

Package ‘plnr’

September 23, 2026

Title A Framework for Planning and Executing Analyses

Version 2026.9.23

Description A comprehensive framework for planning and executing analyses in R.

It provides a structured approach to running the same function multiple times with different arguments, executing multiple functions on the same datasets, and creating systematic analyses across multiple strata or variables.

The framework is particularly useful for applying the same analysis across multiple strata (e.g., locations, age groups), running statistical methods on multiple variables (e.g., exposures, outcomes), generating multiple tables or graphs for reports, and creating systematic surveillance analyses.

Key features include efficient data management, structured analysis planning, flexible execution options, built-in debugging tools, and hash-based caching.

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, purrr, R6, stats, tidyr, usethis, utils, uuid

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

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Richard Aubrey White [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0002-6747-1726>>)

Maintainer Richard Aubrey White <hello@rwhite.no>

Repository CRAN

Date/Publication 2026-09-23 19:10:02 UTC

Contents

create_rmarkdown	2
example_action_fn	3
example_data_fn_nor_covid19_cases_by_time_location	4
expand_list	5
get_anything	6
is_run_directly	6
nor_covid19_cases_by_time_location	7
Plan	8
set_opts	25
test_action_fn	26
try_again	27
Index	29

create_rmarkdown	<i>Create an example R Markdown project structure</i>
------------------	---

Description

create_rmarkdown() creates a complete example project structure for an R Markdown analysis that uses the plnr framework. It creates a standardized directory structure. It also creates example files that show how to use plnr for data analysis and for report generation.

Usage

create_rmarkdown(home)

Arguments

home Character string. The path where create_rmarkdown() creates the project.

Details

The created project includes:

- A main run.R script that initializes the project and demonstrates plnr usage
- Example analysis functions in the R directory
- A template R Markdown document
- Standard project directories (results, paper, raw)

Value

NULL. create_rmarkdown() creates files and directories under home.

See Also

`vignette("plnr")` for the framework the generated `run.R` uses. That `run.R` builds one [Plan](#). It then makes one `add_data()` call, and one `add_analysis()` call per output.

Examples

```
# Create a temporary directory for the example
temp_dir <- tempfile("plnr_example_")
create_rmarkdown(temp_dir)

# View the created structure
list.files(temp_dir, recursive = TRUE)

unlink(temp_dir, recursive = TRUE)
```

example_action_fn	<i>Example action function that shows the analysis structure</i>
-------------------	--

Description

`example_action_fn()` shows how to structure an action function for the `Plan` class. It prints the names of the data and `argset` components it receives.

Usage

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

Arguments

data	A named list that holds the datasets for the analysis.
argset	A named list that holds the arguments for the analysis.

Value

NULL. `example_action_fn()` prints information about the input data and `argset`.

See Also

`vignette("adding_analyses")` for worked single-function and multi-function plans. Those plans attach an action function to a plan by `fn_name`.

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

Examples

```
# Create a new plan
p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Create batch of argsets
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)

# Add analyses to plan
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)

# View argsets and run example
p$get_argsets_as_dt()
p$run_one("analysis_1")
```

example_data_fn_nor_covid19_cases_by_time_location
An example data_fn that returns a data set

Description

An example data_fn that returns a data set

Usage

```
example_data_fn_nor_covid19_cases_by_time_location()
```

See Also

vignette("adding_analyses"). It defines the same kind of zero-argument data_fn. It then attaches that data_fn to a plan with fn_name.

