argsets_as_dt()
```

`Plan$add_argset_from_df()`: Add one argset for each row of a data frame.

Usage:

```
Plan$add_argset_from_df(df)
```

Arguments:

`df` A data frame. Each row is one argset. A column name names the analyses.

Returns: NULL, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_argset_from_df(data.frame(name = c("a", "b"), var_1 = c(1, 2)))
p$get_argsets_as_dt()
```

`Plan$add_argset_from_list()`: Add one argset for each element of a list.

Usage:

```
Plan$add_argset_from_list(l)
```

Arguments:

`l` A list of named lists. Each inner list is one argset. An element name names the analysis.

Returns: NULL, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_argset_from_list(plnr::expand_list(var_1 = 1:2, var_2 = c("i", "j")))
p$get_argsets_as_dt()
```

`Plan$add_analysis()`: Add an analysis: one argset and one action function. If an analysis with this name exists, `add_analysis()` replaces it.

Usage:

```
Plan$add_analysis(name = uuid::UUIDgenerate(), fn = NULL, fn_name = NULL, ...)
```

Arguments:

`name` Character string. The name of the analysis. The default is a UUID.

`fn` Optional. The action function.

`fn_name` Optional. The name of the action function, as a character string. `"pkg::fn"` also works. Give `fn` or `fn_name`, not both.

`...` Named arguments. They make up the argset.

Returns: The argset, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis(name = "analysis_1", fn_name = "plnr::example_action_fn")
p$run_one("analysis_1")
```

`Plan$add_analysis_from_df()`: Add one analysis for each row of a data frame.

Usage:

```
Plan$add_analysis_from_df(fn = NULL, fn_name = NULL, df)
```

Arguments:

`fn` Optional. The action function of every analysis.

`fn_name` Optional. The name of the action function of every analysis. A column `fn_name` in `df` overrides it.

`df` A data frame. Each row is one argset. A column name names the analyses.

Returns: NULL, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_df(
  fn_name = "plnr::example_action_fn",
  df = data.frame(name = c("a", "b"), var_1 = c(1, 2))
)
p$run_one("a")
```

`Plan$add_analysis_from_list()`: Add one analysis for each element of a list.

Usage:

```
Plan$add_analysis_from_list(fn = NULL, fn_name = NULL, l)
```

Arguments:

`fn` Optional. The action function of every analysis.

`fn_name` Optional. The name of the action function of every analysis. An element `fn_name` in an inner list overrides it.

`l` A list of named lists. Each inner list is one argset. An element name names the analysis.

Returns: NULL, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$run_one("analysis_1")
```

`Plan$apply_action_fn_to_all_argsets()`: Give every analysis in the plan the same action function. It replaces any action function that an analysis already has.

Usage:

```
Plan$apply_action_fn_to_all_argsets(fn = NULL, fn_name = NULL)
```

Arguments:

`fn` Optional. The action function.

`fn_name` Optional. The name of the action function, as a character string.

Returns: NULL, invisibly.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_argset_from_list(list(
  list(name = "analysis_1", var_1 = 1),
  list(name = "analysis_2", var_1 = 2)
))
```

```

))
p$apply_action_fn_to_all_argsets(fn_name = "plnr::example_action_fn")
p$run_one("analysis_1")

```

`Plan$apply_analysis_fn_to_all()`: Deprecated. Use `apply_action_fn_to_all_argsets()`.

Usage:

```
Plan$apply_analysis_fn_to_all(fn = NULL, fn_name = NULL)
```

Arguments:

`fn` Optional. The action function.

`fn_name` Optional. The name of the action function.

Returns: NULL, invisibly.

`Plan$x_length()`: Count the analyses in the plan.

Usage:

```
Plan$x_length()
```

Returns: An integer.

`Plan$x_seq_along()`: Get the positions of the analyses, `seq_along(analyses)`.

Usage:

```
Plan$x_seq_along()
```

Returns: An integer vector.

`Plan$set_progress()`: Set a progress bar that `run_all()` ticks once for each analysis.

Usage:

```
Plan$set_progress(pb)
```

Arguments:

`pb` An object with a `tick()` method, such as one from `progress::progress_bar$new()`.

Returns: `pb`, invisibly.

`Plan$set_progressor()`: Set a progressr progressor that `run_all()` calls once for each analysis.

Usage:

```
Plan$set_progressor(pb)
```

Arguments:

`pb` A function from `progressr::progressor()`.

Returns: `pb`, invisibly.

`Plan$set_verbose()`: Set verbose, which `new()` describes.

Usage:

