Package 'pipr'

May 9, 2026

Title Client for the Poverty and Inequality Platform ('PIP') API

Version 1.4.0

Description An interface to compute poverty and inequality indicators for more than 160 countries and regions from the World Bank's database of household surveys, through the Poverty and Inequality Portal (PIP).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://worldbank.github.io/pipr/, https://github.com/worldbank/pipr

BugReports https://github.com/worldbank/pipr/issues

Suggests covr, testthat (>= 3.0.0), spelling, knitr, rmarkdown, markdown, callr, ggplot2, tidyr, ggthemes, forcats, scales, dplyr, readr

Language en-US

Imports arrow, attempt, curl, jsonlite, tibble, purrr, cli, rlang, utils, httr2, stringr, vroom

Depends R (>= 4.1.0)

Config/testthat/edition 3

Date 2025-12-17

NeedsCompilation no

Author Tony Fujs [aut], Aleksander Eilertsen [aut], Ronak Shah [aut], R.Andrés Castañeda [aut, cre], Giorgia Cecchinato [aut], World Bank [cph]

Maintainer R.Andrés Castañeda <acastanedaa@worldbank.org>

Repository CRAN

Date/Publication 2025-12-22 18:00:01 UTC

Contents

`args_to_string` *convert arguments and values of a function to a string to parse into other functions*

Description

convert arguments and values of a function to a string to parse into other functions

Usage

```
args_to_string(il)
```

Arguments

`il`	list from `as.list(environment())` right after the function is called.

Value

character

`build_request` *Build request version 2*

Description

Build request version 2

Usage

```
build_request(server, api_version, endpoint, ...)
```

Arguments

`server`	character: Server. For WB internal use only
`api_version`	character: API version
`endpoint`	character: PIP API endpoint
`...`	other parameters

Value

httr2 request

`build_request_old` *build_request, OLD version*

Description

build_request, OLD version

Usage

```
build_request_old(server, api_version, endpoint, ...)
```

Arguments

`server`	character: Server. For WB internal use only
`api_version`	character: API version
`endpoint`	character: PIP API endpoint
`...`	other parameters

Value

httr2 request

call_aux *call a table from .pip env*

Description

call a table from .pip env

Usage

```
call_aux(table = NULL)
```

Arguments

table	character: name of table in .pip env. If NULL, it displays the names of tables available in .pip env

Value

data frame of auxiliary table

Examples

```
# call one table

get_aux("gdp", assign_tb = TRUE, replace = TRUE) # PR 63
call_aux("gdp")

# see the name of several tables in memory
tb <- c("cpi", "ppp", "pop")
lapply(tb, get_aux, assign_tb = TRUE, replace = TRUE) # PR 63
call_aux()
```

change_grouped_stats_to_csv *Change the list-output to dataframe (Function from pipapi)*

Description

Change the list-output to dataframe (Function from pipapi)

Usage

```
change_grouped_stats_to_csv(out)
```

Arguments

out	output from wbpip::gd_compute_pip_stats

Value

dataframe

`check_api` *Check internet connection and API status*

Description

Check internet connection and API status

Usage

```
check_api(api_version = "v1", server = NULL)
```

Arguments

`api_version`	character: API version
`server`	character: Server. For WB internal use only

Value

character

Examples

```
## Not run:
check_api()

## End(Not run)
```

`datt_rural` *Datt (1998) grouped data for rural india, 1983*

Description

Dataset from Datt (1998) with grouped data for rural India in 1983.

Usage

```
datt_rural
```

Format

A data frame with 13 observations on the following 6 variables:

monthly_pc_exp Welfare range class

mean_monthly_pc_exp Mean welfare for given welfare range class

percentage_of_persons Percentage of individuals in given welfare class

L Cumulative welfare

p Cumulative population

area rural

@source Datt, G. (1998). See get_cp vignette.

`datt_urban` *Grouped data for urban india, 1983*

Description

Dataset from Sarvekshana N26 Vol 9 N 4, created by the author following Datt(1998) methodology with grouped data for urban India in 1983.