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

Examples

```
# A data function takes no arguments and returns one data set
d <- example_data_fn_nor_covid19_cases_by_time_location()
dim(d)

# Its intended use is as an `fn_name` passed to Plan$add_data()
```

```
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*Create a cross product of lists*

Description

`expand_list()` creates a cross product of multiple lists. It works like `tidyr::expand_grid()`, but you do not need to wrap the arguments in an extra `list()`. Use it to build combinations of analysis parameters.

Usage

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

Arguments

... Named arguments. Each one holds a vector or a list of values to combine.

Value

A list of lists. Each inner list holds one combination of values from the input arguments.

See Also

[Plan](#). Its `add_argset_from_list()` method takes this list. See `vignette("adding_analyses")`, which builds argsets with `plnr::expand_list()`, then applies one action function to all of them.

Examples

```
# Create combinations of parameters
combinations <- plnr::expand_list(
  a = 1:2,
  b = c("a", "b")
)

# View the combinations
str(combinations)

# Compare with tidyr::expand_grid
tidyr::expand_grid(list(
  a = 1:2,
  b = c("a", "b")
))
```

get_anything	<i>Get objects with package namespace support</i>
--------------	---

Description

get_anything() extends base::get() to support package namespace scoping, for example "pkg::var". Use it for package exports and for namespace-qualified objects.

Usage

```
get_anything(x)
```

Arguments

x	Character string that names the object to get. The name is either a simple object name, or a namespace-qualified name such as "pkg::var".
---	---

Value

The requested object.

See Also

vignette("plnr"). Its "Function Naming" section covers the fn_name strings that a [Plan](#) resolves with get_anything().

Examples

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

# Get a simple object (same as base::get)
x <- 1
get_anything("x")
```

is_run_directly	<i>Check whether code runs directly, or from within a function</i>
-----------------	--

Description

is_run_directly() reports whether the code runs directly in the global environment, or from within a function call. Use it during development and debugging. A function can then behave one way when you run it directly, and another way when a larger analysis plan calls it.

Usage

```
is_run_directly()
```

Value

A logical value. TRUE means the code runs directly, from the global environment. FALSE means the code runs from within a function call.

See Also

`vignette("plnr")`. Its "Debugging Tools" section calls `is_run_directly()` inside an action function, to load data and `argset` while you develop that function.

Examples

```
# When run directly
is_run_directly() # TRUE

# When run from within a function
test_fn <- function() {
  is_run_directly() # FALSE
}
test_fn()
```

`nor_covid19_cases_by_time_location`

Covid-19 data for PCR-confirmed cases in Norway (nation and county)

Description

The Norwegian Surveillance System for Communicable Diseases (MSIS) supplies this data. The date corresponds to when the PCR-test was taken.

Usage

```
nor_covid19_cases_by_time_location
```

Format

A `csfmt_rts_data_v1` with 11028 rows and 18 variables:

granularity_time day/isoweek
granularity_geo nation, county
country_iso3 nor
location_code norge, 11 counties
border 2020
age total
isoyear Isoyear of event
isoweek Isoweek of event

isoyearweek Isoyearweek of event
season Season of event
seasonweek Seasonweek of event
calyear Calyear of event
calmonth Calmonth of event
calyearmonth Calyearmonth of event
date Date of event
covid19_cases_testdate_n Number of confirmed covid19 cases
covid19_cases_testdate_pr100000 Number of confirmed covid19 cases per 100.000 population

Details

The data records the raw number of cases, and the number of cases per 100.000 population.

The extraction date of this data is 2022-05-04.

Source

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

Plan

R6 Class for Planning and Executing Analyses

Description

The Plan class organizes and runs multiple analyses on one or more datasets. It enforces a structured approach to analysis in three ways.

1. Data Management:

- Load data once and reuse it across analyses
- Keep data cleaning separate from analysis
- Track data changes with a hash

2. Analysis Structure:

- Require all analyses to use the same data sources
- Standardize analysis functions to accept only data and argset parameters
- Organize analyses into clear, maintainable plans

3. Execution Control:

- Support both single-function and multi-function analysis plans
- Run analyses either in sequence or in parallel
- Supply built-in debugging tools

Details

The framework uses three main concepts:

- **Argset:** A named list that holds a set of arguments for an analysis
- **Analysis:** A combination of one argset and one action function
- **Plan:** A container that holds one data pull and a list of analyses

Public fields

`analyses` List of analyses. Each analysis holds one argset and one action function.

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 instance.