```
Plan$set_verbose(x)
```

Arguments:

`x` Logical.

Returns: x, invisibly.

`Plan$set_use_foreach()`: Set `use_foreach`, which `new()` describes.

Usage:

`Plan$set_use_foreach(x)`

Arguments:

x Logical or NULL.

Returns: x, invisibly.

`Plan$get_data()`: Load every data source, and return the datasets as a named list.

Usage:

`Plan$get_data()`

Returns: A named list with one element for each dataset, and an element `hash`. `hash$current` is the spookyhash digest of the datasets. `hash$current_elements` holds one digest for each dataset. `plnr` does not read these digests.

If the only data source is `data_-----go_up_one_level` and it returns a named list, that list holds the datasets.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
str(p$get_data(), max.level = 1)
```

`Plan$get_analysis()`: Get one analysis.

Usage:

`Plan$get_analysis(index_analysis)`

Arguments:

`index_analysis` The position of the analysis, or its name.

Returns: The analysis, as `analyses` holds it. Its `argset` also holds `index_analysis`.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$get_analysis("analysis_1")
```

`Plan$get_argset()`: Get the `argset` of one analysis.

Usage:

`Plan$get_argset(index_analysis)`

Arguments:

`index_analysis` The position of the analysis, or its name.

Returns: The argset, a named list.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$get_argset("analysis_1")
```

`Plan$get_argsets_as_dt()`: Get every argset as one row of a `data.table`.

Usage:

```
Plan$get_argsets_as_dt()
```

Returns: A `data.table`. Its first columns are `name_analysis` and `index_analysis`, and it has one more column for each argset element.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$get_argsets_as_dt()
```

`Plan$run_one_with_data()`: Run one analysis on data that you give it.

Usage:

```
Plan$run_one_with_data(index_analysis, data, ...)
```

Arguments:

`index_analysis` The position of the analysis, or its name.

`data` A named list of datasets, normally from `get_data()`.

`...` Further arguments for the action function. See 'Details'.

Returns: The value that the action function returns.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
```

```

    fn_name = "plnr::example_action_fn",
    l = list(
      list(name = "analysis_1", var_1 = 1),
      list(name = "analysis_2", var_1 = 2)
    )
  )
  data <- p$get_data()
  p$run_one_with_data("analysis_1", data)

```

Plan\$run_one(): Run one analysis. `run_one()` loads the data with `get_data()` each time you call it.

Usage:

```
Plan$run_one(index_analysis, ...)
```

Arguments:

`index_analysis` The position of the analysis, or its name.

`...` Further arguments for the action function. See 'Details'.

Returns: The value that the action function returns.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$run_one("analysis_1")

```

Plan\$run_all_with_data(): Run every analysis on data that you give it.

Usage:

```
Plan$run_all_with_data(data, ..., .plnr.options = NULL)
```

Arguments:

`data` A named list of datasets, normally from `get_data()`.

`...` Further arguments for the action function. See 'Details'.

`.plnr.options` A list of options for `plnr`. The action function does not get it. `chunk_size` goes to `foreach` as `.options.future = list(chunk.size = chunk_size)`. Only a future-based backend, such as `doFuture`, reads it. The default is 1.

Returns: A list, invisibly. Element `i` is the value that the action function of analysis `i` returns. Without `foreach`, a `NULL` value at the end of the plan adds no element, so the list can be shorter than the plan.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

```

```

p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
data <- p$get_data()
results <- p$run_all_with_data(data)

```

Plan\$run_all(): Run every analysis. `run_all()` loads the data once with `get_data()`, and passes it to every analysis.

Usage:

```
Plan$run_all(...)
```

Arguments:

... Further arguments for the action function, and `.plnr.options`. `run_all_with_data()` describes both.

Returns: A list, invisibly, as `run_all_with_data()` returns it.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
results <- p$run_all()

```

Plan\$run_all_progress(): Run every analysis, as `run_all()` does, inside `progressr::with_progress()`. A `progressr` progressor then shows progress. It stops with an error when the `progressr` package is not installed.

Usage:

```
Plan$run_all_progress(...)
```

Arguments:

... Passed to `run_all()`.

Returns: A list, invisibly, as `run_all()` returns it.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),

```

```

      list(name = "analysis_2", var_1 = 2)
    )
  )
  if (requireNamespace("progressr", quietly = TRUE)) {
    results <- p$run_all_progress()
  }

```

Plan\$run_all_parallel(): Run every analysis in forked processes with `pbmccapply::pbmccapply()`, which shows a progress bar. It loads the data once with `get_data()`. Windows cannot fork. There, `pbmccapply` sets `mc.cores` to 1, warns, and runs the analyses in sequence.