Usage

```
datt_urban
```

Format

A data frame with 13 observations on the following 6 variables:

monthly_pc_exp Welfare range class

mean_monthly_pc_exp Mean welfare for given welfare range class

percentage_of_persons Percentage of individuals in given welfare class

L Cumulative welfare

p Cumulative population

area urban

@source Sarvekshana N26 Vol 9 N 4, and Datt, G. (1998) for methodology. See get_cp vignette.

`delete_cache` *Deletes content of the cache folder*

Description

Deletes content of the cache folder

Usage

```
delete_cache()
```

Value

Side effect. Deletes files.

Examples

```
## Not run: delete_cache()
```

`display_aux` *Display available auxiliary tables*

Description

Display available auxiliary tables

Usage

```
display_aux(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  simplify = TRUE,
  server = NULL,
  assign_tb = TRUE
)
```

Arguments

version	character: Data version. See get_versions()
ppp_version	ppp year to be used
release_version	date when the data was published in YYYYMMDD format
api_version	character: API version
format	character: Response format either of c("rds", "json", "csv")
simplify	logical: If TRUE (the default) the response is returned as a tibble
server	character: Server. For WB internal use only
assign_tb	logical: Whether to assign table to .pip env. Default is TRUE

Value

invisible tibble with names of auxiliary tables

Examples

```
## Not run:
display_aux()

## End(Not run)
```

---

get_aux *Get auxiliary data*

Description

get_aux() Get an auxiliary dataset. If no table is specified a vector with possible inputs will be returned.

get_countries() Returns a table countries with their full names, ISO codes, and associated region code

Usage

```
get_aux(
  table = NULL,
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  simplify = TRUE,
  server = NULL,
  assign_tb = FALSE,
  replace = FALSE
```

```
)

get_countries(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_regions(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_cpi(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_dictionary(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_gdp(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_incgrp_coverage(
```

```
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_interpolated_means(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_hfce(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_pop(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_pop_region(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_ppp(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
```

```
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_region_coverage(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)

get_survey_means(
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  server = NULL
)
```

Arguments

table	Aux table
version	character: Data version. See `get_versions()`
ppp_version	ppp year to be used
release_version	date when the data was published in YYYYMMDD format
api_version	character: API version
format	character: Response format either of c("rds", "json", "csv")
simplify	logical: If TRUE (the default) the response is returned as a `tibble`
server	character: Server. For WB internal use only
assign_tb	assigns table to specified name to the `.pip` environment. If `FALSE` no assignment will performed. If `TRUE`, the table will be assigned to exactly the same name as the one of the desired table. If character, the table will be assigned to that name.
replace	logical: force replacement of aux files in `.pip` env. Default is FALSE.

Value

If `simplify = FALSE`, it returns a list of class "pip_api". If `simplify = TRUE`, it returns a tibble with the requested data. This is the default. Only for `get_aux()`, If `assign_tb = TRUE` or character, it returns TRUE when data was assign properly to .pip env. FALSE, if it was not assigned.

Functions

- `get_countries()`: Returns a table countries with their full names, ISO codes, and associated region code
- `get_regions()`: Returns a table regional grouping used for computing aggregate poverty statistics.
- `get_cpi()`: Returns a table of Consumer Price Index (CPI) values used for poverty and inequality computations. statistics
- `get_dictionary()`: Returns a data dictionary with a description of all variables available through the PIP API.
- `get_gdp()`: Returns a table of Growth Domestic Product (GDP) values used for poverty and inequality statistics.
- `get_incgrp_coverage()`: Returns a table of survey coverage for low and lower-middle income countries. If this coverage is less than 50%, World level aggregate statistics will not be computed.
- `get_interpolated_means()`: Returns a table of key information and statistics for all years for which poverty and inequality statistics are either available (household survey exists) or extra- / interpolated. Please see `get_dictionary` for more information about each variable available in this table.
- `get_hfce()`: Returns a table of Household Final Consumption Expenditure (HFCE) values used for poverty and inequality computations.
- `get_pop()`: Returns a table of population values used for poverty and inequality computations.
- `get_pop_region()`: Returns a table of total population by region-year. These values are used for the computation of regional aggregate poverty statistics.
- `get_ppp()`: Returns a table of Purchasing Power Parity (PPP) values used for poverty and inequality computations.
- `get_region_coverage()`: Return a table of regional survey coverage: Percentage of available surveys for a specific region-year.
- `get_survey_means()`: Returns a table of all available surveys and associated key statistics. Please see `get_dictionary` for more information about each variable available in this table.