Usage:

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

Arguments:

`verbose` Logical. Whether to show verbose output. The default is TRUE in interactive mode, or when `config$force_verbose` is TRUE.

`use_foreach` Logical. Whether to use `foreach` for parallel processing. NULL lets the program decide. FALSE uses a loop. TRUE uses `foreach`.

Returns: A new Plan instance.

`Plan$add_data()`: Add a new dataset to the plan.

Usage:

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

Arguments:

`name` Character string. The name of the dataset.

`fn` Optional. A function that returns the dataset.

`fn_name` Optional. A character string that names a function that returns the dataset.

`direct` Optional. The dataset object itself.

Returns: NULL. The method changes the plan in place.

Examples:

```
p <- plnr::Plan$new()

# Add data using a function
data_fn <- function() { return(plnr::nor_covid19_cases_by_time_location) }
p$add_data("data_1", fn = data_fn)

# Add data using a function name
p$add_data("data_2", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Add data directly
p$add_data("data_3", direct = plnr::nor_covid19_cases_by_time_location)

# View added data
p$get_data()
```

`Plan$add_argset()`: Add a new argset to the plan.

Usage:

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

Arguments:

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

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

Returns: NULL. The method changes the plan in place.

Examples:

```
p <- plnr::Plan$new()

# Add argsets with different arguments
p$add_argset("argset_1", var_1 = 3, var_b = "hello")
p$add_argset("argset_2", var_1 = 8, var_c = "hello2")

# View added argsets
p$get_argsets_as_dt()
```

Plan\$add_argset_from_df(): Add multiple argsets from a data frame.

Usage:

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

Arguments:

df A data frame. Each row is one new argset.

Returns: NULL. The method changes the plan in place.

Examples:

```
p <- plnr::Plan$new()

# Create data frame of argsets
batch_argset_df <- data.frame(
  name = c("a", "b", "c"),
  var_1 = c(1, 2, 3),
  var_2 = c("i", "j", "k")
)

# Add argsets from data frame
p$add_argset_from_df(batch_argset_df)

# View added argsets
p$get_argsets_as_dt()
```

Plan\$add_argset_from_list(): Add multiple argsets from a list.

Usage:

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

Arguments:

l A list of lists. Each inner list is one new argset.

Returns: NULL. The method changes the plan in place.

Examples:

```
p <- plnr::Plan$new()

# Create list of argsets
batch_argset_list <- list(
  list(name = "a", var_1 = 1, var_2 = "i"),
  list(name = "b", var_1 = 2, var_2 = "j"),
  list(name = "c", var_1 = 3, var_2 = "k")
)
```

```
)

# Add argsets from list
p$add_argset_from_list(batch_argset_list)

# View added argsets
p$get_argsets_as_dt()
```

Plan\$add_analysis(): Add a new analysis to the plan.

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 function to use for the analysis.

fn_name Optional. A character string that names the function to use.

... Further arguments to add to the argset.

Returns: NULL. The method changes the plan in place.

Examples:

```
p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Add analysis
p$add_analysis(
  name = "analysis_1",
  fn_name = "plnr::example_action_fn"
)

# View argsets and run analysis
p$get_argsets_as_dt()
p$run_one("analysis_1")
```

Plan\$add_analysis_from_df(): Add multiple analyses from a data frame.

Usage:

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

Arguments:

fn Optional. The function to use for all analyses.

fn_name Optional. A character string that names the function to use.

df A data frame. Each row is one new analysis.

Returns: NULL. The method changes the plan in place.

Examples:

```

p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Create data frame of analyses
batch_argset_df <- data.frame(
  name = c("a", "b", "c"),
  var_1 = c(1, 2, 3),
  var_2 = c("i", "j", "k")
)

# Add analyses from data frame
p$add_analysis_from_df(
  fn_name = "plnr::example_action_fn",
  df = batch_argset_df
)

# View argsets and run example
p$get_argsets_as_dt()
p$run_one(1)

```

`Plan$add_analysis_from_list()`: Add multiple analyses from a list.

Usage:

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

Arguments:

`fn` Optional. The function to use for all analyses.