Usage:

```
Plan$run_all_parallel(mc.cores = getOption("mc.cores", 2L), ...)
```

Arguments:

`mc.cores` The number of processes.

`...` Further arguments for the action function. See 'Details'.

Returns: A list, invisibly. Element `i` is the value that the action function of analysis `i` returns.

Plan\$clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`vignette("plnr")` for the concepts. `vignette("adding_analyses")` for worked single-function and multi-function plans.

Other plan helpers: [expand_list\(\)](#), [get_anything\(\)](#), [is_run_directly\(\)](#), [set_opts\(\)](#)

Examples

```

## -----
## Method `Plan$add_data()`
## -----

p <- plnr::Plan$new()
data_fn <- function() {
  return(plnr::nor_covid19_cases_by_time_location)
}
p$add_data("data_1", fn = data_fn)
p$add_data("data_2", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_data("data_3", direct = plnr::nor_covid19_cases_by_time_location)
names(p$get_data())

## -----
## Method `Plan$add_argset()`
## -----

```

```

p <- plnr::Plan$new()
p$add_argset("argset_1", var_1 = 3, var_b = "hello")
p$add_argset("argset_2", var_1 = 8, var_c = "hello2")
p$get_argsets_as_dt()

## -----
## Method `Plan$add_argset_from_df()`
## -----

p <- plnr::Plan$new()
p$add_argset_from_df(data.frame(name = c("a", "b"), var_1 = c(1, 2)))
p$get_argsets_as_dt()

## -----
## Method `Plan$add_argset_from_list()`
## -----

p <- plnr::Plan$new()
p$add_argset_from_list(plnr::expand_list(var_1 = 1:2, var_2 = c("i", "j")))
p$get_argsets_as_dt()

## -----
## Method `Plan$add_analysis()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis(name = "analysis_1", fn_name = "plnr::example_action_fn")
p$run_one("analysis_1")

## -----
## Method `Plan$add_analysis_from_df()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_df(
  fn_name = "plnr::example_action_fn",
  df = data.frame(name = c("a", "b"), var_1 = c(1, 2))
)
p$run_one("a")

## -----
## Method `Plan$add_analysis_from_list()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)

```

```

    )
  )
  p$run_one("analysis_1")

  ## -----
  ## Method `Plan$apply_action_fn_to_all_argsets()`
  ## -----

  p <- plnr::Plan$new()
  p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
  p$add_argset_from_list(list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  ))
  p$apply_action_fn_to_all_argsets(fn_name = "plnr::example_action_fn")
  p$run_one("analysis_1")

  ## -----
  ## Method `Plan$get_data()`
  ## -----

  p <- plnr::Plan$new()
  p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
  str(p$get_data(), max.level = 1)

  ## -----
  ## Method `Plan$get_analysis()`
  ## -----

  p <- plnr::Plan$new()
  p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
  p$add_analysis_from_list(
    fn_name = "plnr::example_action_fn",
    l = list(
      list(name = "analysis_1", var_1 = 1),
      list(name = "analysis_2", var_1 = 2)
    )
  )
  p$get_analysis("analysis_1")

  ## -----
  ## Method `Plan$get_argset()`
  ## -----

  p <- plnr::Plan$new()
  p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
  p$add_analysis_from_list(
    fn_name = "plnr::example_action_fn",
    l = list(
      list(name = "analysis_1", var_1 = 1),
      list(name = "analysis_2", var_1 = 2)
    )
  )

```

```

p$get_argset("analysis_1")

## -----
## Method `Plan$get_argsets_as_dt()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$get_argsets_as_dt()

## -----
## Method `Plan$run_one_with_data()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
data <- p$get_data()
p$run_one_with_data("analysis_1", data)

## -----
## Method `Plan$run_one()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
p$run_one("analysis_1")

## -----
## Method `Plan$run_all_with_data()`
## -----

p <- plnr::Plan$new()

```

```

p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
data <- p$get_data()
results <- p$run_all_with_data(data)

## -----
## Method `Plan$run_all()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
results <- p$run_all()

## -----
## Method `Plan$run_all_progress()`
## -----

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = list(
    list(name = "analysis_1", var_1 = 1),
    list(name = "analysis_2", var_1 = 2)
  )
)
if (requireNamespace("progressr", quietly = TRUE)) {
  results <- p$run_all_progress()
}

```

set_opts

Set plnr options

Description

set_opts() sets options for the whole package.