Examples

```
## Not run:
# Get list of tables
x <- get_aux()

# Get GDP data
df <- get_aux("gdp")

# Get countries
df <- get_aux("countries")

# Display auxiliary tables
get_aux()
```

```
# Display and assign to .pip env the selected auxiliary table
get_aux(assign_tb = TRUE)

# Bind gdp table to "gdp" in .pip env
get_aux("gdp", assign_tb = TRUE)

# Bind gdp table to "new_name" in .pip env
get_aux("gdp", assign_tb = "new_name")


## End(Not run)
## Not run:
# Short hand to get countries
get_countries()

## End(Not run)
## Not run:
# Short hand to get regions
get_regions()

## End(Not run)
## Not run:
# Short hand to get cpi
get_cpi()

## End(Not run)
## Not run:
# Short hand to get dictionary
get_dictionary()

## End(Not run)
## Not run:
# Short hand to get gdp
get_gdp()

## End(Not run)
## Not run:
# Short hand to get incgrp_coverage
get_incgrp_coverage()

## End(Not run)
## Not run:
# Short hand to get interpolated_means
get_interpolated_means()

## End(Not run)
## Not run:
# Short hand to get hfce
get_hfce()

## End(Not run)
## Not run:
# Short hand to get pop
```

```
get_pop()

## End(Not run)
## Not run:
# Short hand to get pop_region
get_pop_region()

## End(Not run)
## Not run:
# Short hand to get ppp
get_ppp()

## End(Not run)
## Not run:
# Short hand to get region_coverage
get_region_coverage()

## End(Not run)
## Not run:
# Short hand to get survey_means
get_survey_means()

## End(Not run)
```

`get_cache_info` *Provides some information about cached items*

Description

Provides some information about cached items

Usage

```
get_cache_info()
```

Value

character.

Examples

```
## Not run: get_cache_info()
```

`get_cp` *Get Country Profiles*

Description

Get Country Profiles

Usage

```
get_cp(
  country = "all",
  povline = 2.15,
  version = NULL,
  ppp_version = 2017,
  release_version = NULL,
  api_version = "v1",
  format = c("arrow", "rds", "json", "csv"),
  simplify = TRUE,
  server = NULL
)
```

Arguments

`country`	character: A vector with one or more country ISO 3 codes or 'all'
`povline`	numeric: Poverty line
`version`	character: Data version. See `get_versions()`
`ppp_version`	ppp year to be used
`release_version`	date when the data was published in YYYYMMDD format
`api_version`	character: API version
`format`	character: Response format either of c("rds", "json", "csv")
`simplify`	logical: If TRUE (the default) the response is returned as a `tibble`
`server`	character: Server. For WB internal use only

Value

If `simplify = FALSE`, it returns a list of class "pip_api". If `simplify = TRUE`, it returns a tibble with the requested data. This is the default. Only for `get_aux()`, If `assign_tb = TRUE` or character, it returns TRUE when data was assign properly to .pip env. FALSE, if it was not assigned.

Examples

```
## Not run:
# One country, all years with default ppp_version = 2017
res <- get_cp(country = "AGO")

# All countries, povline = 1.9
res <- get_cp(povline = 1.9)

# All countries and years with default values
res <- get_cp()

## End(Not run)
```

`get_cp_ki` *Get Country Profiles Key Indicators*

Description

Get Country Profiles Key Indicators

Usage

```
get_cp_ki(
  country = NULL,
  povline = 2.15,
  version = NULL,
  ppp_version = 2017,
  release_version = NULL,
  api_version = "v1",
  simplify = TRUE,
  server = NULL
)
```

Arguments

`country`	character: A vector with one or more country ISO 3 codes or 'all'
`povline`	numeric: Poverty line
`version`	character: Data version. See `get_versions()`
`ppp_version`	ppp year to be used
`release_version`	date when the data was published in YYYYMMDD format
`api_version`	character: API version
`simplify`	logical: If TRUE (the default) the response is returned as a `tibble`
`server`	character: Server. For WB internal use only

Value

If simplify = FALSE, it returns a list of class "pip_api". If simplify = TRUE, it returns a tibble with the requested data. This is the default. Only for get_aux(), If assign_tb = TRUE or character, it returns TRUE when data was assign properly to .pip env. FALSE, if it was not assigned.