`fn_name` Optional. A character string that names the function to use.

`l` A list of lists. Each inner list is one new analysis.

Returns: `NULL`. The method changes the plan in place.

Examples:

```

p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Create list of analyses
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)

# Add analyses from list
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",

```

```

    l = batch_argset_list
  )

# View argsets and run example
p$get_argsets_as_dt()
p$run_one("analysis_1")

```

Plan\$apply_action_fn_to_all_argsets(): Apply one action function to all the argsets.

Usage:

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

Arguments:

fn Action function.

fn_name Action function name. `p <- plnr::Plan$new() p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location") batch_argset_list <- list(list(name = "analysis_1", var_1 = 1, var_2 = "i"), list(name = "analysis_2", var_1 = 2, var_2 = "j"), list(name = "analysis_3", var_1 = 3, var_2 = "k")) p$add_argset_from_list(fn_name = "plnr::example_action_fn", l=batch_argset_list) p$get_argsets_as_dt() 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()` instead.

Usage:

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

Arguments:

fn Action function.

fn_name Action function name.

Plan\$x_length(): Number of analyses in the plan.

Usage:

```
Plan$x_length()
```

Plan\$x_seq_along(): Generate a regular sequence from 1 to the number of analyses in the plan.

Usage:

```
Plan$x_seq_along()
```

Plan\$set_progress(): Set an internal progress bar.

Usage:

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

Arguments:

pb Progress bar.

Plan\$set_progressor(): Set an internal progressor progress bar.

Usage:

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

Arguments:

pb progressor progress bar.

Plan\$set_verbose(): Set the verbose flag.

Usage:

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

Arguments:

x Boolean.

Plan\$set_use_foreach(): Set the use_foreach flag.

Usage:

```
Plan$set_use_foreach(x)
```

Arguments:

x Boolean.

Plan\$get_data(): Extract the data added with add_data() and return it as a named list.

Usage:

```
Plan$get_data()
```

Returns: A named list. Most elements come from add_data().

One extra named element is called 'hash'. 'hash' holds the data hashes of particular datasets and variables. digest::digest() calculates them with the 'spookyhash' algorithm.

'hash' holds two named elements:

- current (the hash of the entire named list)
- current_elements (the hash of the named elements within the 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$get_data()
```

Plan\$get_analysis(): Extract one analysis from the plan.

Usage:

```
Plan$get_analysis(index_analysis)
```

Arguments:

index_analysis Either an integer in 1:length(analyses), or a character string with the name of the analysis.

Returns: An analysis.

Examples:

```
p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
```

```

    list(name = "analysis_2", var_1 = 2, var_2 = "j"),
    list(name = "analysis_3", var_1 = 3, var_2 = "k")
  )
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$get_analysis("analysis_1")

```

`Plan$get_argset()`: Extract one argset from the plan.

Usage:

```
Plan$get_argset(index_analysis)
```

Arguments:

`index_analysis` Either an integer in `1:length(analyses)`, or a character string with the name of the analysis.

Returns: An argset.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$get_argset("analysis_1")

```

`Plan$get_argsets_as_dt()`: Get all argsets and return them as a data.table.

Usage:

```
Plan$get_argsets_as_dt()
```

Returns: A data.table that holds all the argsets within a plan.

Examples:

```

p <- plnr::Plan$new()
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$get_argsets_as_dt()

```

`Plan$run_one_with_data()`: Run one analysis. You supply the data.

Usage:

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

Arguments:

`index_analysis` Either an integer in `1:length(analyses)`, or a character string with the name of the analysis.

`data` A named list. You normally get it from `p$get_data()`.

`...` Not used.

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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
data <- p$get_data()
p$run_one_with_data("analysis_1", data)
```

`Plan$run_one()`: Run one analysis. The method gets the data from `self$get_data()`.

Usage:

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

Arguments:

`index_analysis` Either an integer in `1:length(analyses)`, or a character string with the name of the analysis.

`...` Not used.

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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$run_one("analysis_1")
```

`Plan$run_all_with_data()`: Run all analyses. You supply the data.