Usage

```
set_opts(force_verbose = FALSE)
```

Arguments

`force_verbose` Logical. TRUE makes every [Plan](#) that you create afterwards verbose, even in a non-interactive session. The default is FALSE.

Value

`force_verbose`, invisibly.

See Also

`vignette("plnr")` for the concepts.

Other plan helpers: [Plan](#), [expand_list\(\)](#), [get_anything\(\)](#), [is_run_directly\(\)](#)

Examples

```
set_opts(force_verbose = TRUE)
set_opts(force_verbose = FALSE)
```

test_action_fn	<i>Return 1, whatever the input</i>
----------------	-------------------------------------

Description

`test_action_fn()` is a placeholder action function. It ignores its arguments.

Usage

```
test_action_fn(data, argset)
```

Arguments

<code>data</code>	Not used.
<code>argset</code>	Not used.

Value

The number 1, a double.

See Also

`vignette("plnr")` for what an action function is.

Other example and test functions: [example_action_fn\(\)](#), [example_data_fn_nor_covid19_cases_by_time_location\(\)](#)

Examples

```
test_action_fn(data = list(), argset = list())

p <- plnr::Plan$new()
p$add_data(name = "deaths", direct = data.frame(deaths = 1:4, year = 2001:2004))
p$add_analysis(name = "analysis_1", fn_name = "plnr::test_action_fn")
p$run_one("analysis_1")
```

try_again

*Retry code, with a random delay between attempts***Description**

try_again() evaluates x up to times times, and stops at the first attempt that raises no error. After each failed attempt it waits a number of seconds drawn from a uniform distribution on [delay_seconds_min, delay_seconds_max]. The delay does not grow with the attempt number, so this is not exponential backoff.

Usage

```
try_again(
  x,
  times = 2,
  delay_seconds_min = 5,
  delay_seconds_max = 10,
  verbose = FALSE
)
```

Arguments

x	The code to run. try_again() evaluates it once for each attempt.
times	The maximum number of attempts. The default is 2.
delay_seconds_min	The shortest delay in seconds. The default is 5.
delay_seconds_max	The longest delay in seconds. The default is 10.
verbose	Logical. TRUE reports each failed attempt as a warning, and a success after a retry as a message. The default is FALSE.

Details

try_again() is adapted from try_again() in testthat. It adds the delay and the verbose argument.

Value

TRUE, invisibly, after the first attempt that succeeds. When every attempt fails, try_again() signals the error of the last attempt.

See Also

vignette("plnr") for the concepts. try_again() itself is not part of the [Plan](#) workflow.

Other utilities: [create_rmarkdown\(\)](#)

Examples

```
try_again(1 + 1)

# Fail once, then succeed. The delays are 0 so that the example runs fast.
attempt_n <- 0
try_again(
  {
    attempt_n <- attempt_n + 1
    if (attempt_n < 2) stop("not ready yet")
  },
  times = 3,
  delay_seconds_min = 0,
  delay_seconds_max = 0
)
attempt_n
```

Index

- * **datasets**
 - `nor_covid19_cases_by_time_location`,
7
- * **example and test functions**
 - `example_action_fn`, 3
 - `example_data_fn_nor_covid19_cases_by_time_location`,
4
 - `test_action_fn`, 23
- * **plan helpers**
 - `expand_list`, 5
 - `get_anything`, 5
 - `is_run_directly`, 6
 - `Plan`, 8
 - `set_opts`, 22
- * **utilities**
 - `create_rmarkdown`, 2
 - `try_again`, 24

`base::get()`, 6

`create_rmarkdown`, 2

`create_rmarkdown()`, 25

`example_action_fn`, 3

`example_action_fn()`, 4, 23

`example_data_fn_nor_covid19_cases_by_time_location`,
4

`example_data_fn_nor_covid19_cases_by_time_location()`,
3, 23

`expand_list`, 5

`expand_list()`, 6, 7, 18, 23

`get_anything`, 5

`get_anything()`, 5, 7, 8, 18, 23

`is_run_directly`, 6

`is_run_directly()`, 5, 6, 18, 23

`nor_covid19_cases_by_time_location`, 4,
7

`Plan`, 2, 3, 5–7, 8, 23, 25

`set_opts`, 22

`set_opts()`, 5–7, 18

`test_action_fn`, 23

`test_action_fn()`, 3, 4

`try_again`, 24

`try_again()`, 3