Examples

```
## Not run:
# One country, all years with default ppp_version = 2017
res <- get_cp(country = "IDN")

# All countries, povline = 1.9
res <- get_cp(country = "IDN", povline = 1.9)


## End(Not run)
```

---

get_gd *Get grouped stats*

Description

Get grouped stats from the PIP API.

Usage

```
get_gd(
  cum_welfare,
  cum_population,
  estimate = c("stats", "lorenz", "params"),
  requested_mean = NULL,
  povline = NULL,
  popshare = NULL,
  lorenz = NULL,
  n_bins = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  simplify = TRUE,
  server = NULL
)
```

Arguments

cum_welfare	numeric: Cumulative welfare values, expressed in shares. Any length. They should be monotonically increasing, and sum to 1.
cum_population	numeric: Cumulative population values, expressed in shares. Any length. They should be monotonically increasing, and sum to 1.

estimate	character: One of "stats", "lorenz", "params".
requested_mean	numeric: Requested mean.
povline	numeric: Poverty line. Required for estimate = "stats".
popshare	numeric: Proportion of the population living below the poverty line
lorenz	character: Lorenz curve methodology. Either "lb" or "lq".
n_bins	numeric: Number of bins. Required for estimate = "lorenz".
api_version	character: API version
format	character: Response format either of c("rds", "json", "csv")
simplify	logical: If TRUE (the default) the response is returned as a `tibble`
server	character: Server. For WB internal use only

Value

data.frame

Examples

```
## Not run:

datt_data <- data.frame(p = c(0.0092, 0.0339, 0.0850, 0.160, 0.2609, 0.4133,
                              0.5497, 0.7196, 0.8196, 0.9174, 0.9570, 0.9751,
                              1),
                        L = c(0.00208, 0.001013, 0.03122, 0.07083, 0.12808,
                              0.23498, 0.34887, 0.51994, 0.64270, 0.79201,
                              0.86966, 0.91277, 1))

# estimate = 'stats': retrieve poverty statistics.
stats <- get_gd(cum_welfare = datt_data$L, cum_population = datt_data$p,
                estimate = "stats",
                requested_mean = 19, # default is 1.
                povline = 2.15)  # default is 1.

# estimate = 'lorenz': retrieve Lorenz curve data points for a specified number of bins.

## Best lorenz curve methodolody selected by default:
lorenz <- get_gd(cum_welfare = datt_data$L,
                 cum_population = datt_data$p,
                 estimate = "lorenz",
                 n_bins = 100)  # must be specified, default is NULL.

## Specify lorenz curve methodology:
### Beta Lorenz ("lb")
lorenz_lb <- get_gd(cum_welfare = datt_data$L,
                 cum_population = datt_data$p,
                 estimate = "lorenz",
                 lorenz = "lb",
                 n_bins = 100)

### Quadratic Lorenz ("lq")
```

```
lorenz_lq <- get_gd(cum_welfare = datt_data$L,
                    cum_population = datt_data$p,
                    estimate = "lorenz",
                    lorenz = "lq",
                    n_bins = 100)

# estimate = 'params': retrieve regression parameters used for the lorenz curve estimation.
params <- get_gd(cum_welfare = datt_data$L,
                    cum_population = datt_data$p,
                    estimate = "params")

## End(Not run)
```

`get_pip_info` *Get PIP info*

Description

Get information about the API.

Usage

```
get_pip_info(api_version = "v1", server = NULL)
```

Arguments

`api_version`	character: API version
`server`	character: Server. For WB internal use only