Usage:

```
Plan$run_all_with_data(data, ...)
```

Arguments:

`data` A named list. You normally get it from `p$get_data()`.

... Not used.

Returns: A list. Each element is 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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
data <- p$get_data()
p$run_all_with_data(data)
```

`Plan$run_all()`: Run all analyses. The method gets the data from `self$get_data()`.

Usage:

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

Arguments:

... Not used.

Returns: A list. Each element is 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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$run_all()
```

`Plan$run_all_progress()`: Run all analyses and show a progress bar. The method gets the data from `self$get_data()`.

Usage:

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

Arguments:

... Not used.

Returns: A list. Each element is 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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$run_all_progress()
```

Plan\$run_all_parallel(): Run all analyses in parallel. The method gets the data from `self$get_data()`.

This method works only on linux computers. It uses `pbmccapply` as the parallel backend.

Usage:

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

Arguments:

`mc.cores` The number of cores to use.

... Not used.

Returns: A list. Each element is the value that the action function 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 an introduction to argsets, analyses and plans. See `vignette("adding_analyses")` for worked single-function and multi-function plans.

Examples

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

```

p <- plnr::Plan$new()

# Add data using a function
data_fn <- function() { return(plnr::nor_covid19_cases_by_time_location) }
p$add_data("data_1", fn = data_fn)

# Add data using a function name
p$add_data("data_2", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Add data directly
p$add_data("data_3", direct = plnr::nor_covid19_cases_by_time_location)

# View added data
p$get_data()

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

p <- plnr::Plan$new()

# Add argsets with different arguments
p$add_argset("argset_1", var_1 = 3, var_b = "hello")
p$add_argset("argset_2", var_1 = 8, var_c = "hello2")

# View added argsets
p$get_argsets_as_dt()

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

p <- plnr::Plan$new()

# Create data frame of argsets
batch_argset_df <- data.frame(
  name = c("a", "b", "c"),
  var_1 = c(1, 2, 3),
  var_2 = c("i", "j", "k")
)

# Add argsets from data frame
p$add_argset_from_df(batch_argset_df)

# View added argsets
p$get_argsets_as_dt()

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

p <- plnr::Plan$new()

```

```

# Create list of argsets
batch_argset_list <- list(
  list(name = "a", var_1 = 1, var_2 = "i"),
  list(name = "b", var_1 = 2, var_2 = "j"),
  list(name = "c", var_1 = 3, var_2 = "k")
)

# Add argsets from list
p$add_argset_from_list(batch_argset_list)

# View added argsets
p$get_argsets_as_dt()

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

p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Add analysis
p$add_analysis(
  name = "analysis_1",
  fn_name = "plnr::example_action_fn"
)

# View argsets and run analysis
p$get_argsets_as_dt()
p$run_one("analysis_1")

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

p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Create data frame of analyses
batch_argset_df <- data.frame(
  name = c("a", "b", "c"),
  var_1 = c(1, 2, 3),
  var_2 = c("i", "j", "k")
)

# Add analyses from data frame
p$add_analysis_from_df(
  fn_name = "plnr::example_action_fn",
  df = batch_argset_df
)

```

```

)

# View argsets and run example
p$get_argsets_as_dt()
p$run_one(1)

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

p <- plnr::Plan$new()

# Add example data
p$add_data("covid_data", fn_name = "plnr::example_data_fn_nor_covid19_cases_by_time_location")

# Create list of analyses
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)

# Add analyses from list
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)

# View argsets and run example
p$get_argsets_as_dt()
p$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")
p$get_data()

## -----
## 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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",

```

```

    l = batch_argset_list
  )
  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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)

```

```

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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
data <- p$get_data()
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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)

```

```

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")
batch_argset_list <- list(
  list(name = "analysis_1", var_1 = 1, var_2 = "i"),
  list(name = "analysis_2", var_1 = 2, var_2 = "j"),
  list(name = "analysis_3", var_1 = 3, var_2 = "k")
)
p$add_analysis_from_list(
  fn_name = "plnr::example_action_fn",
  l = batch_argset_list
)
p$run_all_progress()

```

set_opts

Set package configuration options

Description

set_opts() sets package-wide options, such as the verbosity of output messages. It changes the internal configuration state of the package.

Usage

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

Arguments

force_verbose Logical. Whether to force verbose output messages, whatever the interactive state is. The default is FALSE.

Value

NULL. set_opts() changes the internal configuration of the package.

See Also

[Plan](#). Its verbose argument is on by default when the session is interactive, or when force_verbose is TRUE. See vignette("plnr") for an introduction to the framework.

Examples

```
# Enable verbose output
set_opts(force_verbose = TRUE)

# Disable verbose output
set_opts(force_verbose = FALSE)
```

test_action_fn	<i>Test action function that returns a constant value</i>
----------------	---

Description

test_action_fn() always returns 1. Use it to test the Plan framework.

Usage

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

Arguments

data	A named list that holds the datasets. This example does not use it.
argset	A named list that holds the arguments. This example does not use it.

Value

The integer 1

See Also

vignette("plnr") for the data/argset contract. An action function **MUST** accept at least the supplied data and argset values. It **MAY** take further arguments. A directly supplied fn receives the argset positionally, so its second formal need not be named argset.

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

Examples

```
# Called directly, it ignores both arguments and returns 1
test_action_fn(data = list(), argset = list())

# Its intended use is as a placeholder action function inside a plan
p <- plnr::Plan$new()
p$add_data(
  name = "deaths",
  direct = data.table::data.table(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 execution with a random delay between attempts*

Description

`try_again()` runs code multiple times, with a random delay between attempts. The delay is drawn from a uniform distribution on `[delay_seconds_min, delay_seconds_max]` before every retry. It does not grow with the attempt number, so this is not exponential backoff. Use it for transient failures in operations that a later attempt can succeed at, such as network requests or file operations.

Usage

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

Arguments

<code>x</code>	The code to run, as an expression.
<code>times</code>	Integer. The maximum number of attempts to make. The default is 2.
<code>delay_seconds_min</code>	Numeric. The minimum delay in seconds between attempts. The default is 5.
<code>delay_seconds_max</code>	Numeric. The maximum delay in seconds between attempts. The default is 10.
<code>verbose</code>	Logical. Whether to show progress information. The default is FALSE.

Details

`try_again()` is adapted from the `try_again` function in the `testthat` package. It adds features that control the retry behavior and the verbosity.

Value

TRUE, invisibly, after an attempt succeeds. `try_again()` throws an error with the last error message when every attempt fails.

See Also

`vignette("plnr")` for an introduction to the framework. `try_again()` is a general-purpose retry helper. It is not part of the [Plan](#) workflow.

Examples

```
# A call that succeeds on the first attempt returns immediately
print(try_again(1 + 1))

# A call that fails once and then succeeds on the second attempt. The delays
# are set to zero so the example runs instantly; they default to 5-10 seconds.
attempt_n <- 0
try_again(
  {
    attempt_n <- attempt_n + 1
    if (attempt_n < 2) stop("not ready yet")
    "succeeded"
  },
  times = 3,
  delay_seconds_min = 0,
  delay_seconds_max = 0
)

# The expression really was evaluated twice
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, [26](#)

create_rmarkdown, [2](#)

example_action_fn, [3](#)

example_action_fn(), [4](#), [26](#)

example_data_fn_nor_covid19_cases_by_time_location,
[4](#)

example_data_fn_nor_covid19_cases_by_time_location(),
[3](#), [26](#)

expand_list, [5](#)

get_anything, [6](#)

is_run_directly, [6](#)

nor_covid19_cases_by_time_location, [7](#)

Plan, [3](#), [5](#), [6](#), [8](#), [25](#), [27](#)

set_opts, [25](#)

test_action_fn, [26](#)

test_action_fn(), [3](#), [4](#)

try_again, [27](#)