Value

list

Examples

```
## Not run:
get_pip_info()

## End(Not run)
```

get_stats *Get poverty and inequality statistics*

Description

Get poverty and inequality statistics

Usage

```
get_stats(
  country = "all",
  year = "all",
  povline = NULL,
  popshare = NULL,
  fill_gaps = FALSE,
  nowcast = FALSE,
  subgroup = NULL,
  welfare_type = c("all", "income", "consumption"),
  reporting_level = c("all", "national", "urban", "rural"),
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("arrow", "rds", "json", "csv"),
  simplify = TRUE,
  server = NULL
)

get_wb(
  year = "all",
  povline = NULL,
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  api_version = "v1",
  format = c("rds", "json", "csv"),
  simplify = TRUE,
  server = NULL
)

get_agg(
  year = "all",
  povline = NULL,
  version = NULL,
  ppp_version = NULL,
  release_version = NULL,
  aggregate = NULL,
```

```
  api_version = "v1",
  format = c("rds", "json", "csv"),
  simplify = TRUE,
  server = NULL
)
```

Arguments

country	character: A vector with one or more country ISO 3 codes or 'all'
year	integer: A vector with one or more years or 'all'
povline	numeric: Poverty line
popshare	numeric: Proportion of the population living below the poverty line
fill_gaps	logical: If TRUE, will interpolate / extrapolate values for missing years
nowcast	logical: If TRUE, will return nowcast estimates.
subgroup	character: If used result will be aggregated for predefined sub-groups. Either 'wb_regions' or 'none'.
welfare_type	character: Welfare type either of c("all", "income", "consumption")
reporting_level	character: Geographical reporting level either of c("all", "national", "urban", "rural")
version	character: Data version. See `get_versions()`
ppp_version	ppp year to be used
release_version	date when the data was published in YYYYMMDD format
api_version	character: API version
format	character: Response format either of c("rds", "json", "csv")
simplify	logical: If TRUE (the default) the response is returned as a `tibble`
server	character: Server. For WB internal use only
aggregate	character: Aggregate name. See `get_aux("countries")` for available options.

Value

If `simplify = FALSE`, it returns a list of class "pip_api". If `simplify = TRUE`, it returns a tibble with the requested data. This is the default. Only for `get_aux()`, If `assign_tb = TRUE` or character, it returns TRUE when data was assign properly to .pip env. FALSE, if it was not assigned.

Examples

```
## Not run:
# One country-year
res <- get_stats(country = "AGO", year = 2000)

# All years for a specific country
res <- get_stats(country = "AGO", year = "all")
```

```
# All countries and years
res <- get_stats(country = "all", year = "all")

# All countries and years w/ alternative poverty line
res <- get_stats(country = "all", year = "all", povline = 3.2)

# Fill gaps for years without available survey data
res <- get_stats(country = "all", year = "all", fill_gaps = TRUE)

# Proportion living below the poverty line
res <- get_stats(country = "all", year = "all", popshare = .4)

# World Bank global and regional aggregates
res <- get_stats("all", year = "all", subgroup = "wb")

# Short hand to get WB global/regional stats
res <- get_wb()

# Short hand to get fcv stats
res <- get_agg(aggregate = "fcv", server = "qa")

# Custom aggregates
res <- get_stats(c("ARG", "BRA"), year = "all", subgroup = "none")

## End(Not run)
```

`get_versions` *Get versions*

Description

Get available data versions.

Usage

```
get_versions(api_version = "v1", server = NULL, simplify = TRUE)
```

Arguments

`api_version`	character: API version
`server`	character: Server. For WB internal use only
`simplify`	logical: If TRUE (the default) the response is returned as a `tibble`

Value

tibble or list

Examples

```
## Not run:
get_versions()

## End(Not run)
```

`parse_error_body` *parse_error_body*

Description

Helper function to parse error messages generated by the PIP API

Usage

```
parse_error_body(resp)
```

Arguments

`resp`	A httr response

Value

character

`pip_is_transient` *pip_is_transient*

Description

Helper function to determine if an error is due to the number of requests going over the rate limit

Usage

```
pip_is_transient(resp)
```

Arguments

`resp`	A httr response

Value

logical

retry_after *retry_after*

Description

Helper function to determine how much time to wait before a new query can be sent

Usage

```
retry_after(resp)
```

Arguments

resp	A httr response

Value

numeric

unnest_ki *Unnest the key indicators*

Description

Unnest the key indicators

Usage

```
unnest_ki(out)
```

Arguments

out	parsed and simplified output from cp-key-indicators endpoint

Value

data frame, unnested.

Functions

- `unnest_ki()`: takes the simplified output from cp-key-indicators endpoint and unnests it.

Index